Doctoral theses at NTNU, 2020:389

Doctoral thesis

Achim Gerstenberg

Prototyping Cyber-Physical Systems using Wayfaring

An Experiment and Insights for Early-Stage Development

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Kunnskap for en bedre verden

Achim Gerstenberg

Prototyping Cyber-Physical Systems using Wayfaring

An Experiment and Insights for Early-Stage Development

Thesis for the Degree of Philosophiae Doctor

Trondheim, December 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Engineering
Department of Mechanical and Industrial Engineering

© Achim Gerstenberg

ISBN 9788232651245 (printed ver.)
ISBN 9788232651252 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2020:389

Printed by NTNU Grafisk senter

Abstract

The two main aims of the thesis are firstly to qualitatively and quantitatively research the effect of using a prototype-driven wayfaring approach for the early-stage development of cyber-physical systems to give normative recommendations to developers and secondly to show and discuss how this kind of research with human subjects can be conducted and improved.

Many development methodologies rely on early definition of requirements and design specifications and lack the agility to make fast design adaptations. However, this agility is necessary to cope with unforeseen design problems that arise during the early stages of highly uncertain complex development projects. Prototype-driven approaches, that rely on frequent testing and evaluation of prototypes for advancing the design, offer the necessary agility but research is required to better understand what influences the outcome of such a development process and when it is appropriate to use it. I answer these questions in the context of developing cyber-physical systems by studying two case studies and a controlled quantitative experiment in the context of programming a mobile autonomous robot. The first case study of a small but multidisciplinary project is used to extend the wayfaring approach and shows how prototyping helps to discover and solve unforeseen problems, especially while merging different disciplines. The second case study explains how the wayfaring approach can be applied in highly uncertain large-scale development projects as shown here with the example of the ATLAS detector development at CERN.

The controlled experiment quantitatively explores whether early testing of prototypes has an influence on the performance of the design outcome and also explores qualitatively what influences this performance. The research shows that a wayfaring-like approach can be suitable during the explorative concept evaluation phase of solving highly uncertain engineering problems.

Testing prototypes and abductively reflecting on the test results helps to discover unforeseen design errors and in finding possible solutions. However, the experiment could not show a statistically significant performance difference between participants who tested their designs early and often compared to participants who could not test their design early. It showed large individual performance differences between participants regardless of their experimental condition. Qualitatively, these individual differences can be explained by variations in error finding ability which not only depends on the participant's mental ability and programming experience but also on when, how and what the participants test and how the code is written. Furthermore, the results suggest that urged early testing without sufficient questioning of design choices can lead to design fixation supposedly enhanced by the sunk cost of building and testing prototypes. The insights lead to the normative suggestions to use early low-resolution prototyping to keep the sunk costs to a minimum, prototype only the critical functions of several disciplines quasi-simultaneously, prototype different concepts in parallel and prioritize the critical functions whose solutions appear to be least likely to succeed.

Well controlled quantitative human subject experiments are necessary to better understand, quantify and compare design methodologies. Such quantitative experiments in design methodology research are rare and often not controlled enough to unambiguously attribute the observed results to the stimulus. This thesis uses the above mentioned robot experiment as a case to show measures like automating the experiment procedures or avoiding direct human-human interaction between the experimenter and the participant to increase repeatability and internal validity of the experiment. The thesis also discusses lessons learned and when using the wayfaring approach is appropriate for developing such experiments.

Acknowledgements

First and foremost I want to thank Martin Steinert for being a very helpful, motivating and patient supervisor as well as the founder of TrollLabs that gave me this unique opportunity to make the switch from physics to early-stage product development. It was beautiful to be part of TrollLabs from almost the beginning and see it evolve to what it has become now.

I greatly appreciate the efforts of Cecilia Haskins for chairing my defense and Tobias Larsson and Malte Jung for taking their time to read my thesis, give constructive feedback and be my opponents at the defense. I wish Covid would not prevent you from coming to Trondheim. I thank Matilde, Carl Christian and Jørgen for being part of an awesome ME310 team and great colleagues, Carlo and Heikki for honest, sharp and constructive feedback and being delightful and fun office partners as well as many other Trolls, namely Stephanie, Matilde, Andreas, Kristoffer, Jørgen and Jørgen, Matt, Yngve, Evangelos, Marius, Henrikke, Håvard, Torjus, Pasi and Sampsa who were a pleasure to work with.

I also want to thank my flatmates for letting me take over the upstairs livingroom as an office and in particular Are and Else for giving me emotional support and putting up a ”PhD on track” sticker above my bed as a constant reminder. I can soon pass it on to Are.

Last but not least I want to thank my mom for many years of lovingly supporting me from a distance, giving me her opinions and being the person who helped me with a lot of proof reading.

Thanks also to my new home Norway with its splendid nature which gave me the opportunity for great hikes, sailing and skiing.

Tusen takk!

Preface

This thesis has been submitted to the Norwegian University of Science and Technology (NTNU) for the degree of Philosophiae Doctor (PhD). All work has been conducted at the Department of Mechanical and Industrial Engineering (MTP) at NTNU, more specifically within TrollLABS by Achim Gerstenberg. The research was supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Self-driving cars, mobile autonomous robots and robotic surgery are examples of cyber-physical systems (CPS) because they use computing (cyber) to interact context dependently with the physical world. CPSs use a combination of sensory inputs to react to and influence the physical world with actuators. Developing CPSs therefore requires developing the physical artifacts and the computational algorithms and combine them to achieve the desired contextual behavior in the physical world [Khaitan et al. 2015]. The interplay of the deterministic algorithms with the non-deterministic nature of the physical world can lead to chaotic robot behaviors and complicates the development of CPSs (see 2.3.2, 2.3.3). This thesis addresses how the prototype-driven approach of wayfaring (see 2.2.4) can be applied to the development of CPS and how to conduct research about the methodology of early-stage CPS development. Throughout the thesis the focus lies on the early stages of CPS development which is the phase of idea generation and assessment as well as concept development (see 2.2.1), i.e. exploring the solution space and finding and selecting concepts using prototype-driven approaches.

1.1 Research questions

The work of this doctoral thesis was conducted at TrollLABS. As it is common in this group, it focuses on the early-stages of highly uncertain engineering development projects and more concretely on the development of cyber-physical systems as well as how to do research on the development methodologies and behaviors. In the thesis I look into two research questions:

1. **How can a prototype-driven approach be applied to the early-stage development of cyber-physical systems and when is it appropriate to use this approach?**

   I address this research question with several qualitatively observing case-study projects and combine the qualitative findings with the results of a quantitative experimental study.

2. **How can a controlled quantitative experiment for researching early-stage development of cyber-physical systems be designed and conducted?**

   The second research question is on the meta level of the first research question. It focuses on insights gained from setting up and conducting the research necessary to answer the first research question. I use examples and lessons learned from conducting a quantitative experiment to illustrate how it can be done and what is important in such experiments.

1.2 Aim and target audience

This thesis is aimed at two different target audiences. At first it aims to advise developers of cyber-physical systems by giving normative procedural and behavioural recommendations that are helpful during the early concept creation and selection phase. These include recommendations on how to test prototypes and avoid common pitfalls based on insights from case studies and the experiment conducted within the framework of this thesis. The findings that are relevant for developers can be found in chapter 3. Secondly, the thesis is aimed at researchers in the field of engineering design methodology that research the development processes of cyber-physical systems. The thesis explains how and why to execute experimental details in quantitative experiments to achieve a high degree of parameter control and internal validity. I present and explain the design choices and lessons learned of the experiment conducted as part of this PhD so that researchers benefit from this examples for their own experiments in chapter 6.

1.3 Contributions

During my time as a PhD student I have contributed to 10 publications, 9 of which are published. I am the main author and contributor in all publications related to the quantitative experiment (contributions 5 to 10) and in the other publications I was either also the main author or the publication resulted from a master thesis that I co-supervised. In the following I list all publications I contributed to, give a short summary of each, how it contributes to this thesis and what my contribution within the publication is. Figure 1.1 shows the scientific contributions of the PhD categorized by content type and to which research question they contribute. The list below the figure is in chronological order.

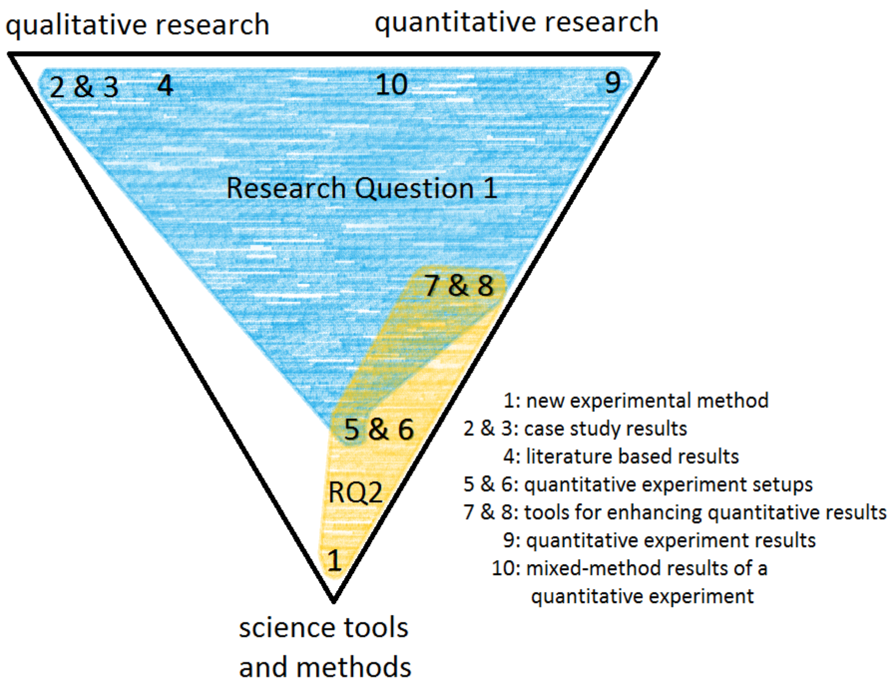

qualitative research
quantitative research
2 & 3
4
10
9
Research Question 1
7 & 8
5 & 6
RQ2
1
science tools and methods
1: new experimental method
2 & 3: case study results
4: literature based results
5 & 6: quantitative experiment setups
7 & 8: tools for enhancing quantitative results
9: quantitative experiment results
10: mixed-method results of a quantitative experiment

Figure 1.1: Overview of scientific contributions to this PhD work

Publication 1: published July 2015

"Distributed Experiments in Design Sciences, a Next Step in Design Observation Studies?"

Carlo Kriesi, Heikki Sjöman, Martin Steinert, ***Achim Gerstenberg****, and 42 other authors*

This publication describes an experimental setup that can be conducted by different researchers at different locations under comparable conditions with well-defined instructions and physical materials provided by the distributor (in this case Carlo Kriesi). The experiment was conducted in five different locations in four countries and the obtained data was sent back to the distributor. The aim of this approach is to increase external validity by decreasing the influence of local biases. This publication describes the underlying design task, the implementation of materials, instructions and sensors as well as encountered challenges of a distributed experiment. I was one of the researchers who received the experiment equipment, conducted the experiment according to the instructions and returned the acquired data to the experiment distributor. In addition, I contributed with intensive feedback as the experiment was designed and organised in TrollLabs. The publication is used in chapter 6 as examples for the importance of a well-defined experiment setup, how to control an experimental protocol and for how to enhance external validity of experiments.

Publication 2: published October 2015

"Bridging Tangible and Virtual Interaction: Rapid Prototyping of a Gaming Idea"

Thov Reime, Heikki Sjöman, ***Achim Gerstenberg****, Pekka Abrahamsson, and Martin Steinert*

This publication resulted from the master thesis of Thov Reime (supervised by myself, Heikki Sjöman and Martin Steinert). It describes development of a physical car model that can detect its assembled configuration, transfer this information to a computer where this physically assembled car model is digitally modelled to be used in a driving simulator game for children. The technical solution uses laser cut acrylic car parts, the Arduino micro controller board to measure resistances to detect the current assembly, and the "Processing" software to illustrate the assembly digitally. The task, which had to be done in a very short time frame, was worked on in wayfaring approach. The development process is described and reflected on in the publication by addressing the technical challenges and the implementation of the car model. It gives examples for prioritizing critical functions, low resolution prototyping, unknown unknowns and dynamically changing requirements and quasi simultaneous prototyping, early merging of disciplines and discovering interdependencies. This publication is used as a case-study in chapter 3.1.

Publication 3: published October 2015

"A Simultaneous, Multidisciplinary Development and Design Journey - Reflections on Prototyping"

Achim Gerstenberg, *Heikki Sjöman, Thov Reime, Pekka Abrahamsson, and Martin Steinert*

In this publication I demonstrate how to extend the wayfaring approach based on the observation of the course of action of Thov Reime in his master thesis project (see publication 2). Although Thov was supervised by Heikki Sjöman, Martin Steinert and myself, we intervened very little into the development process and did not instruct Thov to use the wayfaring approach. The observations are made retrospectively. The publication reveals in detail how unknown unknowns can be discovered and dealt with, repeated probing cycles lead to abductive learning and dynamically changing designs requirements and how quasi-simultaneous prototyping and early merging of different disciplines can discover and handle interdependecies. This publication is used as a case-study in chapter 3.1 for explaining the extended wayfaring mindset.

The one page summary behind this publication in the appendix C.3 is the poster presented at the conference.

Publication 4: published August 2017

"Large-scale Engineering Prototyping - Approaching Complex Engineering Problems CERN-style"

Achim Gerstenberg *and Martin Steinert*

This publication proposes how wayfaring can be applied to large-scale engineering projects with high degrees of uncertainty and therefore with dynamically changing design requirements. The publication points out the difficulties with modular top-down approaches for large-scale projects that require a high degree of flexibility and uses Philipp Türtscher's analysis of the development of the ATLAS detector at CERN [Türtscher 2008, Türtscher et al. 2014] and combines bottom-up approach with wayfaring showing how justification and interlaced knowledge enables wayfaring in large-scale projects. This publication is used and described in more detail in chapter 3.2.

Publication 5: published August 2018

"Open-ended Problems - A Robot Programming Experiment to Compare and Test Different Development and Design Approaches"

Achim Gerstenberg *and Martin Steinert*

This publication highlights the need for controlled quantitative experiments in engineering design research. It describes in detail an experimental setup for a human subject study investigating design research and provides two potential study concepts that could be realized with the experimental setup. A focus lies on how to control the interaction between the participant and the experimenter and quantitatively record the participant's behavior. The experimental setup uses an open-ended robot programming task with a quantifiable performance metric to test the influence of different stimuli on robot programming performance. In this thesis this publication serves two purposes: on the one hand it is a detailed description of the experimental setup reported in section 4.2 and and on the other hand it provides many examples for how to conduct a very controlled experiment and gives the rationale for the experimental design, both discussed in chapter 6.

Publication 6: re-submitted August 2020

"Testing the Effect of Desirable Difficulties on Teaching Robotics"

Achim Gerstenberg *and Martin Steinert*

This publication is based on the experiment setup described in publication 5 and adapts this setup for testing the effect of desirable difficulties on teaching robot programming and open-ended task solving capabilities. The paper presents the experimental design to test the hypothesis that participants who receive a flawed introduction and need to find and correct those perform better in a later open-ended task than participants who are introduced to the robot with unflawed instructions. The paper includes preliminary results and the discussing why continuing this experiment was not meaningful. This publication is used in chapter 6 as it is an example of how controlled quantifiable experiments can be designed and how preliminary results can lead to the development of a new experiment design. The new design became the concept that was then investigated thoroughly in this thesis.

Publication 7: published without peer-review March 2019

"Development and Verification of a Simulation for Leveraging Results of a Human Subjects Programming Experiment"

Achim Gerstenberg *and Martin Steinert*

In the experiment described in publication 5 achieving statistically meaningful results from only manually executing the robot programming solutions that the participants designed during the experiment is tedious, very time consuming and overall not very feasible. Therefore, the statistically power of the results was leveraged by using a robot simulation. I programmed such a simulation and this publication describes the development of this simulation for the physical robot as well as the validation that ensures that the simulation is representative of the physical robot behavior. This publication provides the details for replicating and understanding the simulation used to obtain the results reported in chapter 4 as well as in publication 9. In addition, the simulation serves as an example for how to enhance experimental results, in chapter 6.

Publication 8: published May 2019

"Evaluating and Optimizing Chaotically Behaving Mobile Robots with a Deterministic Simulation"

Achim Gerstenberg *and Martin Steinert*

This publication investigates the aspect of chaotical behaviour of autonomous mobile robots and the consequences for its development, programming and evaluation. The robot behavior depends on the interplay between the programmed algorithm and the robot hardware with the environment. Small perturbations in hardware and environment can lead to large deviations in the robot behavior and, therefore, quantifying the robot's performance requires statistical methods. With the simulation already mentioned in publication 7 the mean behavior of the robot and the robustness towards perturbations can be statistically studied which is important for providing similar conditions for all participants in the experiment described in chapter 4. Furthermore, it is of general relevance as a tool for rapid prototyping when developing mobile autonomous robots because a sensitivity analysis can reveal which algorithm and hardware changes are more or less sensitive to perturbation and thus require more attention during the robot development.

Publication 9: published May 2019

"The Relevance of Testing in Engineering Product Development - Investigations on a Robot Programming Task"

Achim Gerstenberg *and Martin Steinert*

In this publication the results of a human subject study investigating the influence of early testing in engineering product development are presented. In the study I applied the experimental setup described in publication 5 to test whether early testing of prototypes has a quantitatively measurable influence on the performance of the developed programming solutions for an autonomous mobile robot. I present the manually recorded results as well as the post-experimentally simulated results using the simulation described in publications 7 and 8 and discuss the outcome. In summary, no statistically significant evidence of performance difference depending on prototype testing during the early development stages was found. This publication is referred to in the results section of chapter 4.

Publication 10: published May 2019

"Fixation on Premature Concept Choices - a Pitfall of Early Prototyping?"

Achim Gerstenberg *and Martin Steinert*

While quantitatively testing the hypothesis described in publication 9, I serendipitly observed that only very few experiment participants changed their originally chosen concept to solve the robotic task. The participants who could not test their design early, and thus could not physically compare different concepts early, made superior concept choices compared to the participants that were allowed to test their designs and could in theory compare different concepts. From this observation I postulate that the sunk cost of building, testing, comparing and then rejecting unfavourable concepts leads to fixation onto the originally chosen concept. This lack of comparing concepts seemed to lead to premature and sometimes poor concept choices.

From this observation I recommend a development approach that has similarities to test-driven design in software engineering based on low resolution prototyping of critical functions to minimize sunk costs while still exploring many possible concepts. This publication is used and discussed in 4.9 and 5 and also serves as an example of how a serendipity finding can lead to a new hypothesis as discussed in chapter 6.

Publication 11: submitted for publication August 2020

"A low cost predictive display for teleoperations: investigating effects on human performance and workload"

*Henrikke Dybvik, Martin Løland, **Achim Gerstenberg**, Kristoffer Bjørnerud Slåttsveen and Martin Steinert*

Operating a remotely operated mobile ground vehicle often occurs through a video feed from the vehicle to the operator. Due to technical restrictions this video feed can be delayed by several seconds and this latency reduces the operators performance of operating the vehicle and increases the operator's mental workload. In this experiment we studied how a simple predictive display can influence operating performance and the operator's workload. The developed predictive display uses shifting and zooming of the video feed's field of view according the instructed movement that the robot is executing after the latency. This creates the illusion of an instantly moving robot. In addition an arrow indicates the robot orientation after the movement has completed. The experiment compares performance and workload with the inherent system delay of 250 milliseconds with a 700 milliseconds latency with and without the described predictive display. We find a statistically significant performance difference between with the 700 milliseconds latency with the predictive display compared to the 700 milliseconds latency condition without a predictive display. There was no significant difference in workload between the conditions.

The physical experiment setup used in this experiment was also developed using the wayfaring approach. Otherwise, this publication is not related to the topic of this PhD. I was involved by helping with the study design, initial data analysis and as a co-writer of the publication.

1.4 Structure of the thesis

This thesis consists of seven chapters and an appendix. The first chapter is this introduction that you now almost finished reading. The second chapter provides background information that I use in the later chapters. This background information is based on existing knowledge and therefore not part of my scientific contribution. The background chapter gives the framing of the type of early-stage and highly uncertain engineering projects this thesis applies to, introduces existing methods for the development of such kind of cyber-physical systems and provides the necessary information about the used research methodologies. In this second chapter I firstly explain mixed-method research as I use this approach when I combine the insights from case-studies and experimental data to answer the first research question, next the early-stage complex engineering design in the Fuzzy-Front End as it is typical for a PhD thesis within TrollLABS and thus the scope where this research applies and last but not least I define cyber-physical systems and the associated problems when developing them.

Chapters three to six include my contributions to research. In the third chapter I look at case studies to describe the prototype-driven development approach anecdotally and provide qualitative observations and suggestions. The first case study is a master project conducted at TrollLabs while the second part combines the prototype-driven approach with literature about the large-scale development of the ATLAS detector at CERN. Chapter four approaches the first research question from the experimental side. It presents the setup and mostly quantitative but also some qualitative results of a controlled experiment exploring the effect of early prototyping during the development of CPS. Chapter 5 combines the qualitative insights from the case studies, the quantitative results from the experiment and also the qualitative results from observing the behavior of the experiment participants into normative conclusions applicable for developers of complex cyber-physical systems.

The 6th chapter addresses the second research question on how to design and conduct experiments investigating the early-stage product development process of cyber-physical systems. I provide a brief history of how wayfaring was also used for the design of the study, how I arrived at the research question and on what study designs were tried and abandoned. Derived from the research presented in chapter 4 and 5 I show how to lay a focus on experimental control in such experiments with human subjects. I use and explain the rationale of experimental design decisions, how those are executed in detail and what the lessons learned are to further improve such experiment designs in the future.

The final and 7th chapter gives a summary of the thesis.

The appendix includes additional data that allows a more complete picture of the results of chapter 4 where I only present the most relevant data for better readability. I also added some of of the paper templates used in the experiment. In the end, the appendix includes a copy of the relevant publications written in the context of this thesis.

Chapter 2

Background information

In this chapter I include the background knowledge that is used later in the thesis. It is therefore shortened to what is needed and not a complete explanation of the theories. The information about mixed-method research is applied in the results of chapter 3 and used extensively for the discussions in chapter 5.

2.1 Mixed-method research

Research aims to develop and test scientific theories. These theories, that can be used to predict future outcomes, are made either inductively or deductively. Those predictions shall be falsifiable such that the prediction, sometimes also called hypothesis, can be confirmed or falsified by observations. If an observation falsifies a hypothesis the according theory is either wrong or incomplete [Popper 1935] and needs to be replaced or extended. The theories are created and extended by generalizing from observations. This is done through either statistical generalization (quantitative research) or analytical generalization (qualitative research).

Quantitative research is often applied to test hypotheses where an observation is predicted from an existing theory and then tested if it can reliably be reproduced with statistical significance. This implies that the measurements are numerical or boolean. If an outcome is observed repeatedly n times then by statistical generalization it is assumed that at the $(n+1)^{th}$ time the same outcome is observed if the conditions are the same as in the n observations before. The certainty of the prediction increases with the number of confirmed observations. This means that for making predictions from statistical generalization a deeper understanding of the phenomena is not necessary. The system can be researched like a black box where some stimulus is introduced (independent variable) and some output is observed (dependent variable). By repeating this process we can calculate correlations between the independent variable and the dependent variable and the likelyhood that these observations occurred by chance. Since the predictions only rely on statistics and not on logical arguments the method is immune to logical mistakes. However, since we have not gained any logical understanding of the system we cannot analytically argue how a modified system that differs from the previously studied system reacts to the stimulus. Therefore, it is essential to replicate the system exactly to reproduce the result. If the influence of an independent variable onto a dependent variable is to be tested then all other confounding parameters need to remain unchanged. The advantage of quantitative research is that a theory can be built without understanding and holds true as long as no evidence that refutes the theory is found. However, this comes at the price that the generalizations are very specific and the scope of the theory is narrow because a theory is only valid under the circumstances it was developed. If induction is used to discover a pattern one has to be careful to not generalize too quickly. It is unlikely that one specific pattern is statistically significant by pure chance but it is far more likely that some pattern out of all possible observable patterns is statistically significant by chance (i.e. p-hacking). This means that if a pattern is discovered from repeated evidence one can only be certain that this discovery is valid after explicitly repeating the experiment to specifically test the new hypothesis. Only if the hypothesis cannot be refuted the finding can be confirmed.

In contrast to quantitative research **qualitative research** is based on deducing predictions analytically from existing theories and developing or extending theories by analytical generalization through explaining the observations with the most likely and logically plausible explanation. Therefore, qualitative research can rely on single or few observations. The predictions are made deductively by combining the theory with observable premises. This means that both the theory can be logically flawed and premises can be incorrectly assumed. Since predictions are made by logically argumenting it is possible to use logic to adapt existing theories, that were developed and tested under certain other circumstances, to then fit under new circumstances. In explanatory case studies the theory is tested by pattern matching [Eisenhardt 1989, Yin 2017], that is the researchers predict a cause-effect relation and explain it with an existing theory and then verify if the predicted pattern can be observed in the cases. Qualitative research can afford to collect non-numerical data since it does not need to be evaluated statistically. Since the aim often is to get a deeper understanding instead of quantifying an effect it does not require that the observations are made with sufficient statistical power and under controlled and similar conditions. This opens up the possibility for interviews that follow the conversational flow, behavioral analysis from direct observation or videos or to take one-time occurrences into account. This is especially important in explorative case studies where the researchers try to abductively construct new logic models [Yin 2017, Burks 1946] from the observations.

Both in quantitative and qualitative research the validity of the study is important. The **construct validity** evaluates if the measured variables actually are representative to test or construct the theory. Construct validity can thus be increased by using multiple sources that investigate the same aspect of the theory. To make sure that the measured variables are representative it is important that the participants understand and follow the instructions as it is desired in the experiment design and the same way as other participants. It is also important that the participants are unaware of the hypothesis and cannot consciously or subconsciously change their behaviors respective to those variables. The same is true for the experimenter. An experimenter that is biased may alter how the experiment is conducted and what is measured. Therefore, it is good practice to use a double-blind setup where neither the participants nor the experimenter is aware of the hypothesis and the measured variables and correlations. If this is not possible or highly impractical the biasing effect of the experimenter can be decreased by strict procedures and automation of the experimental procedures such that the experimenter has less opportunity to influence the study.

Another threat to validity is to **internal validity** that is how an observation can also be explain differently by a rival theory. If a researcher cannot be certain that a theory is the only possible explanation to the observation then this theory is less internally valid. This can be the case if a confounding variable has a correlation with both the independent and dependent variable. In qualitative research it is therefore necessary to always ask if an observed pattern makes sense and can be further explained and if there are alternative theories based on other confounding variables that can cause the same outcome. If two variable correlate statistically significantly in quantitative research and they are the only parameters that are changed then the existence of a confounding variable can be ruled out. Keeping all parameters except for the independent and dependent variable constant is however sometimes not possible. Then it is important to measure any possible confounding variable. If this confounding variable correlates with the independent and dependent variable then a rival theory can be made that the confounding variable causes the observed result of the dependent variable. Let us for example assume that ice cream consumption and drowning deaths correlate. It is weird to believe that consuming ice cream causes drowning deaths and logically impossible that drowning causes ice cream consumption. The correlation can be explained by the weather. On warm days more people eat ice cream and more people go swimming and drown. The outside air temperature and sunshine are therefore confounding variables that correlate with both ice cream consumption and drowning deaths and explain the correlation. If we want to find out if ice cream consumption causes drowning without the confounding effect of the weather we could set up an experiment where we keep the air temperature and sunshine constant and half of the participants receive ice cream before swimming and the control group gets none. If no correlation is observed anymore we can rule out that ice cream consumption causes drowning and the weather remains as a possible explanation. The opposite experiment would even show an anti-correlation. Nobody that first drowned has then eaten ice cream afterwards.

Another aspect is **external validity**. This is if the findings of the study are generalizable beyond the study. The findings may be internally valid but if they do not apply in the real world the resulting theory is mostly useless. In qualitative research and especially in case studies external validity is usually high as those cases are taken from the real world and are not experiments designed to represent reality while still aiming for internal validity. However Yin as well as Lipset et al. [Yin 2017, Lipset et al. 1956] point out that the goal is to do a generalizing and not a particularizing analysis where the resulting theory can explain more that the particularly observed case. Another way of endangering external validity is by choosing study participants that poorly represent the population the resulting theory is supposed to be applied to; either actively by the researcher approaching or denying participants or through self-selection bias where by selecting themselves participants with certain traits that influence the study are more likely to participate. By sampling participants with widely different attributes the findings become more externally valid but introduce many new potentially confounding variables that are difficult to control. This trade-off between internal and external validity extends beyond participant selection as making an experiment design more controlled and internally valid in many cases means controlling parameters that would normally vary in real world contexts. The influence of those varying parameters can be statistically studied if the sample-size with a similar parameter value is sufficiently large. This is seldom the case. Then the influence can only be studied qualitatively and this is how mixed-method research comes into the picture.

Mixed-method research combines quantitative and qualitative research in a single study and Creswell and Clark [Creswell et al. 2007] define it as

> [...] collecting, analyzing and mixing both quantitative and qualitative data in a single study or series of studies. Its central premise is that the use of quantitative and qualitative approaches in combination provides a better understanding of research problems than either approach alone.

The main reason for using mixed-method research is that qualitative research can compensate the shortcomings of quantitative research and vice-versa. While quantitative research and statistical generalization provides little insights into the reasons why the results are measured, qualitative methods like interviews or observations can ask and provide those reasons. On the other hand, qualitative research relies on personal interpretations made by the researcher and can therefore be more easily biased. However, qualitative research can provide the deeper insights that can explain the quantitative results and helps developing rival theories and discovering confounding variables even if they do not statistically significantly correlate with the dependent variable. Furthermore, quantitatively developed theories apply only within the very similar context that they were constructed with and theories developed from qualitative research can use the found principles to extend and generalize the quantitatively found theories. In the ideal case quantitative and qualitative observations can cross-validate each other and thereby strengthen construct validity.

Lastly all research must be **reliable**; meaning that if the study is repeated under similar conditions the researchers will come to the same conclusions regardless if the same researcher or a different researcher repeats the study. This requires detailed documentation about the conditions and procedures of the study and publishing those such that other researchers can replicate the study.

2.2 The TrollLABS mindset

This thesis is written at TrollLABS. TrollLABS is a research lab within the Department of Mechanical and Industrial Engineering at the Norwegian University of Science and Technology. It provides a workshop where selected master students and PhDs can work on engineering projects and build early prototypes without much bureaucratic hurdles. The principle is that it is very fast and easy to try out ideas immediately. So on the one hand TrollLABS is a place where prototypes are built and on the other hand it is a research facility where the prototyping and development process is studied. The projects at TrollLABS are in the Fuzzy-Front-End of product development, usually complex and with a high degree of uncertainty because the developers try to solve this particular engineering problem for the first time and therefore have little prior knowledge about this particular problem. In this thesis the focus lies on the prototyping of cyber-physical systems.

2.2.1 Fuzzy-Front-End

Fuzzy Front End is the very beginning of the innovation process. It can also be described as phase 0 or idea generation and concept creation phase. If product development is regarded as a linear process with increasing fidelity over time, like Eppinger and Ulrich [Eppinger et al. 2011] and Herstatt and Verworn [Herstatt and Verworn 2004] do, then the Fuzzy-Front-End is the beginning part of this timeline. This product development process is shown in figure 2.1. The Fuzzy-Front-End consists of idea generation and assessment (phase 1) and concept development and planning (phase 2). In this linear process prototype development and testing comes as late as phase 4 and thus after the Fuzzy-Front-End. A prototype in this context is a high fidelity execution of the detailed planning in order to verify the planned functionality of the product before production. In contrast, the TrollLABS understanding of Fuzzy-Front-End includes prototype development and testing as a fundamental part of the Fuzzy-Front-End and a prototype is a physical artifact with the purpose to test an idea, gain understanding and thereby reduce uncertainty. It can be seen as an experiment that aims to answer if an idea can work with the purpose of learning and discovering unforeseen problems and benefits. Whenever I talk about prototyping from now on I mean this explorative kind of prototyping. In this context I define Fuzzy-Front-End as the part of the development process where the requirements for the solution are still unknown. This includes divergent thinking (how many ways can I solve this problem) and convergent analysis of prototypes (often rough evaluations of concepts based on testing prototypes).

2.2.2 Complicated and complex tasks

Another aspect that determines the development approach is the context of the problem and its solutions. In this thesis I use the Cynefin framework by Snowden and Boone [Snowden et al. 2007] to define if a project is complex or not. They categorize problems as simple, complicated, complex and chaotic. **Simple** problems have a clear cause-effect relation with one specific solution and can be solved by using best practices and routines because the situation can easily be understood, categorized and then re-

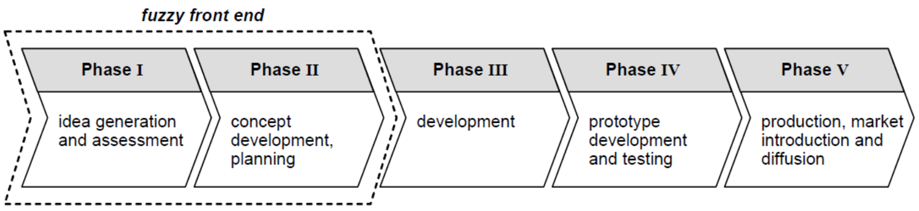

fuzzy front end
Phase I
idea generation and assessment
Phase II
concept development, planning
Phase III
development
Phase IV
prototype development and testing
Phase V
production, market introduction and diffusion

Figure 2.1: The Fuzzy Front End of product development as seen by Herstatt et al. [Herstatt, Stockstrom et al. 2006]

sponded to using rules based on prior experience. **Complicated** problems still have a clear cause-effect relationship and a solution is known to exist but it requires expertise to see and analyse the cause-effect relationships and find a solution. In the **complex** domain decisions need to be made based on incomplete data. More information is gained through experimentation (probing) and the solution emerges. Although cause-effect relationships exist and do not change they are not all known. This means that it is not a priori known if a solution exists, a solution is thus not imposable (plannable) and the development can only be explained logically in hindsight. These projects usually involve unknown unknowns; that is an unforeseeable problem without a known solution. Multiple master and PhD projects at TrollLABS are complex in this sense because it is unclear if a solution exists at all or how a solotion may look like at the beginning of the project [Blindheim 2019; Kriesi 2018; Kriesi, Steinert, Marmaras et al. 2019; Niedziela et al. 2014, Craig et al. 2015; Steinert 2020; Pidić et al. 2018; Sjöman, Kalasniemi et al. 2018; Sjöman, Soares et al. 2018; Sjöman, Autiosalo et al. 2018; Brede et al. 2019]. There are even engineering consultancies that specialize on complex problems and often reject projects where they are capable of naming a concept that can solve the problem before further engaging in the project [*Misty West - about webpage* 2019] (personal communication with the founder Josh Usher in Aug. 2017). **Chaotic** problems have cause-effect relationships that change and no persistent patterns emerge. Solving chaotic problems is not part of this thesis. This is not to be confused with the development of chaotically behaving robots which is mentioned later in the thesis.

2.2.3 Development methods

As engineering projects have different degrees of complexity Boone and Snowden suggest also different approaches to solving those problems [Snowden et al. 2007]. In complicated scenarios the solution may not be easily recalled but it can still be derived with expertise and the available information. It is therefore assumed that such solutions are plannable. The tasks can be clarified and defined and then system requirements and specifications can be determined theoretically at the beginning of the development process. The further development can then be segmented into modules [Pahl et al. 2007; VDI 1997; Eppinger et al. 2011; Cooper 1990]. The development process can be linear as shown for the systems approach by Pahl and Beitz [Pahl et al. 2007] in figure 2.2 and they describe it as a stepwise process with the goal of discovering the optimal solution although they say that returning to earlier steps may be needed if required earlier defined goals are not met. The first step is information gathering which includes market analysis, trend studies or already having known requirements to formulate the problem. Finding these requirements may come from the market analysis or trend studies but may also be provided to the developers through contractual agreements that are predefined before the development starts [Boehm 2000]. After the problem has been clearly formulated the developers can start with system implementation planning. Pahl and Beitz dedicate a section to "problem solving as information processing" where they describe how information is received (market analysis, patent, journals, etc.), processed (analysis, synthesis, calculation) and transmitted (drawings, tables, assembly and user manuals). In the systematic approach the focus lies on using existing information by finding a known pattern with an existing solution (inductively) or by analytically deducing the solution from theories (i.e. calculations). Validation of the found solution occurs towards later stages of the development.

Other sequential product development methods that use predefined requirements/specifications and assume plannability are V-model [Rook 1986], waterfall model [Royce 1987], and VDI guideline 2222 [VDI 1997]. Michael Schrage [Schrage 1993] describes these aforementioned methods as specification driven.

2.2.4 Wayfaring

The spec-driven development methods follow a hylomorphic mindset where an artifact is first thought an then formed. The making of an artifact becomes the materialization of a thought. Tim Ingold [Ingold 2016] suggest a different mindset he calls wayfaring for "finding one's way" in life and extends it to making of artifacts. According to Ingold, such thinking through making is based on mindful observations with a focus on improvisation. The next step of making emerges creatively from an improvised correspondence with the artifact. To improvise is to follow the ways of the world as they unfold [Ingold 2008] after carefully paying attention and accepting what is in order to build upon it. This entails to perceive the artifact without

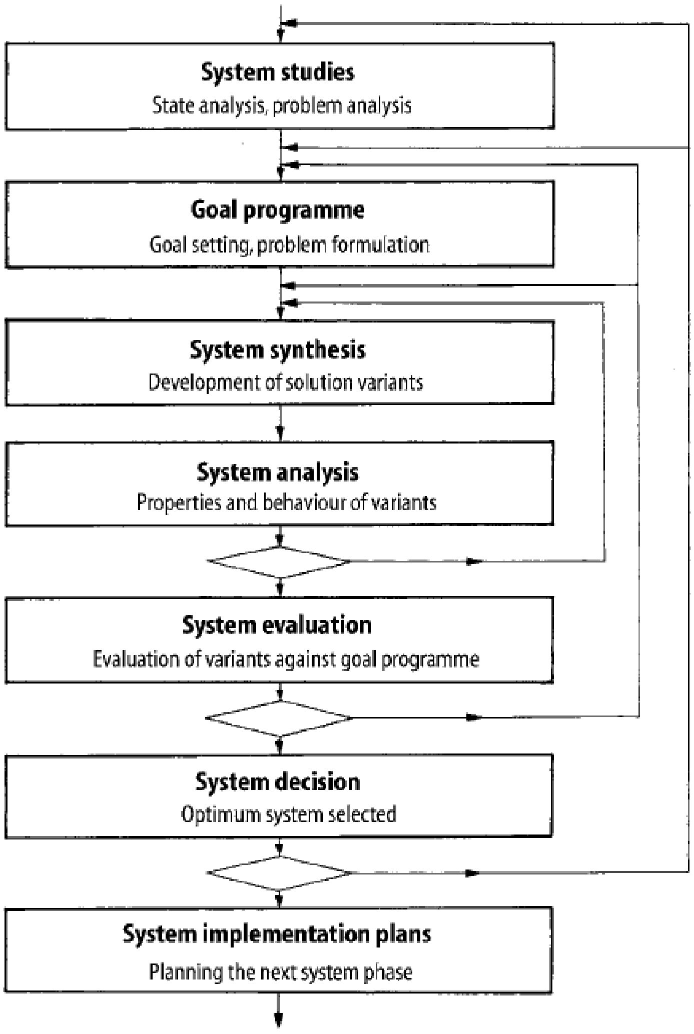

System studies
State analysis, problem analysis
Goal programme
Goal setting, problem formulation
System synthesis
Development of solution variants
System analysis
Properties and behaviour of variants
System evaluation
Evaluation of variants against goal programme
System decision
Optimum system selected
System implementation plans
Planning the next system phase

Figure 2.2: Steps of the systems approach [taken from Pahl et al. 2007]

a preconceived idea, whether novel or not, and instead being alert to the situation and feel the way ahead. It is an opportunistic journey guided by prototypes that also assumes that unpredictable problems will occur. Ingold compares this wayfaring process to a hunter being attentive to the environment and the animal tracks to find his way to the prey. In his book "Lines: a brief history" [Ingold 2016] Ingold describes two modes of movement along a line. One is wayfaring where the "line goes for a walk". The line of travel advances from its tip experiencing and actively engaging with the environment. This line is constantly adjusting its bearing, may stop and even return to a previous place and has no predefined final destination and thus is endless because there is always somewhere further to go to.

The other movement along a line Ingold calls transport. Transport, by contrast to wayfaring, is destination-oriented with the aim to relocate to a different position and thus complete the movement. Ingold describes the movement passively as the traveler "is transported" like a passenger without the aim of experiencing the way across the environment. Ingold further describes how knowledge is gained and used. Transport-like travel leads to and relies on mapping knowledge like modern maps that are spatial representations that allow the traveller to assemble a route-plan in the form of a chain of connections and thereby virtually to reach his destination even before setting out. It requires knowledge about the start and end point and the travel plan pre-exists its enactment on the ground. During wayfaring-like travel knowledge is gained along the way by observation of the environment. This gained knowledge can then immediately inform the traveller and determine the further path.

Schrage [Schrage 1999] introduces the term "serious play" where product developers play and improvise with prototypes being attentive and accepting to the outcome. Prototypes shall not primarily be used to confirm expectations but to discover what the developers need to know. Serious play in product development can follow pre-existing rules, principles and patterns but, just similar to play in games, it is openly accepting and inherently responsive to the current situation equal to the wayfaring mindset proposed by Ingold.

Steinert and Leifer [Steinert and Leifer 2012] as well as Edelman et al. [Edelman et al. 2012] embrace Ingold's concept of wayfaring and apply it to product development. Steinert and Leifer however introduce a small degree of planning as they begin their wayfaring product development with a development towards the "perceived target". However, the main idea of wayfaring, namely to observe the situation, adapt the further development accordingly and include serendipitly appearing opportunities, remains. The development path is guided by building prototypes, testing them and then abduct the next development from observing the testing of the prototype. Hence their understanding of wayfaring as a product development approach is congruent with Schrage's notion of a prototype-driven approach. This means that a design is finalized when the developers decide to stop the development and since this final design emerges from previous experiences it cannot be predicted. Since the development follows experiences instead of a plan there is no one correct or best way to design a product using wayfaring. This is similarly to an improvising musician. A different group of developers also using a wayfaring approach would likely have a different experience and thus develop something different. During improvisation, unlike performing sheet music, there is not a single correct way to play.

Edelman et al. furthermore observed that wayfaring behaviors of the developers facilitates radical breaks in a redesign task whereas a transport mindset leads to incremental redesign with a focus on optimization instead of "out of the box" thinking [Edelman et al. 2012]. Although wayfaring-like travel does not have one single bearing Ingold nonetheless implicitly assumes an overarching direction or purpose. In his example of a hunter who's path is determined by the landscape and the traces Ingold still assumes that finding a way forward is for the purpose of finding the prey. The hunter may be open to what this prey is and certainly where to go to find it but there is still an overarching purpose. Steinert and Leifer pick up this notion of hunting and when applied to development this means that the goal is to solve a technical problem. If you want to fly to Mars pure unguided wayfaring resulting in a submarine on wheels that holds a candle is of little use. Although the final design is not known until the developers decide that the current design is sufficient their development is guided by a vision. Once this design is discovered it can be translated into design specifications and then a specification-driven approach can be applied to optimize the design. Steinert and Leifer call this "bringing it home".

In case the final design can be defined from the beginning and the development can be planned then there is no need for wayfaring which would be less resource efficient and would likely not come to the same outcome. Wayfaring can include "dead-ends" where testing a prototype reveals that the chosen direction was not supporting the vision. On the other hand, developing with a wayfaring approach is inherently flexible and can adapt when the situation changes. Furthermore, wayfaring allows solving problems where no known solution exists and thus cannot be reached following a plan. Or in other words, a hunt cannot be planned on a map without knowing where the prey is. Therefore, it is assumed that using the wayfaring approach is especially useful in solving complex problems.

2.3 Cyber-physical systems

2.3.1 Definition of a Cyber-Physical System

In this thesis I define a cyber-physical system (CPS) as a system with multiple sensory inputs from the physical world used in a mathematical calculation (computation) that influences the behavior of the system in the physical world (physical to digital to physical). The output is typically through actuators like motors or loudspeakers. Those CPS often use feedback loops as those actuators influence their physical surrounding which is then again measured with the sensors that influence the computation.

A mobile autonomous robot is an example of a CPS that uses often multiple sensors to measure its surroundings, an algorithm that takes the sensor values and uses an imperative program code to compute motors commands to move the robot appropriately.

2.3.2 The problem of non-deterministic and chaotic behavior

The focus in this thesis lies on CPS that use a single-threaded imperative program to process the sensor data. These programs follow a single-threaded sequence of instructions (code) and are a deterministic model of how to execute the computations and how the system reacts to the computation results. This means that if the computation is repeated under precisely same conditions the result is unambiguous. However, in the real world the conditions that determine the input to the computation are not precisely reproducible. Therefore the combination of a single-threaded imperative program with a physical system is non-deterministic because the deterministic execution of the program depends on the non-deterministic physical environment. For example, the physics equations that are used in the control algorithm of an aerial drone are deterministic but the actual flight path is not.

Depending on the hardware of the CPS, the programmed code and the environment the CPS shows a behavior that can be sensitive to small perturbations in the state of the hardware or the environment [Nehmzow et al. 2003]. For example can a small difference in a sensory input cause the the code to execute vastly different instructions and thus the CPS behaves differently. This sensitivity to perturbation is one of the characteristics for chaotic behavior. This does not mean that the behavior is random. It means that the behavior still follows unambiguous instructions but it quickly becomes unpredictable over a short time. This is especially the case when the response to the sensory input is binary in that either a command is executed or not. For example the path and speed of an autonomous car on a highway is predictable for a longer time because fewer situations occur that require an either or action than the same car driving on a crowded city road where for example a traffic light causes a very binary decision like to stop or not.

This chaotic behavior can be quantified by comparing how different the behavior of a CPS is when it is repeated from an almost similar state. For example Nehmzow and Walker let a mobile autonomous robot drive in different environments and compared the predictability over time [Nehmzow et al. 2003] by recording the track. Using an algorithm [Wolf et al. 1985] that quantifies how a fiducial track separates over time from a track captured at a later time but from almost similar position and orientation Nehmzow and Walker could quantify the influence of the environment on Lyapunov exponents that describe the sensitivity of the system to perturbations. One thing to note though is that a CPS with less sensitivity to perturbations and thus more predictable behavior is not necessarily desired. The path of the autonomous car stopping at a traffic light is vastly different from the car not stopping or an autonomous mobile lawn mowing robot that drives a less predictable route can still be more efficient at completely covering an unknown territory and thus cutting the lawn faster.

2.3.3 Implications for the development of Cyber-Physical Systems

When developing software it is important to test and verify that the system actually behaves the way it is intended to. This process is inherently more complex for CPS because it does not only depend on the deterministic code but also on the non-deterministic surroundings and hardware. While in a pure software project, like a calculator, a flaw is reliably reproducible and can be debugged by going through the clock cycles step by step and following the state of the processor at every clock-cycle, going through the code of a CPS step by step influences how the CPS reacts. This means that the flawed (and unflawed) behavior cannot be reproduced if the clock cycles are manually controlled. Therefore, the developers do not have the temporal in-depth look at which code is executed when the CPS executes a certain behavior. Instead they need to guess from qualitative observations of the behavior which part in the code caused each behavior. If the autonomous car keeps driving at a red traffic light they can only assume which part of the code (or hardware) is faulty. Finding a flaw then becomes increasingly difficult if this behavior is not repeatable and at a successive test the car stops at the red traffic light. While this can never happen with a deterministicly behaving system it can happen with CPS.

This problem also means that single or a low number of tests are not suitable to detect rarely occurring behaviors and that if the CPS is only tested once the observed behavior may not be a typical behavior. The non-deterministic behavior does not only complicate the qualitative observation and evaluation of the CPS but also the quantitative. With a deterministic system one measurement is sufficient to quantify the behavior but for non-deterministic systems the behavior needs to be evaluated statistically to find the average behavior, how much the behavior fluctuates and how likely undesired behaviors are to occur.

2.4 Research Methodology

The Design Research Methodology framework by Blessing and Chakrabarti Blessing et al. 2009 differentiates between 4 different research stages. These are research clarification, descriptive study 1, prescriptive study and descriptive study 2. The research clarification leads to a research goal and the corresponding research question. It is based on existing evidence, usually from literature, combined with assumptions. From this a research question and concrete hypothesis and criteria how it can be measured and falsified can be developed.
The descriptive study 1 is usually exploratory and has the aim to empirically better understand the researched phenomenon in order to develop a targeted support (intervention) that can positively impact the design process. The development and testing of the support is done during the prescriptive study and the descriptive study 2 aims to evaluate the impact of the support. Blessing and Chakrabarti point out that design methodology research can include some or all of these stages and does not necessarily need to begin with the research clarification.

In this PhD thesis I apply the different research stages to better understand and enhance the prototype-driven approach of Wayfaring and compare it to a spec-driven approach. Both the specification-driven approach described in subsection 2.2.3 as well as Wayfaring (subsection 2.2.4) are applicable approaches during the Fuzzy-Front-End of product development. However, the two approaches greatly differ in their way of using prototypes. The specification-driven methods rely on designing and planning based on combining existing knowledge in new ways following design guidelines and prototypes are used to confirm correct planning. Although Pahl and Beitz do not exclude earlier prototypes categorically the emphasis clearly is on theoretically designing and predicting the outcome with mental, analytical and digital models. Since the developers look for existing solutions the spec-driven approach minimizes the risk of "reinventing the wheel" and thereby is often regarded as efficient.

In contrast, prototype driven-approaches still use existing knowledge where it is already known to the developers or easily attainable but the design is guided by testing prototypes, observing and then abductively reasoning how the outcome can be explained and what this implies for the next design iteration.

This difference in use of prototypes between the two approaches begs the question of how prototypes influence the development and how the applicability of the approach depends on the design context. Since TrollLabs focuses on the early stages of complex systems development and I got involved in mechatronics and cyber-physical systems projects at TrollLabs it made sense to narrow down the research towards Wayfaring during the early development of complex cyber-physical systems. Hence, the first research question is "How can a prototype-driven approach be applied to the early-stage development of cyber-physical systems and when is it appropriate to use this approach?".

In order to answer this research question I use a combination of further research clarification and descriptive studies.

The research clarification is a literature study. Based on the literature I contribute by combining the wayfaring mindset with the literature about complex large-scale development of the ATLAS detector at CERN.

As a descriptive study, I empirically observed a master student using the wayfaring approach during his master thesis to better understand the influence of testing. The student was introduced to wayfaring in his studies but was neither influenced in choosing the wayfaring approach nor actively directed how to apply it. The study is based on reflections on the development process both from direct observation and from retrospectively asking the master student about his design rationale.

The prescriptive study looks if the support, which is allowing and encouraging experiment participants to test their prototypes early, can successfully be executed. In trial runs of the experiment I can confirm that allowing, actively reminding and encouraging participants to test leads to early testing of prototypes. Hence, the working of the support is confirmed in this short prescriptive study.

The following descriptive study 2 is a direct quantitative experimental comparison under controlled conditions between a spec-driven approach relying on planning and late confirmatory testing of prototypes against a prototype-driven approach. In this descriptive study I evaluate the influence of early testing on task solving performance. The same study also serves as an exploratory descriptive study to better understand the behaviors associated with testing. It is important to understand that the descriptive study that is quantitative evaluating the support relies on hypothesis testing and statistically comparing predefined measures. Some of these are the quantifiable performance metric, how often and when the participants tested their prototypes, how much code they type between tests and the participant's programming experience. While this experiment setup is mainly meant quantitative for hypothesis testing it is simultaneously used as an exploratory descriptive study that qualitatively looks at the test outcomes, the participant's behaviors and tries to explain the quantitative observations. This follows the mixed-method research approach described in 2.1. It is from this qualitative observations I mainly conclude when it is appropriate to apply the wayfaring approach.

The second research question is "how can a controlled quantitative experiment for research-ing early-stage development of cyber-physical systems be de-signed and conducted?". This includes how to control the instruction interaction with the experiment participants and automating data capture, enhancement and analysis. It arose because I want to develop a highly controlled experimental setup to answer the first research question but there was little literature and reference experiments within design research that I could use as inspiration. Therefore, developing a controlled experiment setup for studying the first research question became in itself exploratory research. The second research question is answered with a descriptive study that is self-reflecting on the experiment setup development and testing. The descriptive study does not answer or even quantify the effects of designing an experiment in this way on the experiment outcome but shares the qualitative insights.

Chapter 3

Anecdotal insights from prototyping Cyber-Physical Systems

This chapter delivers the qualitative part for answering the first research question. It uses a master project at TrollLabs as a case study. It describes the application of the prototype driven wayfaring approach in a concrete example and showcases some new ways the wayfaring approach can be extended. This case study is based on contributions 2 and 3, where the first describes the technical aspects of the project and the latter the methodological aspects of extending the wayfaring approach.

The other part of this anecdotal chapter explores how the wayfaring approach can be and in parts was applied in a large-scale development project like the ATLAS detector at CERN. It combines the literature work by Philipp Türtscher [Türtscher 2008, Türtscher et al. 2014] about the ATLAS detector development with the ideas from wayfaring.

3.1 Qualitative insights from the Fibo car case

As hinted earlier in the section about development methods, wayfaring relies on exploratory prototyping and abductive reasoning from test results to influence the next design iteration. Consequently can the final result not be predicted at the beginning of the development. This dynamic emergence of requirements is illustrated in figure 3.1 where the developers start at point A and, at the beginning, envision a solution at point V. Therefore, they start building a first prototype into this direction to test their assumptions. From the test results it becomes eminent that their initially imagined solution cannot work out as planned and the solution needs a redesign. The circles in this figure represent a probing cycle. A probing cycle consists, as depicted in figure 3.2, out of designing, building and testing a prototype. It starts with designing a new prototype based on the finding from testing during the previous probing cycle. The design phase is first divergent as in ”how many ways can we solve the problem” and then convergent by analytically reasoning which concept(s) to build and test. Several layers of circles represent different disciplines or domains in the project. One prototype can include all disciplines but sometimes it is meaningful to branch the domains to explore different concepts within just one or a few domains separately but quasi-simultaneously in order to merge them again later.

It can also happen that testing a prototype reveals that the chosen concept is not worth pursuing further and the wayfaring reaches a dead end. In this case the developers can either go back to a previous prototype and chose a different concept from there or abandon the project completely. Thereby, wayfaring combines well with real option theory [Abad et al. 2015] and can safe time and resources by finding out sooner why it is not meaningful to continue a project. Since the development outcome is influenced by the intermediate prototypes, the final result at point V^x is usually not at the initially envision point V. When the design requirements and specifications are explored to the point where the developers can be confident that the discovered concept is a satisfactory solution for the project, a specification-driven approach like Scrum, Pahl-Beitz, VDI-model 2221, V-model etc. can be applied to redesign and optimize the chosen concept into a finished product.

This wayfaring approach was used in the Fibo car project and this case serves to illustrate and extend it.

3.1.1 What is the Fibo car project?

The end result of the Fibo car project is a tangible user interface for configuring a car model that can electronically detect its own configuration, transfer this information to a computer with the aim to simulate this car model configuration in a digital car racing game. The project started with a cooperate sponsor who was interested in developing an interactive toy for STEM education for children and young teenagers. At a first meeting the idea was brought up as ”something like Kerbal Space program but for cars and configurable in the real world”. The aim was to build a first version of this tangibly configurable car model for early user testing and for illustration of the idea to potential investors within a few months. The development was therefore time critical and contributions 2 and 3 describe the first 6 weeks of this development process. The project was mainly

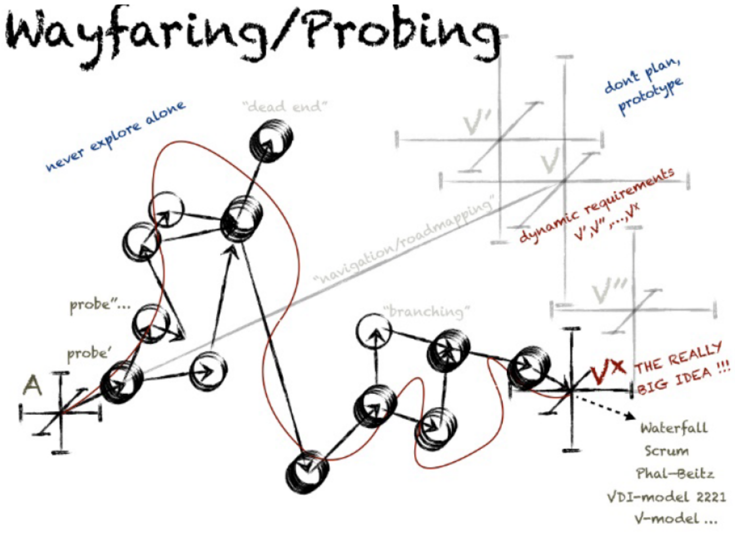


Figure 3.1: Overview of a multidisciplinary wayfaring journey from A to V^x. Figure taken from publication 3.

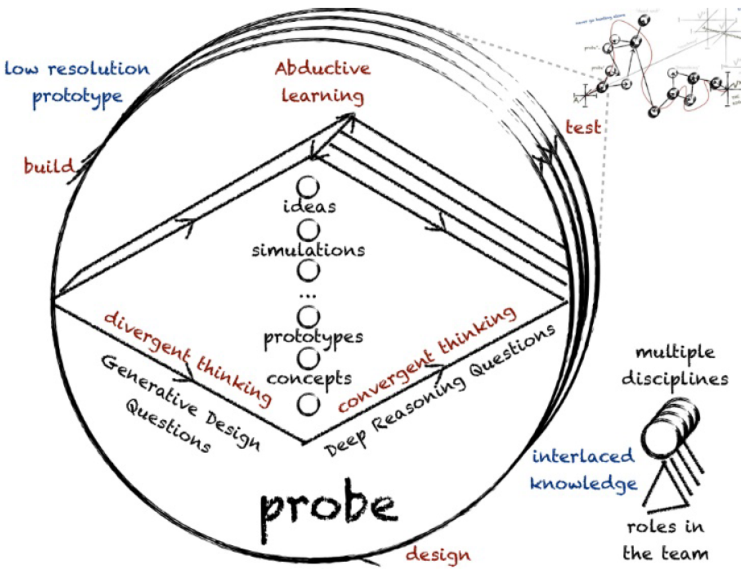


Figure 3.2: Probing cycle. Figure taken from publication 3.

conducted by Thov Reime as part of his master project at TrollLABS. He utilized a prototype-driven approach during the development. He took the Fuzzy Front-End course [Steinert 2020] that introduces the principles of wayfaring but during the master project he received little methodological guidance. The described development process is a qualitative post-project analysis. In this case several methodological characteristics were observable and these are described below. The technical solution after 6 weeks of development, and as shown in contribution 2, entails a tangible car model with one central part and four peripheral parts that can be assembled into a tangible car model. An Arduino-like micro controller board in the central part uses resistors in the peripheral parts to detect the current configuration and transfers this information to a PC where the configuration is digitally represented.

3.1.2 Eliciting unknown unknowns, probing and abductive reasoning

Unknown unknowns are things we are not aware of that we do not know them. In this context this means complications the developers are unaware of and do not have a solution for. Since they are by definition not predictable they make the outcome of a project uncertain. Eliciting these unknown unknowns early is therefore essential to decrease uncertainty and prototyping can help to discover them. The developers build prototypes that they think will work but then unexpectedly do not work for an unpredicted reason. After the problem is discovered it is a known unknown because now the developer is aware of it but still does not know how to solve it. Finding a solution requires abductive reasoning to formulate a hypothesis for the cause of the problem and experimentation to confirm or falsify it in order to be able to design a solution. Abductive reasoning is a creative process that tries to find the most likely explanation. This is necessary when it cannot be derived deductively or inductively. The unknown unknown was discovered by testing a prototype and became a known unknown. By abductively reasoning the cause of the problem is estimated and based on this a new improved prototype is built and tested. If the problem is solved the initially unknown unknown becomes a known known. If however, a solution cannot be found it remains a known unknown. A known unknown, or a solution (known known) that seems unreasonable, lead to a dead-end in the development process. Finding these dead-ends quickly is essential because quitting or fundamentally redesigning a project in later stages leads to higher cost and lost time. Therefore, early testing of prototypes to learn (instead of late prototypes to verify the design) are a way to map out opportunities, understand the pitfalls and benefits of different solutions before committing to a design and spending more resources on optimizing this design. These design opportunities or dead-ends and road blocks determine the development path and the design requirements evolve dynamically and therefore this approach is called wayfaring.

An example for an unknown unknown during the development of the Fibo car was that the resolution of the resistance measurement was too low for differentiating between a large number of different car parts, especially if they are connected in series. This was an unforeseen learning and ruled out this concept as infeasible for a commercialized version of the product. Being aware of this now known unknown means that the developers could abandon this concept and not spend and thus waste more time on it. Another example of an unexpected problem was the connector design. The initial design was carelessly designed and not universal, meaning that two equal connectors could not connect to each other (a male and female connector was needed) and this restricted the configurability of the car model. I would rather call this an unknown known because it is an easy to fix design flaw that as soon as one discovers it the solution is eminent. Nonetheless, it is still important to discover it early to save redesign resources later. This connector design itself was a result of testing a purely magnet based connector that proved to be not mechanically strong enough to hold the attached car piece. This example shows how test results influence the further development and design requirements. There are also situations where unknown knowns can become known knowns by studying existing information. This was the case during week 5 when the master student tried to build a wireless data transmission from the LightBlueBean micro controller board in the car model to the computer that displays the car model digitally. The transmission of serial data through bluetooth to a computer running Windows 8.1 was not possible. This information is available and knowing this could have prevented testing this combination. In this case however, I argue that building and testing a prototype and then finding the cause is less time consuming then finding this information prior to testing the prototype because the developer does not know what information to look for. In this case testing a prototype is the faster method to find unknown knowns.

3.1.3 Low resolution prototyping of critical functions

The wayfaring approach implies that failing is desired. Obviously not in the sense that the problem shall not be solved but in the sense that if the anticipated solutions cannot solve the problem we want to find out rather sooner than later. This has the implications that the functionalities that are critical for the solution (e.g.. a plane needs to at least fly) and the possible solutions that are most uncertain need to be tested first such that if these critical functions cannot be fulfilled the project can be abandoned as soon as possible. The other way to safe time and resources is to build low-resolution prototypes. The question before building a prototype shall be how to gain most insights for solving the critical function with the lowest usage of resources. This can for example be done by good choice of prototyping material, restricting the amount of features in the prototype, choice of prototyping tools (e.g. do I need digital design or is handcrafted good enough?) and approximation where details are not necessary. By exploring different concepts for solving the critical functions the developers can mitigate risk of problems that may stop or delay the development in later stages and can compare different solutions before committing to design requirements for further optimized designs. In the Fibo car project we can find several examples of low-resolution prototyping of critical functions. One critical function is to correctly detect the configuration of the car model. In order to find out if using resistors to identify the configuration is a viable solution it is not even necessary to design any car parts or even connectable parts. Testing the measurement electronics and the resistors on a bread board is faster, more adaptable and sufficient to find the flaws of this method. Another example is finding a convenient way to transfer data from the central micro controller to the computer that displays and later simulates the digital car model. For verifying that the data transmission works the master student first displayed a rocket and a chair from a library. Although the final solution needs to display car model parts and not a rocket and a chair testing the data transmission had higher priority because it is a critical function that is less likely to succeed. By prioritizing the critical function that is more likely to cause unforeseen problems more uncertainty can be decreased earlier.

3.1.4 Quasi-simultaneous prototyping of domains - finding interdependencies

Usually an early low-resolution prototype does not fulfill all critical functions from different domains within this one prototype and the critical functions are therefore prototyped and conceptually solved separately. However, making the different domains work together is also a critical function and hence should be tested early. This means it can make sense to combine different domains even if the individual solutions are not fully developed or known to not be used in a finalized product. This is to find interdependencies between domains.

3.2 Wayfaring in large-scale projects - CERN example

So far I have laid out how wayfaring can be used to deal with uncertainty, how to incorporate quasi simultaneous prototyping to discover interdependencies sooner and how different concepts can be tested and compared with low resolution prototypes. All this requires a good overview over the projects. In the projects at TrollLabs this was possible because the projects were developed by a small number of people who had the necessary expertise to solve all aspects of the problem. However, there are projects that even a small well-connected group of people cannot solve because of the high workload as well as the diverse expertise that is needed. A common approach to such projects is to divide the project into well-defined smaller subprojects called modules where a person or a small group take responsibility for such modules [Sanchez et al. 1996, Baldwin et al. 2000]. To ensure that the modules work together when merged the specifications for the interfaces between the modules are defined before the design starts. Modularizing at the beginning of the project predefines a solution that incorporates only these modules and therefore reduces the solution space and makes conceptual design decisions at a time when the uncertainty is highest. The idea of wayfaring is that the concept is not planned before the design starts but emerges through repeated building and testing of prototypes and the concept choices are made after unknown unknowns are found and the uncertainty has decreased. The question then is how wayfaring can be adapted to be applicable in complex large-scale projects. The problem becomes how to coordinate many developers with different expertise while still keeping the design flexible and agile enough to react to unforeseen problems. To give a general direction to everyone involved without giving precise instructions the project needs a vision. This is a goal that is agreed on by everyone involved and ensures a common ground [Srikanth et al. 2011]. The developers can self-select the way they contribute to this vision based on their expertise. Since no central top-down authority defines responsibilities the developers create a network where everybody has their envelope, that is a self-assigned area of expertise, and take responsibilities according to the needs and abilities in the network. These envelopes can overlap such that different developers can have affirmative or conflicting opinions on designed solutions. Decision cannot be made top-down since such a hierarchy does not exist. Instead decisions emerge from a justification process. That is a sense-making process where the developers need to explain and justify their design decisions to others in the network. In the first place, having to justify design decisions leads to a self-reflection and in a second step other developers try to point out flaws in the design. In complex projects with uncertainty, rational arguments are sometimes not possible as the knowledge does not exist yet. In these cases experimentation is needed to gain factual information. For this low-resolution prototypes can be used to answer specific questions and thereby influence the decision making. Since designs have to be justified to other developers in panel meetings also developers that are not working on the same subproblem but are effected need to evaluate the designs, this justification process creates knowledge across envelopes. This kind of knowledge Türtscher calls interlaced knowledge and it helps with foreseeing interdependencies as the developers learn about the designs of other developers [Türtscher 2008]. This is suboptimal from a knowledge processing point of view but it enables the necessary agile design responses without managerial intervention.

To study this I look at Philipp Türtscher's publication [Türtscher et al. 2014] and his PhD thesis [Türtscher 2008] where he describes how the design of the ATLAS detector at CERN emerged over time as a result of justification. He used interviews to study the emergence of the detector design as well as quantitative semantic analysis to study the network structure of the developers and the amount of justification indicating language throughout the development process. The following are insights and examples from his research about the development of the Atlas detector.

The vision for the development of the Large Hadron Collider, and therefore the ATLAS detector as part of it, is to confirm or disproof the existence of theoretically predicted particles including the Higgs Boson. At the time when this vision was formed the necessary technologies for reaching this common goal were unknown. Therefore, the development process had to allow for experimentation, discoveries and design changes during the several decades of development as new technologies are developed and unforeseen problems (unknown unknowns and unknown knowns) that require design changes arise. In addition, this project is so large and requires expertise from so many fields that it cannot be done by a small group of people. In total more than 3.000 scientists and engineers voluntarily joined the distributed development of only the ATLAS detector from 1992 until the first measurements in 2008 [Türtscher et al. 2014, Cetina 2009]. Consequently, decision cannot be taken by a centralized decision-making body like in a traditional hierarchy because no individual or group had all the necessary scientific and technical knowledge to make such decisions. Possible solutions and controversies are discussed in working groups that make recommendations to the collaboration board which is the main decision making body where every participating research institute has one vote. Decisions required a two-thirds majority vote, placing a strong emphasis on generating a consensus. Voting in the collaboration board is described as formalizing the consensus that had already emerged from the discussions. The consensus is reached after repeated round of justification and rebuttal that includes evidence from simulation studies and tests of individual component prototypes. It is reported that proponents of a particular technology conducted their own research to develop a deep understanding of competing technologies so as to identify their strengths and shortcomings. In some cases design choices are delayed because they could not be based on simulations only and prototypes were built. By semantically analyzing meeting minutes, technical reviews, design reports and communication in mailing lists (2419 timestamped documents) Türtscher identified a lower level of justification in the muon detector group compared to the calorimeter group. By comparing the network density (how many different nodes does a node connect to) of the collaborating developers Türtscher found that the network density of the muon detector group was lower and more isolated within the developers own expertise than in the calorimeter group. This is where the indication that justification leads to interlaced knowledge comes from. It is also backed by interview responses and that both justification and network density increased after a revolt in the muon detector group where the developers demanded more justification. This capability of understanding the design rational of other detector parts allowed for the redesign of the superconducting magnets that had consequences for the inner detector group This interlaced knowledge and global perspective was useful when agreeing that fewer and smaller coils are overall beneficial. This meant that a redesign of the superconducting magnets and all the effected systems surrounding it was necessary. The change from a 10 m inner diameter of 12 magnet coils to 8 smaller coils with 9.4 m diameter meant that the inner detector electronics could not fit inside anymore unless the calorimeter and muon detectors are also redesigned. This was done to give the inner detector electronics space between the calorimeter and the muon detectors to fit cables and cooling pipes. This lead to the unforeseen problem (unknown unknown) that the signal cables of the inner detector now ran close to the power supplies and picked up signal noise. This was eventually solved by shielding those cables but the shielding material introduced disturbance to the calorimeter measurements which was then accepted and accounted for. As we see, dealing with unknown unknowns and redesigning needs a negotiation process between developers and the ability to take a "global perspective". This is not likely to happen by dialogue between many experts in a modularized pre-planned project management and is too complex for managers to understand and coordinate. Another ATLAS example where justification and interlaced knowledge caused a redesign is the cooling system of the inner detector. Developers from other detectors pointed out the risk of water leakage from a binary ice cooling system and convinced the developers of the inner detector to change towards an evaporative cooling system. It was a factual evaluation through a justification process that convinced the inner detector developers to change their design originating from other developers that originally had no knowledge and need for knowing about the cooling system but anyhow gained this interlaced knowledge over time.

Chapter 4

Experiment exploring the effect of early prototyping in cyber-physical-systems development - the Robot Experiment

Building and testing prototypes early in the development process appeared to be essential and shaped the development process in the Fibo car project, the ATLAS detector as well as in many projects conducted at TrollLABS [Kriesi, Blindheim et al. 2016, Kriesi 2018, Leikanger et al. 2016, Jensen et al. 2017] throughout the past years. However, none of these cases was ever conducted simultaneously using a sequential spec-driven approach in comparison to the applied prototype-driven approach. Therefore, it is not possible to compare the two approaches under otherwise similar conditions. The following experimental setup is designed to do exactly that.

4.1 Objective of the experiment

In this experiment the aim is to compare the influence on the task-solving performance of a spec-driven approach versus a prototyping-driven approach in an open-ended task that reflects the characteristics of early-stage cyber-physical developments. The experiment is a randomized controlled study where half of the experiment participants are allowed and encouraged to build and test prototypes frequently while the other half of the participants cannot test their designs during the early stages of the development.

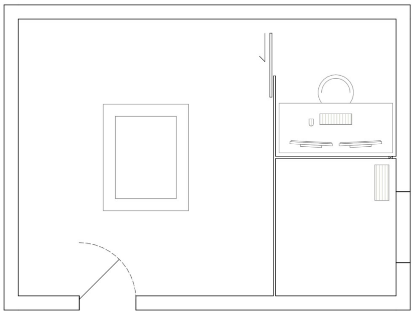
4.2 Experimental setup

4.2.1 Participants

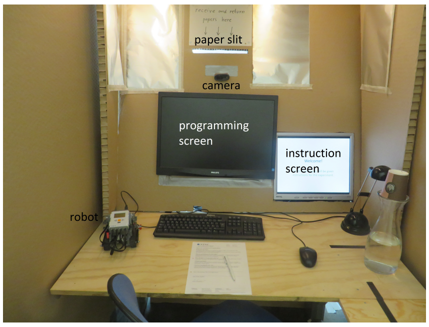
31 participants took part in the study and 27 are included in the quantitative results. They are recruited from the third year of the cybernetics and robotics program at NTNU. During this year the students organize fundraising for a field trip to Japan. The participants are paid 125 Norwegian Kroner for the participation in the experiment regardless if the measurements were completed or not. The money was paid to the general fundraising account for the whole class and the participants did not receive an individual payment. The participants were recruited through a contact person from their study program. This contact person was otherwise not part of the study. The participants sign-up in a form for the day and time they are available for participation. They are blind to which condition they are assigned to when signing up. The contact person sent out a standardized mail the day before the measurement explaining the location of the experiment and that the experiment will be conducted without direct personal interaction with another person and that the experiment will start automatically when the participant enters the experiment room. The experimental condition was varied in the time of day to avoid a bias. 13 participants (planners) were not allowed to test their designs within the first 80 minutes of the programming phase and 14 participants (testers) were given the stimulus to test their prototypes regularly and from the beginning of the programming phase. 11 participants in the planning condition are male and 2 are female while 11 male and 3 female participants were given the testing condition. The average age of the planners is 23,6 years with a standard deviation of 1,0 years and the average age of the testers is 23,7 years with a standard deviation of 1,1 years.

4.2.2 Physical environment

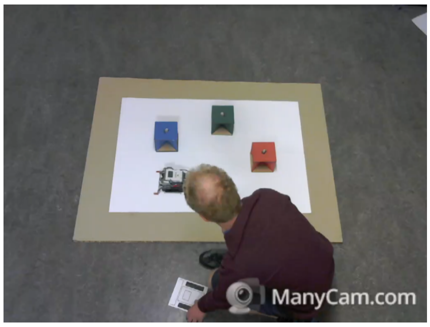
The experiment room is divided into three areas (see figure 4.1a). The participant has access to the testing area and the programming booth and the experimenter is in the control area. The three areas are separated by honeycomb reinforced and about 2 meter high cardboard walls that can block the view between the areas. The programming booth and the experiment area have sliding cardboard doors such that the testing area can be accessed either by the participant or the experimenter while maintaining visual separation. Once it is needed, the playground on that the robot drives is placed on the floor such that the distance from the edge of the play-



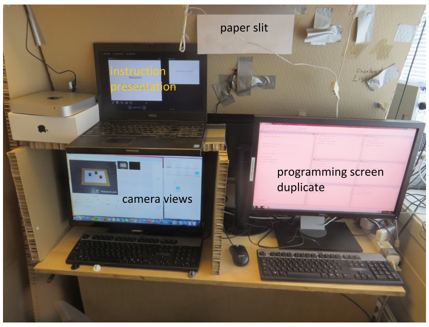
(a) Floor plan



(b) Programming booth



(c) Camera view on the playground



(d) Control room

ground to the nearest wall is similar in all directions. A camera under the ceiling captures videos of this area and provides a live video feed to the experimenter (figure 4.1c shows the camera view). In the programming booth the participant sits down in front of a table with two screens (see figure 4.1b). The larger screen in the center is the programming screen that the participants use to program their codes. To the right of this screen is the instruction screen where the experimenter can present predefined instructions through a Power Point presentation. On the table is the keyboard used for programming, the robot connected to the programming computer, drinking water, paper cup and a pen for taking notes and filling out the consent form and questionnaires. Above the programming screen is a camera for video capture and live video feed to the experimenter. Above the camera is a slit in the cardboard wall that connects the programming booth and the control area. It is used to exchange paper documents between the experimenter and the participants. It is covered with a lid to avoid visual contact between the participants and the experimenter. In the experiment control area the experimenter can see a duplicate of the programming screen, the live video feed from the two cameras and controls the power point presentation that is shown on the instruction screen (see figure 4.1d).

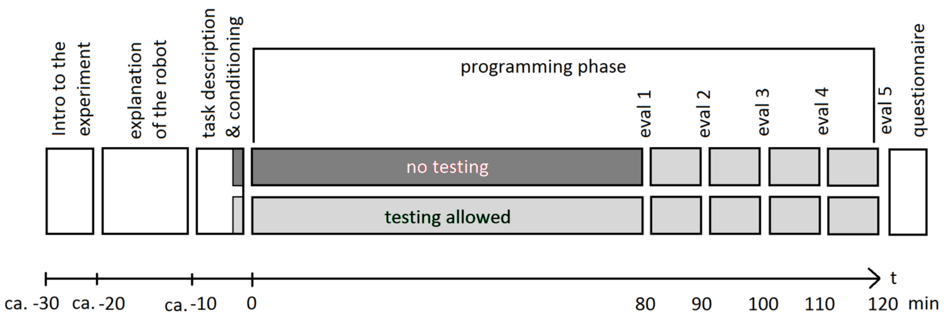


Figure 4.2: Timeline of the experiment execution.

4.2.3 Timeline of the experiment

The timeline of the experiment is shown in figure 4.2.

The **introduction to the experiment** starts when the participant enters the room with a pre-recorded male computer voice that greets the participants and reminds them that the information is given through these prerecorded voice or text instructions and that the participants shall sit down inside the programming booth. At the desk the participants find an empty consent form that informs about the purpose of the experiment, what data is recorded, that the data is recorded anonymously, that the participation is voluntary and can be stopped at any moment by the participant and that the participants shall keep the content of the experiment confidential to provide non-biased conditions for future participants. After consenting to participate in the experiment the screen and video recordings are started.

The participants are given a paper questionnaire asking for their programming experience in various relevant languages (see appendix A.1). These are C and C-like language because the NXC (not exactly C) language is fairly similar and used during the experiment, Arduino and Lego Mindstorms because these types of projects are usually cyber-physical and therefore comparable to the task in the experiment and Python because it is the language most taught to the cybernetics students that participated in the study. They are also asked to rank all the programming languages they have used. This is done to ensure an answer on the languages mentioned above but also providing an opportunity to add additional languages. The questionnaire, as well as all other papers exchanged between the experimenter and the participants are fed through a covered slit in the wall to avoid visual contact between the experimenter and the participant. To estimate the actual programming skills of the participants and introduce them to the syntax used later without showing them the actual robot or task they are given a code example that includes variables, a function, a loop, conditional and logic expressions. The participants shall go through the code and track the value of a variable until the end of the code. The result and the time needed is recorded. The maximum time available is three minutes. The code example is presented to the participant on paper and can be found in appendix A.1.

The **explanation of the robot** is given on the instruction screen accompanied with prerecorded voice instructions. The images shown on the instruction screen and the voice instructions are coordinated in a Power Point presentation. This ensures an equal presentation for every participant. The instruction includes how the robot can move, what sensors it has and how a code can be loaded onto the robot. The actual physical robot is present on the desk in the programming booth throughout the entire experiment. After the introduction on the instruction screen the participants are handed out a data sheet about the robot (see appendix A.1) and a paper explaining the functions available in the software library written for this robot (see appendix A.1). The data sheet includes the calibration data about the robot movement and the sensor measurements. The participants have a blank paper for taking notes and are given 15 minutes to study this material. A countdown timer on the instruction screen shows the remaining time. To ensure the participant knows how to load and execute a code on the robot an example code that plays sounds is shown on the programming screen. At the same time an instruction text combined with a voice instruction explains how a code is compiled and loaded onto the robot and then executed. After successful execution of the code a timeline of the experiment is shown to give the participant an indication what to expect. This timeline with an arrow to the corresponding point in time is shown at different places throughout the experiment to guide the participant. Then the programming screen is turned off.

A paper version of the **task description** (see appendix A.1) is handed out and immediately followed by a predefined visual and voice instruction of the task on the instruction screen. Meanwhile, the experimenter arranges the "playground" where the robot needs to drive to fulfill the task. Before this the testing area remains empty to not give the participants any impression about the possible task. After the presentation of the task has ended the participant can go to the playground and inspect it before the participant is told which experimental condition to follow. Then the **programming phase** starts. The stimulus is introduced during the first 80 minutes of the programming phase. The planners are informed that they cannot load the code onto the robot within the first 80 minutes of the programming phase but they can compile their codes to find syntax errors. The testers are instructed that they shall test often and at least every 5 minutes. A voice instruction will remind them if they do not test within five minutes. They are informed that there are six smaller programming windows on the programming screen to test code snippets separately. The planners also have these smaller windows available. During the programming phase a timer shows the remaining time until the next evaluation of the solution. A voice reminder of the remaining time is given at 60, 30, 15, 5 and 3 minutes before the initial evaluation. At three minutes a reminder is given to make sure the code compiles when the evaluation starts. In the planning condition the keyboard key for loading the compiled code onto the robot is deactivated until the first evaluation. Just before and after the first evaluation a text on the instruction screen and the computer voice asks the participants how satisfied they are with the current solution on a 0 to 6 Likert scale. They answer by saying a number. This is immediately followed by the first evaluation. All five evaluations follow the same process. The programming screen is turned off such that the participant cannot continue to program and the cardboard wall separating the programming booth with the playground is closed. Then the experimenter can arrange the setup of the cubes and the robot starting position for the evaluation. The robot's starting position is indicated by a paper sketch that the participant later replaces with the actual robot. The starting positions of the cubes are similar in every evaluation but the robot's starting position changes for each evaluation. The participants do not know this starting position before the evaluation. The same sequence of robot starting positions is used for every participant. The code is saved and loaded onto the robot. When the experimenter has arranged the setup a combined instruction from the instruction screen and a voice instruction asks the participants to come to the playground, replace the robot sketch with the actual robot and start the execution of the code as indicated by the voice instruction. The experimenter can observe the behavior and performance of the robot from the camera above the playground and documents the performance manually. The evaluation ends when either the robot completes the task, falls off the cardboard surface and therefore cannot complete the task or the participant determines that the robot will not complete the task for example because the robot does not move or is stuck. A voice instruction then asks the participants to return to the programming booth and continue programming until the next evaluation. The remaining time until the next evaluation is shown together with the timeline of the experiment on the instruction screen. The programming time before the 2nd, 3rd, 4th and 5th evaluation is ten minutes before each evaluation and each time a three minute remaining time warning is given. After the first evaluation all participants can load their codes onto the robot and test them during the programming phase. The voice instructions reminding the testing participants to test are not used after the first evaluation anymore. In total there are five evaluations and after the last evaluation the participants fill out a questionnaire where they are asked about biographical information, how much they enjoyed the experiment and how satisfied they are with their solution. The experiment ends with an unstructured interview where the experimenter can ask more specific questions about the solutions and explain the purpose of the experiment in more detail. This is a face-to-face conversation between the participant and the experimenter.

4.2.4 The task

The task is to autonomously use the robot to remove three cube-shaped objects from a rectangular white area in the shortest time possible from starting the robot until the three cubes are removed. The white rectangle is approximately 1,50 meter by 1,20 meter and is surrounded by a darker 17cm wide cardboard colored fringe. This cardboard is about three centimeters high above the floor. This means that if the robot falls off this cardboard it cannot drive back onto it and therefore cannot remove any more cubes from the top of the cardboard. Sketch 4.3 shows the cardboard, the white area and the starting position of the cubes. The starting positions of the cubes remain the same at every evaluation whereas those of the robot were randomly generated. Positions where the robot initially directly faces a cube are excluded. The series of robot starting positions is the same for each participant. The cubes must be taken off from the cardboard by the participant as soon as no part of the cube is touching the white area anymore. A ten seconds time bonus is subtracted from the overall time for completing the task if the robot indicates the correct corresponding color of the cube while removing it. A ten second penalty is given if the wrong color is indicated. The color is indicated by playing a tone with the frequency that corresponds to the cube color (red cube = 400 Hz, green cube = 800 Hz, blue cube = 1600 Hz). The participants are provided with three blinking lights that they can optionally place wherever they want and they can be used as beacons that the robot can detect. They fit upside down into the ceiling of the cubes and their light then radiates outwards such that the light is detectable from any direction around the cube. The task is presented to the participants through instruction slides on the instruction screen accompanied with corresponding voice instructions and through a paper handout that the participants can keep until the end of the experiment. The paper handout can be found in appendix A.1.

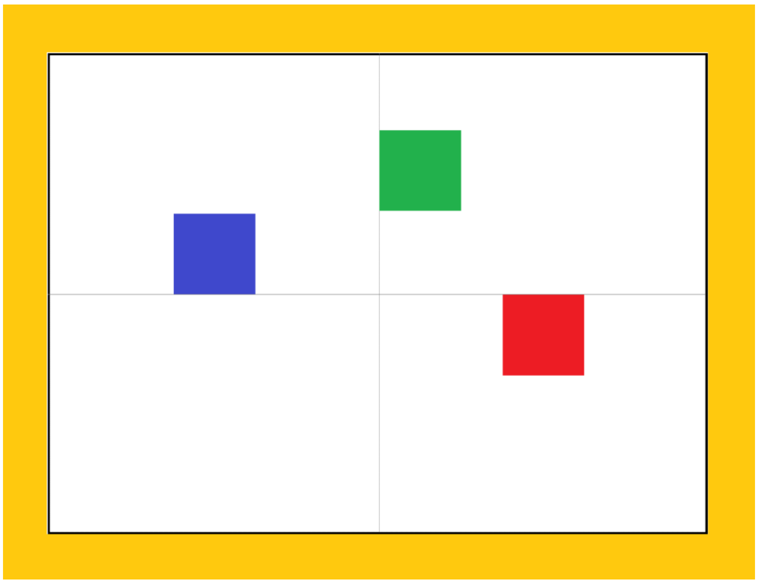


Figure 4.3: Top-down view drawing of the playground with the starting positions of the cubes

4.2.5 The robot

The robot is self-designed and based on LEGO Mindstorms version NXT 2.0. It has two electric motors that drive a belt located on the left and right side of the robot. By using the motors synchronously the robot can move straight forward and backwards and turn by asynchronously using the motors. Each motor speed can be controlled individually. The robot has four sensors.

Figure 4.4 shows the robot.

- **ultrasound sensor:** emits an ultrasound signal and measures the time of flight until the reflected signal returns to the sensor. From this time the NXT controller calculates the distance between the sensor

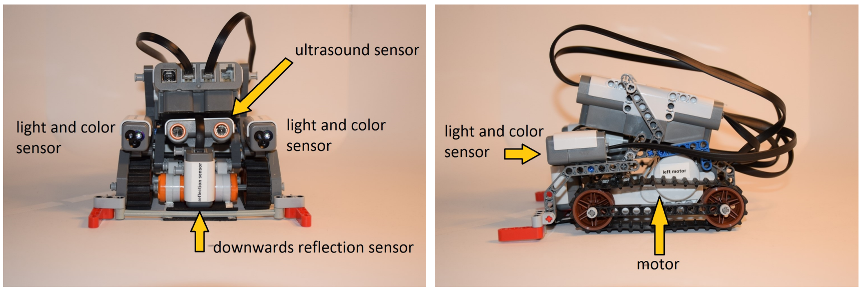


Figure 4.4: Robot seen from the front and the side

and the reflecting object in front of the robot with centimeter precision.

- **downwards reflection sensor:** emits red light and measures how much of this emitted light is reflected back into the sensor. The reflected light intensity is distance and reflectivity dependent. This sensor is pointed downwards towards the ground and measures the ground's reflectivity as the distance remains constant. A darker surface under the robot reflects less light and leads to a lower signal than a brighter surface. With this sensor the robot can differentiate between the white surface, the cardboard surface and the darker floor.
- **two light and color sensors:** There is one such sensor pointing forward on each side of the robot. This sensor can emit red, blue and green light and can measure how much light is reflected back into the sensor. Thereby it can measure how much light falls into the sensor or more explicitly how much of each color is reflected. These sensors can be used to detect the blink light and the color of reflecting surfaces in front of the robot.

The calibration measurements for the robot can be found in the data sheet in appendix A.1 and in contribution 7.

The robot is programmed in the NXC language which stands for "not exactly C" and is therefore a language with a syntax that is very similar to C but adapted for the NXT Lego Mindstorms controller. Additionally, the participants are provided with a library written for this robot. It provides simplified functions that make driving the robot and interpreting the sensor readings easier. Apart from functions for driving the robot and using the sensors this library includes functions for pausing the code execution, timers, playing sounds and showing text and numbers on the NXT's display. A detailed description of these functions can be found in the description that the participants receive and that is shown in appendix A.1 and in contribution 5.

4.2.6 Participant-experimenter interaction

In a controlled study the conditions shall be as similar as possible for every participant. As personal interactions are difficult to control and can have a subconscious effect on the participants behavior [Kahnemann 2011] one aim is to avoid direct personal interaction between the experimenter and the participant and keep the instructions as neutral and as similar as possible for every participant. This primarily includes visual and auditory interactions but also the timing of all interactions.

Instructions to the participants are given by:

1. **computer generated voice:** Pre-defined instruction texts are fed into a text to speech algorithm and the resulting audio files are played back to the participant at the appropriate time. A male voice was used. The timing of the audio playback is either defined through the Power Point presentation that is displayed on the instruction screen, after predefined cues or contextual. The power point presentation ensures that the audio is played synchronously to the displayed information. An example for a cue is starting the Power Point description of the task shortly after the participant received the paper instruction but before the participant starts reading it. Contextual voice instructions are for example if the participant does not close the door and is informed to do so, giving syntax help when the code does not compile or answering questions with "yes" and "no". The experimenter only answers procedural questions and does not answer questions about the solution to the task. The voice instruction is also used if the participant incorrectly places the robot during an evaluation, as a countdown to start the robot during the evaluations and instructing the participant to connect the robot to the USB cable to the programming computer so that the code for the evaluation can be loaded onto the robot. In some rare cases the prerecorded voice messages are not sufficient and the experimenter had to talk to the participants directly with the human voice. Examples for this are if the robot stopped working (happened once) or in the pre-study during a fire alarm and during a nosebleed of the participant. All these cases were either noted in the lab book and later evaluated if and how this influenced the experiment or excluded from the results. The voice instructions can be found in appendix A.1.

2. **Instruction screen and loudspeaker:** This screen is deliberately used to present instructions. The content shown comes from a Power Point presentation that includes text, sketches and audio voice instructions. Either the experimenter advances the slides manually following a predefined routine or whenever possible the slides advance automatically after programmed time intervals.

3. **Programming screen:** The programming screen shows seven programming windows. One large one on the left side and six small ones on the right half. The main code that is tested is in the large window. The six small programming windows can be used to separately

write, compile and, when testing is allowed, test code parts. The experimenter can turn the programming screen on and off to allow and disallow programming and to steer the participant's attention to the instruction screen. The programming screen is only turned on when the participants are supposed to program.

When the participant tries to compile the code and an error message shows up the experimenter starts a voice recording asking if the participant wishes to receive syntax help. If so the experimenter encircles the syntax error with the mouse cursor on the programming screen. This is accompanied with a voice instruction that says what type of syntax error exists there. Options are missing bracket, missing semicolon, spelling mistakes or simply "watch the mouse cursor for a hint" if none other applies. This help from the experimenter requires constant attention as the experimenter needs to already detect the syntax errors before the participant compiles the code and gets the error message in order to respond promptly. The experimenter only helps to find syntax mistakes and not logical programming mistakes.

4. **Paper documents:** These include the consent form, questionnaires, the programming test, information about the robot and the library and the task description. These papers are exchanged through the covered slit in the cardboard wall above the programming screen.

4.3 Raw-data collection

The main interest of the experiment is to investigate the influence of early testing of prototypes on the performance towards solving the task. The task involves removing three cubes in the shortest time possible. Therefore, there are two reasonable performance measures. Firstly, the number of removed cubes until the robot fails the task and secondly if the task is completed the time the robot took to complete it. This data is observed through the camera above the playground, the time is manually stopped and the experimenter notes if the robot moved at all, the times when the robot touched the first cube, the times when the cubes are removed from the white area and if the robot correctly identified the color of the cube that is pushed out.

Other quantitative measurements are:

1. **Keystroke latency logger:** An Arduino Leonardo documents which key and the time with millisecond precision when the participant presses down the key on the keyboard. The Arduino also has the

capability to execute routines when specific keys are pressed or a certain time expires. The Arduino keylogger uses shortcuts of the programming environment software to automatically safe the code. This process is triggered when the testing participants load the code onto the robot, participants in the planning condition compile the code or every 5 minutes without loading the code (during a 2 second typing break). Whenever a participant who is not allowed to load code onto the robot presses F6, which is the shortcut for loading code onto the robot, this keystroke is recorded but not sent to the programming PC and thereby inhibits that participants in the planning condition can test their codes on the robot before the first evaluation. The Arduino keylogger also uses the shortcuts when the experimenter presses a button and then safes the code and loads it onto the robot for the next evaluation. The keylogger keeps track of the iterations and names the code files accordingly.

The time stamps from the keylogger are synchronized to other time stamps during the experiment by entering the unix time stamp at the moment the keylogger is started.

2. **Number of tests:** Since the experiments compares the influence of testing onto the performance it is important to record when and how often participants test their codes with the robot. The number of saved and loaded codes as recorded with the keylogger is used as the measure for the number of tests. If the participant loads the same code onto the robot twice it is counted as one test and when the participant loaded the code onto the robot but then did not execute the loaded code to test it this was noted in the lab book.

3. **Questionnaires:** The questionnaires use a Likert-scale from 0 to 6. Three questionnaires are used during the experiment. The **programming experience questionnaire** asks for experience in coding C/C++ and Arduino as well as experience in using Lego Mindstorms. Further does the questionnaire offer the option to add further programming languages the participant knows.

   The **Post experiment questionnaire** asks first for gender, year of birth, study program and semester as well as for how pleased the participant is with his or her programming and how pleasurable the experiment was for them.

4. **Daytime of the experiment:** Both the scheduled start time for the experiment as well as the actual start time are recorded. This is done

to control if the time of the day has an influence on the results.

5. **Programming test:** Just after the programming experience questionnaire the participants receive an example code with the syntax used during the experiment. It includes variables, a function, loops and conditional statements and is designed to illustrate the syntax but also test if the participants can follow the basic programming needed for the experiment. The code to analyze is:

```
int x = 0;
bool y = false;

int function(int parameter1, int parameter2)
{
    return parameter1 + parameter2;
}

while(x <= 5)
{
    if(x >= 2 && y)
    {
        y = false;
        x = function(x,1);
    }
    else
    {
        x = x + 2;
        y = true;
    }
}
// END OF THE PROGRAM
```

   The correct value of x after the program completes is 6. The time provided for this test is three minutes but the participants can hand in their solution earlier. Both the value of x and the time until the solution is handed in is recorded. The participants are free to abort the experiment if they do not feel comfortable solving this test. Regardless of the result they were allowed to proceed with the experiment.

6. **Starting time of coding**: This is the time the participant uses between receiving the task and starting to code a solution

In order to gain deeper insights and being able to double check and explain outliers I, in addition to the quantitative results, also gather qualitative information.

The qualitative measurements are:

1. **Camera video recordings:** After the participant has returned the signed consent form the camera recordings are turned on. There is one camera above the programming screen facing towards the participant and one camera mounted under the ceiling with the field of view on the floor where the playground is. Both cameras are essential for the experimenter to conduct the experiment and react properly, but they also give the opportunity to observe and reconstruct the behaviour of the participant. For example can the experimenter evaluate facial expressions, when is the participant looking at the instruction screen, the programming screen or at a paper on the desk or how long and what did the participant test. The ceiling camera is also used to monitor and document the robot's performance during evaluations.

2. **Codes:** Saving the code solutions at different time intervals and when the participant tests allows the experimenter to follow the programming process. With the camera recordings from above the playground it is possible to combine and relate the observed behavior of the tested robot with the participant's code changes resulting from observing the test. The experimenter can compare the solutions at different points of time to follow the participant's programming logic and style.

3. **Screen recordings:** The programming screen of the participant is duplicated such that the experimenter has a real-time feedback of what the participant is programming and sees in which programming window the participant is typing. The programming screen is also recorded. It is also much more human readable and intuitive to qualitatively assess the participants programming behavior from a replay of this screen recording than from the time stamps of the keylogger.

4. **Paper notes from the participants:** Together with the information about the robot the participants are given a blank white A4 paper and a pen that they can keep during the experiment to take notes. These papers are mostly used to sketch code ideas.

5. **Lab book:** Besides using the lab book for notes mentioned above like time of the day, time stamps of screen and video recordings the

experimenter notes unexpected occurrences during the experiment. Examples are extensive yawning of a participant, when participants make comments, description of robot behavior and qualitative evaluations of the participants behavior or when things go wrong that can have an influence on the outcome.

4.4 Processing of keystroke data

Thomas et al. [Thomas et al. 2005] suggest that typing latency while programming is indicative for programming performance. They categorized keystrokes into alphabetic characters, numerical characters, control commands, arrow keys, and "other" keystrokes that include brackets, space, carriage return, slash, comma and semicolon. They then observed the latencies between keystrokes of different categories and found that the median latency of transitions from an alphabetical, numerical or control keystroke to "other" keystrokes is inversely correlated to a point score given for correctly completing a programming tasks.

I make the assumption that the relation between programming performance and typing latency does not change throughout the duration of the experiment and therefore use all typing data from the beginning of the programming phase to the end of the experiment. Because we are interested in the latency while the participant is continuously typing it is important to determine the median and not the average latency. Thus long typing pauses are of no consequence.

4.5 Results of the robot experiment

Out of the 31 participants 4 participants – 2 testers and 2 planners – are excluded from the evaluation: 1 participant quit the experiment 8 min after start of the programming phase, 3 other due to substantial technical failure or a bug in the programming library.

Other participants that encountered minor disturbances that were evaluated as having little to no influence on the experiment outcome are included in the study. Examples for these minor disturbances are technical problems with the keyboard that could be resolved almost instantly, one of the three blinking lights stopped working during the final evaluation and was replaced and an evaluation was repeated because the firmware of the robot crashed during the first attempt.

The two main performance metrics for a solution are how many cubes are removed during an evaluation and how long it took to complete the task. Since task completion times can only be evaluated for those solutions which did complete the task this measure is limited, especially for the solutions

Evaluation	N_{plan}	N_{test}	Mdn_{plan}	Mdn_{test}	U	p
1	13	14	0	1	81	0.319
2	13	14	0	2	63	**0.077**
3	13	14	3	3	70	0.135
4	13	14	3	3	88	0.456
5	13	14	3	3	90	0.500

Table 4.1: Significance analysis comparing the number of removed cubes for planners and testers, N_{plan} and N_{test}: number of datapoints, Mdn: median of removed cubes, U-rank statistic and corresponding p-value

from the early evaluations. Therefore, the number of removed cubes is the main performance indicator. Histograms of how many solutions removed no cube, one, two or three cubes in an evaluation are shown in figures 4.5 and 4.6 for planners and testers for all five evaluations.

The number of solutions that remove no cubes decreases over time and the number of solutions that complete the task increases over time as a general trend. However, this trend does not continue from evaluation 4 to evaluation 5 in both conditions. While in the forth evaluation all solutions removed at least one cube, in evaluation 5 solutions that do not remove a single cube reappear. The distribution of the number of removed cubes is bimodal in the first 3 evaluations of the planners and slightly in the first evaluation of the testers. Therefore, the comparison of the planners and testers is given by the medians and the significance is tested with a Mann-Whitney U-test for ordinal data (see table 4.1).

The only weak significant difference between planners ($N_{plan} = 13$ with median $Mdn_{plan} = 0$) and testers ($N_{test} = 14$ with median $Mdn_{test} = 2$) is found in evaluation 2 with a U statistic of 63 which results in a p-value of less than 0.1 (0.077). While in this evaluation the majority of the testers had developed a solution that at least removes one cube and often completes the task, a large portion of the planners had solutions that fail the task before they remove the first cube. After evaluation 3 the planners drastically improve their solutions. In evaluation 3 there is still a trend that the solutions of the testers remove more cubes with a p-value of 0.135 while for evaluation 4 and 5 the p-values increase a lot indicating that the number of removed cubes for the solutions of planners and testers are statistically indistinguishable.

For more detailed quantitative comparison of the planners and the testers, the number of solution that removed 1, 2 or 3 cubes respectively, the mean time for removal with standard deviation as well as the results of

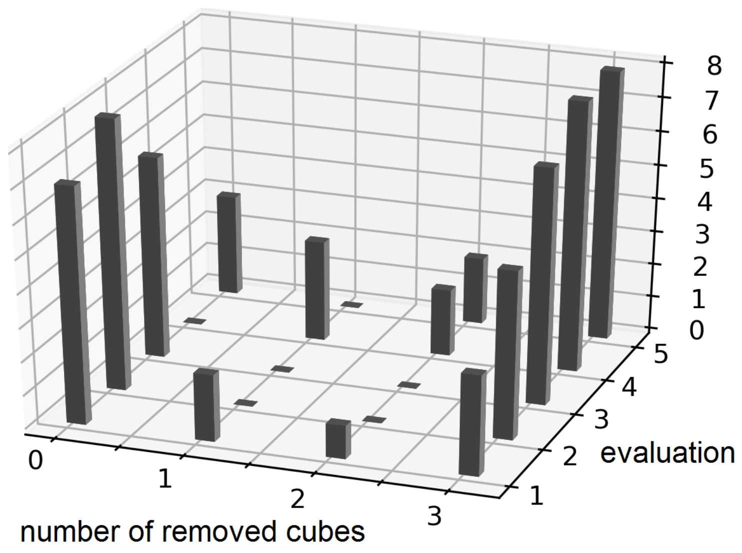


Figure 4.5: Planning condition: Frequency of number of removed cubes in each evaluation shown along the vertical axis

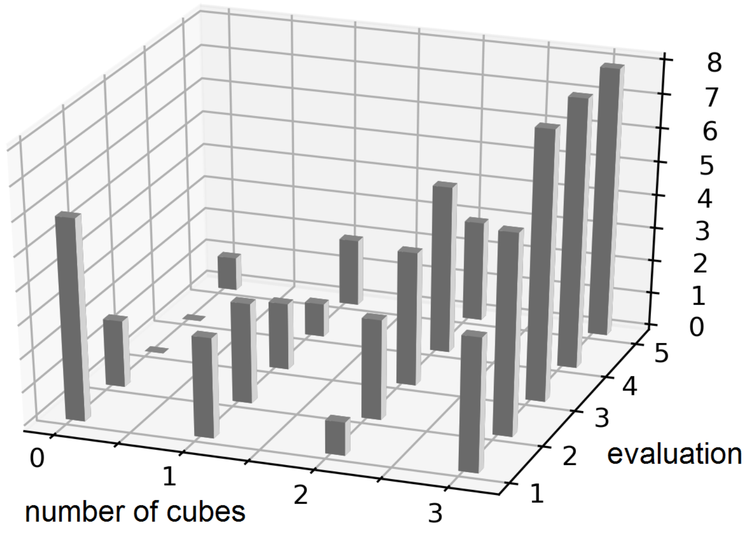


Figure 4.6: Testing condition: Frequency of number of removed cubes in each evaluation shown along the vertical axis

the significance test are given in table 4.2 for each of the 5 evaluations and for the individually best evaluation of each participant. The best solutions is selected firstly by the number of removed cubes and if the values were the same by the shortest time for removal. For the statistical comparison, a t-test that allows for unequal number and standard deviations (Welch's t-test) is used. As the contribution of the colour detection on the performance is only of minor importance it is not included in the evaluation here but only considered in section 4.7. Graphs showing the removal times for the first, second and third cube are presented in appendix B.1.

In all evaluations expect one the testers more often remove the first, second and third cube. Although this may indicate that the testers outperform the planners in the number of removed cubes this result is generally not statistically significant as the difference is small and the standard deviation high. The only weakly significant difference ($p = 0.096$) occurs for the task completion time (3rd cube time) when comparing the individually best performance from every participant. In this case the task completion time of the testers outperforms the task completion time of the planners.

It is striking that all solutions from the planners in evaluations 2 and 3 that removed the first cube then also proceeded to complete the task (see figure 4.5), also seen from the median number of removed cubes which is either zero or all three (Table 4.1). This shows that there is a separation of solutions which are either successful or unsuccessful and very few solutions that partly solve the task and remove one or two cubes.

4.6 Limitations of the performance measurement

The results presented above originate from executing the solutions during the scheduled evaluations. This means that we get one set of data points per solution namely at the 5 different evaluations. At each evaluation the solution is executed once from a different starting position. The task is to find a solution that removes the cubes from any starting position and therefore a test of the solution from only one specific starting position is not very representative for the performance of that solution. Furthermore, as described in subsection 2.3.2 the behavior of the robot is non-deterministic, meaning that executing the solution under seemingly same conditions again does not give the same result. The one existing data point per solution may therefore fluctuate substantially from the average performance of this solution. When comparing the task completion times of solutions of different participants from one experimental condition the standard deviation often is substantial compared to the mean value and therefore makes it difficult to observe a significant difference between experimental groups. From the

		Eval	1^{st} cube			2^{nd} cube			3^{rd} cube		
			N	M	SD	N	M	SD	N	M	SD
Eval 1	Planner	1	6	36	40	4	81	73	3	116	101
Eval 1	Tester	1	8	17	7	5	37	21	4	66	28
Welch's t-value		1	$t = 1.17$			$t = 1.17$			$t = 0.83$		
p-value		1	$p = 0.29$			$p = 0.32$			$p = 0.49$		
Eval 2	Planner	2	5	14	11	5	52	31	5	65	32
Eval 2	Tester	2	12	19	14	9	32	12	6	58	24
Welch's t-value		2	$t = -0.76$			$t = 1.33$			$t = 0.41$		
p-value		2	$p = 0.47$			$p = 0.24$			$p = 0.69$		
Eval 3	Planner	3	7	21	26	7	49	48	7	97	71
Eval 3	Tester	3	14	24	32	12	41	29	8	72	35
Welch's t-value		3	$t = -0.24$			$t = 0.40$			$t = 0.84$		
p-value		3	$p = 0.81$			$p = 0.70$			$p = 0.42$		
Eval 4	Planner	4	13	22	26	10	88	77	8	145	119
Eval 4	Tester	4	14	23	19	13	57	56	8	116	143
Welch's t-value		4	$t = -0.09$			$t = 1.04$			$t = 0.44$		
p-value		4	$p = 0.93$			$p = 0.31$			$p = 0.66$		
Eval 5	Planner	5	10	19	11	10	41	26	8	82	47
Eval 5	Tester	5	12	18	16	11	46	25	9	77	65
Welch's t-value		5	$t = 0.09$			$t = -0.46$			$t = 0.17$		
p-value		5	$p = 0.93$			$p = 0.65$			$p = 0.87$		
Best	Planner	4.3	13	14	9	13	44	51	11	86	71
Best	Tester	3.9	14	13	7	14	30	10	13	46	18
Welch's t-value			$t = 0.321$			$t = 0.972$			$t =$ **1.82**		
p-value			$p = 0.75$			$p = 0.35$			$p =$ **0.096**		

Table 4.2: Removal times of the 1st, 2nd and 3rd cube and significance analysis comparing planners and testers for all 5 and the individually best evaluation of each participant. N: number of solutions that removed the cube(s), M: mean times in seconds with standard deviation SD, Welch's t-values and p-values calculated from the corresponding data above

obtained data it is impossible to say if the large standard deviation is caused by non-deterministic behavior and repeatedly executing the same solution would yield a similar variance or if the task completion time actually varies substantially between different participants.

It is therefore necessary to repeatedly execute each solution from the same starting position to find the average performance and standard deviation from this starting position and then repeat these measurements from a larger number of starting positions to get a fair and comparable evaluation of how this solution generally solves the task. This can in theory be done manually by post-experimentally executing the code repeatedly. However, assuming that setting up the environment and executing the code takes about three minutes per iteration, with 10 iterations per starting position to gain sufficient data for a decent statistical analysis and 20 starting positions for evaluating how universally the solution solves the task then a proper evaluation of just one solution takes 10 hours. In addition, for a comparison of the solutions under similar conditions it is furthermore necessary to avoid, repair or compensate wear and tear of the robot and the environment. These efforts, even if technically possible, are very infeasible.

A digital simulation that simulates the path of the robot and its interactions with the environment can overcome the limitations. The simulation is deterministic and therefore needs only one iteration per starting position. It is based on the codes provided by the participants during the experiment. It is verified qualitatively by comparing the real world and simulated robot behavior as well as statistically with three exemplary solutions. The behaviors of the physical robot and the simulated robot are qualitatively indistinguishable. The simulation generates faster task completion times than the manually measured task completion time of the physical robot. However, the simulation gives the correct relative task completion time when comparing solutions. Since this experiment compares solutions of participants from different experimental conditions a relative comparison is sufficient and the simulation can be used to evaluate and compare the solutions. More details on the purpose, development and verification of the simulation can be found in contribution 7 and in section 6.3. The simulation can provide data from sufficiently many starting positions within minutes instead of hours for one solution, i.e. the evaluation of all solutions of the study data can be done within one day.

4.7 Simulation enhanced results of the robot experiment

The simulation results are gathered from simulating the trajectory from 99 different starting positions for each solution. The coordinates and the robot orientation of these starting positions are generated randomly and then reused for evaluating every solution. Each starting position is executed once per solution. More iterations are not necessary because the simulation is deterministic.

The performance result of a solution is the mean performance over the simulation results of the 99 different starting positions. For comparing groups of solutions, like planners versus testers, I calculate the median (Mdn) or mean (M) result within each group from the mean (m) results of the 99 runs for each solution. Explanations and detailed data are mentioned in appendix B.2.

In the following results, the same participants as in the previously shown data are included (and excluded).

4.8 Influence of testing on performance

In analogy to the previously presented manually recorded results I compare the performance results of the group of participants who were not allowed to test their solutions during the first 80 minutes of the programming phase (planners) with the group of participants that were encouraged to test their solutions regularly from the beginning onward (testers).

For the evaluation the number of removed cubes as well as the time for removal of the cubes, especially of the 3rd cube, which is the completion time, are taken into account. The completion time is only available for those participants who succeeded in solving the task. In the 1st and 2nd evaluation the data for the completion time is based on 6 to 9 participants, in the last evaluation on 12 out of 13 for the planners and 13 out of 14 for the testers. Figures 4.7 and 4.8 each show a histogram over the mean number of removed cubes separated by planners and testers for each of the five evaluations. Unlike in the previous manual evaluation with 1 run per solution, now the mean number of removed cubes over the 99 simulation per solution can lie in between the natural numbers. The results are summed up in 4 bars with a bin size of 0.75 cubes each.

In the simulated data there is also a clear development towards an increasing number of removed cubes from one evaluation to the next with the previously seen reversal from evaluation 4 to 5. The results again are bimodal as the simulated solutions often either perform poorly or solve the

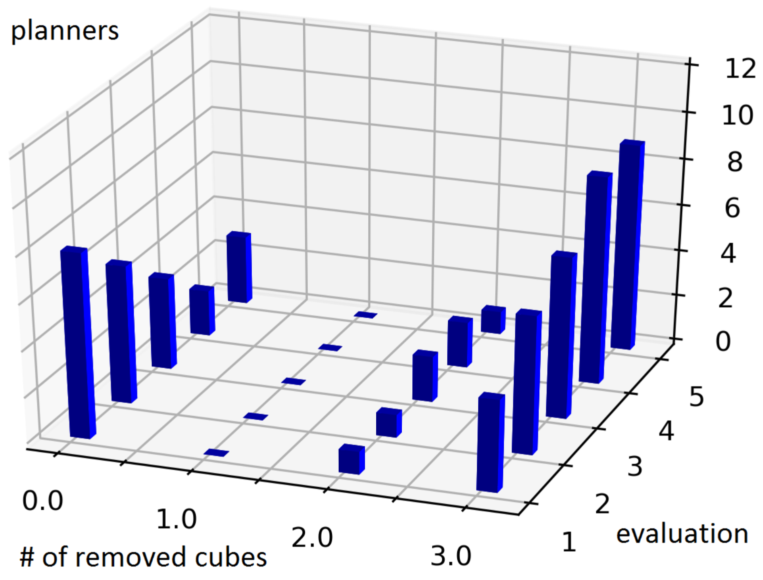


Figure 4.7: Planning condition: Frequency of mean number of removed cubes in each evaluation, simulated results

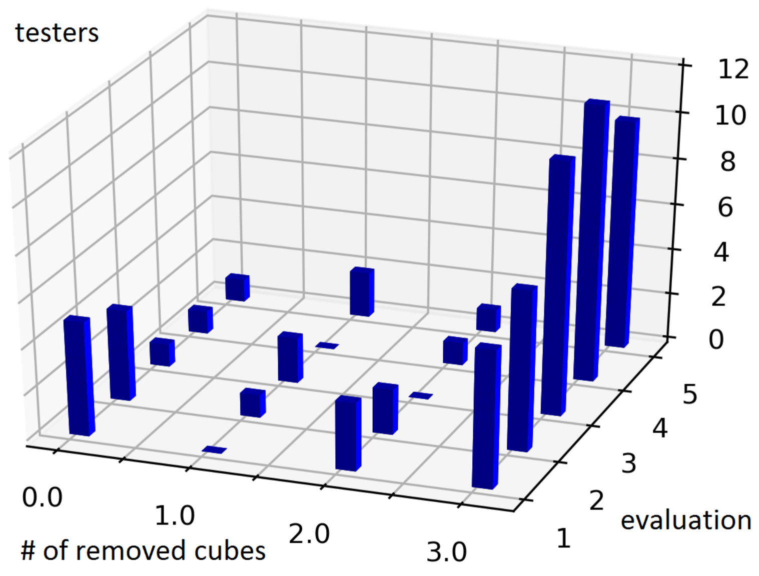


Figure 4.8: Testing condition: Frequency of mean number of removed cubes in each evaluation, simulated results

Evaluation	N_{plan}	N_{test}	$Mdn(m)_{plan}$	$Mdn(m)_{test}$	U	p
1	13	14	0.21	1.95	87	0.432
2	13	14	1.59	2.13	84	0.376
3	13	14	2.78	2.72	87	0.433
4	13	14	2.65	2.67	84	0.376
5	13	14	2.75	2.62	70	0.166

Table 4.3: Significance analysis comparing the number of removed cubes for planners and testers. Mann-Whitney U test, N plan and N test: number of datapoints, Mdn(m): medians of the mean of removed cubes, U-rank statistic U and corresponding p-value p, simulated results

task but seldomly remove one or two cubes. This bimodal behavior seems to be more pronounced for planners than testers.

Due to the fact that the data for the number of removed cubes do not show a normal distribution, the statistical comparison of the planners and testers is based on the medians and tested with Whitney-Mann U test. Table 4.3 shows the simulated results for the median of the number removed cubes (from the mean over the 99 starting positions for each solution) for planners and testers in all evaluations. No significant difference between the two groups is found. The weak significance in the manually recorded data showing that the testers removed more cubes in evaluation 2 cannot be confirmed in the simulated results.

The times of the removal of the cubes gives additional information on the performance. For the comparison of the planners and testers the means (M) of the means (m) for the 99 simulations and standard deviations SD are calculated based on the number N of successful solutions - shown in Table 4.4 with the corresponding significance test result (Welch's t-test). None of the compared times shows a significant difference between planners and testers. As before for the manually recorded results this is caused by standard deviations that are large compared to the associated mean values, especially for the earlier evaluations by the low number of data points because not many solutions remove a larger number of cubes.

The primary task of the experiment is to remove the cubes as fast as possible but optionally an extra time bonus (or penalty) can be gained by indicating the color of the cube that is pushed out. The color detection requires either the use of the red color sensor, a blinking light or a combination of both. In contrast to the results given above, table 4.5 gives the times of removing the 3rd cube with the time bonus and penalty of 10 seconds for every correct or incorrect cube color detection. Including this additional

	Eval	1^{st} cube			2^{nd} cube			3^{rd} cube		
		N	$M(m)$	SD	N	$M(m)$	SD	N	$M(m)$	SD
Planner	1	12	21.6	24.4	7	66.4	60.7	6	111.0	69.8
Tester	1	11	22.4	20.1	9	59.6	39.2	9	109.2	62.0
t-value		$t = -0.08$			$t = 0.26$			$t = 0.05$		
p-value		$p = 0.94$			$p = 0.80$			$p = 0.96$		
Planner	2	11	15.6	13.2	8	47.9	50.3	7	86.2	56.4
Tester	2	13	36.0	43.1	10	82.7	67.2	9	117.7	82.6
t-value		$t = -1.61$			$t = -1.25$			$t = -0.9$		
p-value		$p = 0.13$			$p = 0.23$			$p = 0.38$		
Planner	3	13	17.9	18.5	10	57.0	39.5	10	97.6	54.4
Tester	3	14	19.7	13.2	13	48.8	25.9	13	96.0	49.7
t-value		$t = -0.28$			$t = 0.57$			$t = 0.07$		
p-value		$p = 0.78$			$p = 0.58$			$p = 0.94$		
Planner	4	13	22.1	19.2	12	69.3	53.0	12	118.3	82.4
Tester	4	14	28.5	27.4	13	62.6	43.8	13	111.8	60.5
t-value		$t = -0.70$			$t = 0.34$			$t = 0.22$		
p-value		$p = 0.49$			$p = 0.74$			$p = 0.83$		
Planner	5	13	20.7	23.6	11	48.2	42.5	11	78.8	52.1
Tester	5	14	25.1	22.4	13	62.8	50.3	13	100.6	62.0
t-value		$t = -0.49$			$t = -0.77$			$t = -0.97$		
p-value		$p = 0.63$			$p = 0.45$			$p = 0.36$		
Best evaluation										
Planner	3.62	13	19.2	19.0	12	54.3	50.6	12	91.2	69.8
Tester	3.00	14	22.3	16.4	14	52.8	36.1	14	101.3	52.2
t-value		$t = -0.457$			$t = 0.088$			$t = -0.412$		
p-value		$p = 0.651$			$p = 0.931$			$p = 0.685$		

Table 4.4: Simulated removal times of the 1st, 2nd and 3rd cube and significance analysis comparing planners and testers for all 5 and the individually best evaluation of each participant. N: number of solutions that removed the cube(s), M(m): mean of the mean times in seconds and the corresponding standard deviation SD, Welch's t-values and p-values calculated from the corresponding data above

Condition	Evaluation	N	$M(m)$	SD	t	p
Planner	1	6	105.9	65.4	-0.094	0.926
Tester	1	9	109.1	62.2		
Planner	2	7	77.6	65.0	-1.142	0.272
Tester	2	9	119.0	81.9		
Planner	3	10	94.5	60.0	-0.111	0.913
Tester	3	13	97.0	49.8		
Planner	4	12	112.6	88.5	0.004	0.997
Tester	4	13	112.5	60.8		
Planner	5	11	73.8	59.4	-1.041	0.309
Tester	5	13	110.2	63.8		

Table 4.5: Simulated combined times for 3rd cube removal including time bonus and penalty for cube color detection. N: number of solutions that removed the 3rd cube, M(m): mean of the mean times in seconds and the corresponding standard deviation SD, t-values and p-values from Welch‘s t-test

metric into the evaluation also does not show significant differences between the planning and testing groups.

The statistics use the mean of the 99 simulations and therefore do not take the distribution of the data into account. The figures 4.9 show the time distribution using histograms of how often the solutions of evaluations 1 to 5 for every participant removed the third cube within a 5 second time interval. Each line represents one solution/participant and they are ordered such that the solutions that in sum over the 99 simulations removed the most cubes are furthest in the back. Especially in the earlier evaluations many histograms are a flat line as those solutions never removed the third cube. The sum of the bins are equal to how often this solution/participant completed the task out of the 99 simulations within the test duration time of 400 seconds.

For both planners and testers the number of solutions that remove three cubes increases throughout the evaluations and there two distinguishable distributions recognizable. Some solutions remove the third cube within a more narrow time interval than other solutions irrespective of whether they complete the task the most or least often. It is mentionable that the three narrowest distributions are developed by planners.

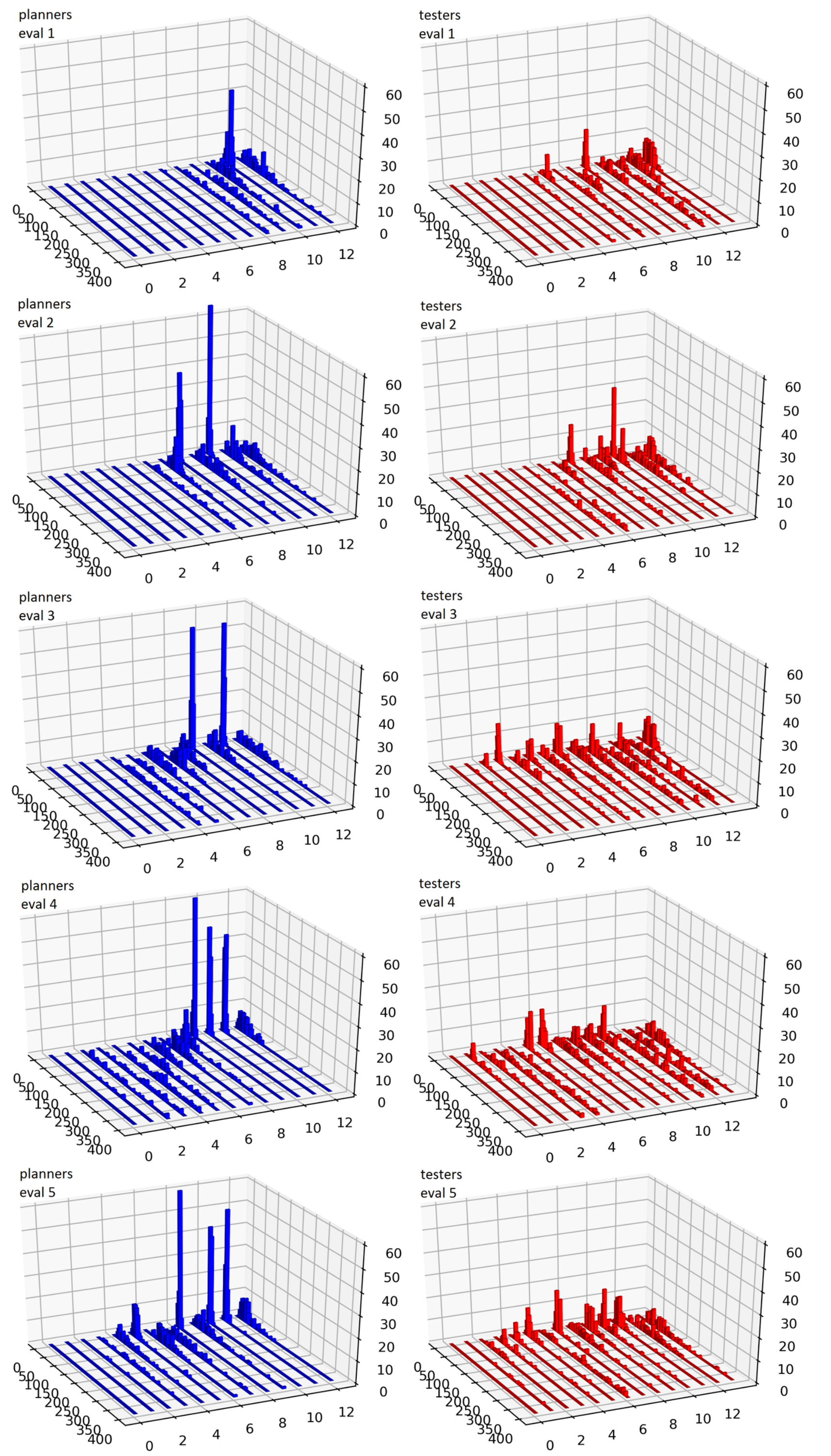


Figure 4.9: Histograms for each participant of the 3rd cube removal time for planners (left) and testers (right). Axis on the left: removal time in seconds (bin size: 5 seconds), axis in the bottom: order of participants

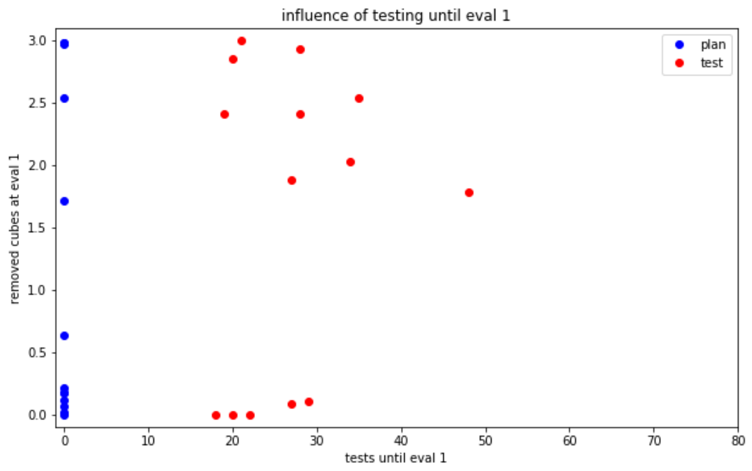


Figure 4.10: Number of tests before the 1st evaluation vs. number of removed cubes

Does the amount of testing influence performance?

So far the comparison between planners and testers relied on the experimental setup that restricts the planners to test within the first 80 minutes of the development phase. Another measure is how often the participants actually test when they are allowed to test and how the amount of testing influences the performance outcome. Figure 4.10 shows the number of removed cubes in the first evaluation vs. the number of tests until the first evaluation as determined by counting how often the participants loaded their code onto the robot. By definition all planners have zero tests and the mean number of tests before the first evaluation for the testers is 26.9 with a standard deviation of 8.2. The results show no significant linear correlation between the number of tests and the number of removed cubes. Testers with few test both achieved high performances and also low performances.

Also regarding the total number of tests up until the 5th evaluation vs. the number of removed cubes shows no correlation (see Figure 4.11). The number of tests after the 1st evaluation is not significantly different between planners and testers (M = 10.8 +/- 4.7 for the planners, M= 12.1 +/- 4.4 for the testers). The complete set of data for all evaluations is compiled in appendix B.2.3. There is also the result for how the number of removed cubes changes between evaluations depending on the number of tests between those evaluations. No significant dependency is observed.

Another aspect to look at is if the amount of typing between tests has an influence on the performance. The longest coherent time where

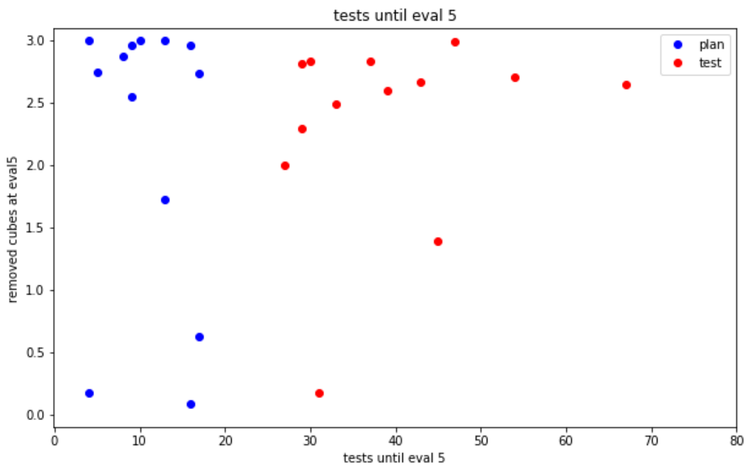


Figure 4.11: Number of tests over the whole time vs. number of removed cubes

the participants can freely type and test is for the testers until the first evaluation. It is also the phase where they start with a blank sheet and have enough uninterrupted time to develop an already working solution. Figure 4.12 shows the average number of keystrokes per test for the testers before evaluation 1. All testers with high performing solutions (removing more than 1.5 cubes) in the first evaluation used between 30 and 80 keystrokes per test (N = 9) while testers with poorly performing solutions (removing less than 0.2 cubes) used 70 to 160 keystrokes per test in average (N = 5). Thus, more concise programming shown by less typing may correlate to better performance. However, it is not clear if the participants with more keystrokes write longer codes or if they type but then also erase more characters.

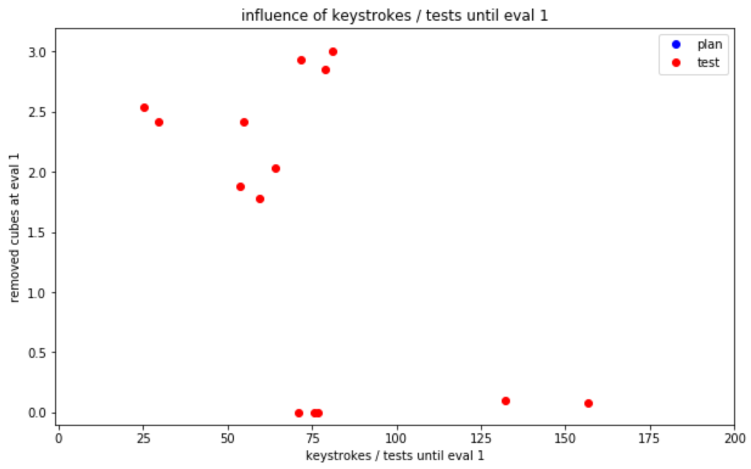


Figure 4.12: Keystrokes per test vs. number of removed cubes in the 1st evaluation

4.9 Influence of testing on concept choice

There are four distinct concepts of detecting the cubes and removing them that were used by the participants. These are:

1. Ultrasound: Using the ultrasonic distance sensor to detect the cubes
2. Blink: Placing the blinking lights as beacons inside the ceiling of the cubes and detect those using the light sensors
3. Hybrid: A combination of using the ultrasonic sensor and the light beacons in the cubes
4. None: Does not attempt to detect the cubes but remove the cubes by driving in a pattern or randomly

One participant in the planning condition in all evaluations drove around randomly without using any sensor to detect the cubes. The rest of the participants used one of the three other concepts (table 4.6). The classification is done by looking which functions appear in the code in the context of cube detection. Using the blink value for cube color differentiation for example is not considered as using the blinking lights for cube detection.

From the description of the hardware handed out to the participant before the programming phase it could be derived that the ultrasonic detection is superior to blink light detection. Thus, it is not surprising that the ultrasonic detection is the most used method. It is striking that only 1

	Ultrasound		Blink		Hybrid	
Evaluation	Planners	Testers	Planners	Testers	Planners	Testers
1	8	7	0	5	4	2
2	7	7	1	5	4	2
3	7	7	1	4	4	3
4	8	7	1	4	3	3
5	8	7	1	5	3	2

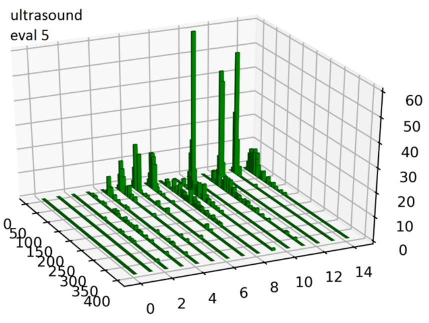
Table 4.6: Concept choice by experimental condition and development between evaluations.

planner but 4 to 5 testers used the blink light method over all evaluations indicating that the testers may have probably studied the instructions less thoroughly. Only 3 participants changed the method from one evaluation to the next, one of them changed and changed back.

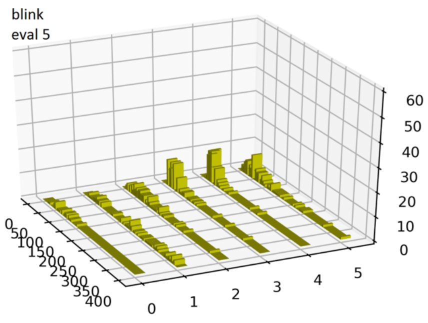
A question to consider is if the different choice of concepts of planners and testers has an influence on the previously shown performance comparison. Figure 4.13 shows the histograms for how often the solutions of the different concepts (planners and testers combined) remove the third cube during the fifth evaluation. Every concept has solutions that solve the task consistently within a narrow time interval but the ultrasound based approach shows more solutions with a narrow distribution of times and is the only concept that has outstandingly narrow distributions where the histogram is two to three times as high as the narrowest distributions of the blink and hybrid solutions.

The task can be solved faster with he ultrasound-based solution. The solution with the fastest mean removal time of the third cube averaged over 99 simulations is 20 seconds with a mean standard deviation of less than 3 seconds for the ultrasound based solution and 52 seconds with a mean standard deviation of 31 seconds for the blink-based solution. This is also confirmed when I program and optimize an ultrasound-based and a blink-based solution myself and compare those. In addition the minimal required code for an ultrasound-based solution that reliably removes three cubes consists of a loop, ultrasound distance measurements and an if-condition (4 lines of code) while the minimal required code for a blinking-based solution is far more elaborate and challenging to program.

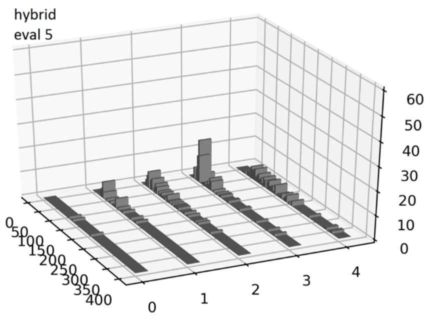
Also in the mean over the participants the ultrasound solutions remove the 3rd cube faster than the blink-based solutions (M: 83 vs. 98 seconds, t = -0.48, p = 0.64) and do so more consistently (SD: 45 vs. 61, t = -1.02, p = 0.32).



(a) Ultrasound



(b) Blink



(c) Hybrid: Ultrasound and blink

Figure 4.13: Histograms of the 3rd cube removal times grouped according to the detection method a) ultrasound detection,b) blinking light detection, c) hybrid, that is both ultrasound and blinking light detection combined. Planners and testers are combined in the graphs, axis on the left: removal time in seconds (interval 5 seconds), axis in the bottom: order of participants

Eval	Planners			Testers			Mann Whitney	
	N	$M(m)$	$SD(m)$	N	$M(m)$	$SD(m)$	U	p
1	8	1.08	1.30	7	1.33	1.30	27	0.500
2	7	1.57	1.45	7	1.17	1.19	19	0.282
3	7	2.18	1.30	7	2.33	0.88	22	0.399
4	8	2.60	0.85	7	2.32	0.98	15	**0.073**
5	8	2.41	1.03	7	2.08	1.02	17	0.111

Table 4.7: Statistics comparing the mean number of removed cubes for planners and testers that only used ultrasound-based solutions N: number of solutions using ultrasound detection, M(m):mean of the mean number of removed cubes over 99 simulations with corresponding SD

Eval	Planners			Testers			t-test	
	N	$M(m)$	$SD(m)$	N	$M(m)$	$SD(m)$	t	p
1	3	113.6	71.5	4	87.6	81.1	0.441	0.678
2	4	80.5	78.2	3	108.8	117.0	-0.387	0.715
3	5	63.9	43.7	7	81.6	53.1	-0.607	0.557
4	7	57.6	39.1	6	127.2	76.2	-2.121	**0.058**
5	7	60.3	52.7	6	110.0	74.7	-1.408	0.187

Table 4.8: Statistics comparing the mean times for removing the 3rd cube for planners and testers that only used ultrasound-based solutions N: number of successful solutions using ultrasound detection, M(m):mean of the mean number of removed cubes over 99 simulations with corresponding SD, Welch's t-test

Although not statistically significant it is still plausible to assume that there is a difference between the performance of ultrasound-based and blink-based solutions and therefore the concept choice can have an influence on the comparison between planners and testers. Therefore, I compare the planners and testers again but exclusively those who chose the ultrasound-based approach. The blink light and hybrid concepts do not have sufficiently many datapoints for a meaningful statistical analysis.

Both in the number of removed cubes (table 4.7) as well as in the removal times (table 4.8) of the 3rd cube the testers outperform the planners in the initial 1st evaluation, however not statistically significant. In the following evaluations the planners remove more cubes (except one evaluation) and do so faster than the testers. The performance difference between planners and testers in the 4th evaluation is weakly significant and the 5th evaluation also shows a tendency in the direction that the planners removed the 3rd cube faster. Surprisingly, for the testers the removal time is slower for the last two evaluations than for the 1st. It could be argued that this comes from the additional solutions that contribute to the statistics as more participants solve the task in later evaluations. But this would not explain why the performance in the 4th and 5th evaluation is lower than in the 3rd.

4.10 Influence of testing on programming behavior

Is there a difference in behavior between planners and testers that may be a confounding factor influencing and/or explaining the previous results? One thing that became noticeable throughout the experiment is that testers start programming sooner after the programming phase started. This is quantified by analyzing the screen recordings. The programming start time is counted as the time when the participant types the first keystroke of meaningful code that can be compiled and executes something testable on the robot. This excludes testing of the keyboard or writing comments only.

When plotting histograms for when planners and testers start typing meaningful code it can be observed that the distribution is not normal but has a tail towards late coding start times. Therefore I use a Mann-Whitney U test to test for statistical significance between the two groups. The median programming start time for planners is 266 seconds and 150 seconds for testers. The U statistic is 46 resulting in a p-value of 0.026, which is below the threshold of significance of 0.05. This means that the testers start programming meaningful code significantly earlier than the planners.

4.11 Influence of experience on performance

A plausible confounding factor that can have an effect on the performance is the prior experience of the participant in programming and specifically in using robots to solve such tasks. Therefore, I aim to control for this by recruiting participants from the same study program and year but also by assessing prior experience in several ways. In the following I will present the experience measurements and their influence on the mean number of removed cubes in the first and fifth evaluation as a proxy for performance.

Self-reported experience

The participants self-reported their experience in programming languages like Python and C as well as their experience in programming Arduino and Lego Mindstorms.

From the post-experiment interviews I know that Python is the most common programming language the participants use in their studies and C is

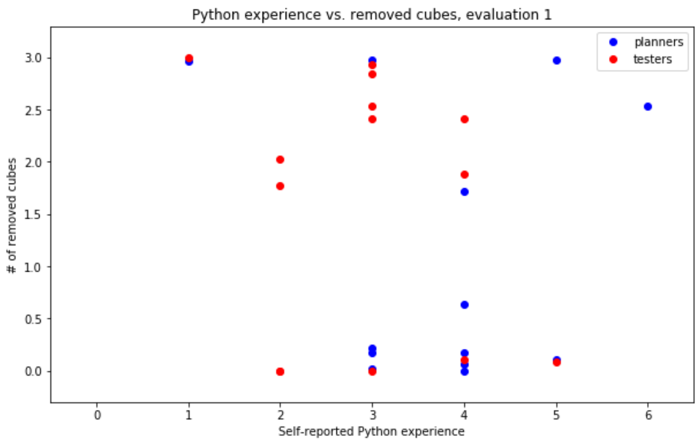


Figure 4.14: Influence of self-reported Python programming experience on performance (number of removed cubes) in evaluation 1.

the closest programming language to NXC (stands for Not Exactly C) which was used in the experiment. In figures 4.14 and 4.15 I show the number of removed cubes in evaluation 1 for the Python experience as a proxy for general programming experience and for the Lego Mindstorms experience as it is, according to my own judgement, the most relevant confounding experience factor for this experiment.

These self-reported experiences do not significantly correlate with the number of removed cubes.

A complete set of graphs is shown in appendix figure B.5 for the other self-reported experience levels and also for correlations with the number of removed cubes in the 5th evaluation - none of them shows a significance. Also calculating the Pearsson coefficient for all self-reported experiences and all evaluations does not show any significant correlation.

Pre-experiment programming test

Immediately after the self-reported experience questionnaire the participants are given a programming test. Figures 4.16 and 4.17 show the number of removed cubes in evaluations 1 and 5 depending on how the participant solved the pre-experiment programming test. The majority of participants solves the test correctly. Initially the test was intended to filter out participants that do not possess the necessary programming skill to conduct the experiment. However, although the test evaluates basic programming skills the majority of participants who failed the programming test developed a well-performing solution (more than 1,5 cubes in average)

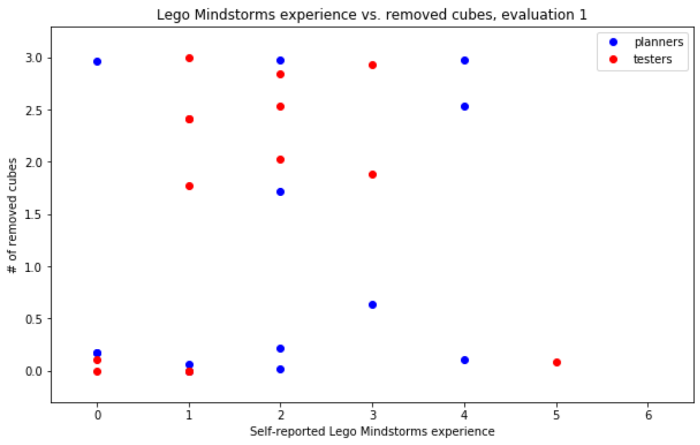


Figure 4.15: Influence of self-reported Lego Mindstorms experience on performance (number of removed cubes) in evaluation 1.

already in the first evaluation whereas half of the participants that correctly solved the test did not develop a well-performing solution (less than one cube in average) in the first evaluation.

The results show no dependency between correctly completing the test and the performance.

Keystroke latency

According to [Thomas et al. 2005] the keystroke latency between certain types of keystrokes correlates inversely with programming performance. Their study found that better performing students have a shorter latency when shifting between keystroke pairs involving special characters like for example brackets, semicolons, operators and other types of characters like alphabetical or numerical characters. The keystroke data were acquired with the self-built Arduino keylogger throughout the programming phase and analysed for these types of keystroke latency and then the median keystroke latency for each participant is calculated. No significant correlation between the median keystroke latencies and the performances measured in the mean number of removed cubes was observed (figure 4.18 and figure B.6 in the appendix).

4.12 Limitations and strengths

The focus of the experimental design is lying on controlling the experimental conditions and procedures as rigorously as possible to ensure a high degree of internal validity. This is the clear strength of this experimental

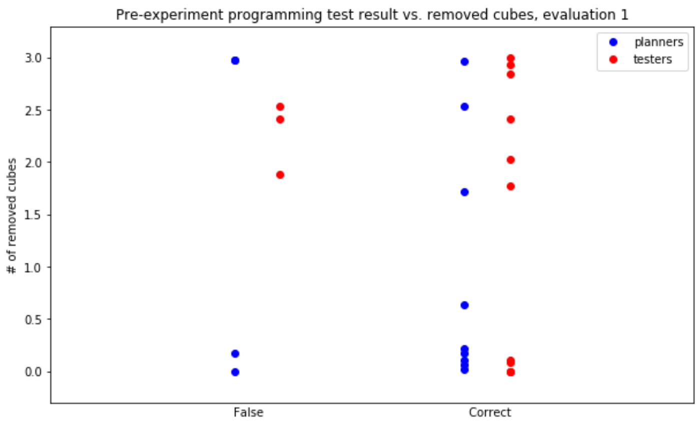


Figure 4.16: Programming test results related to performance, number of removed cubes in evaluation 1

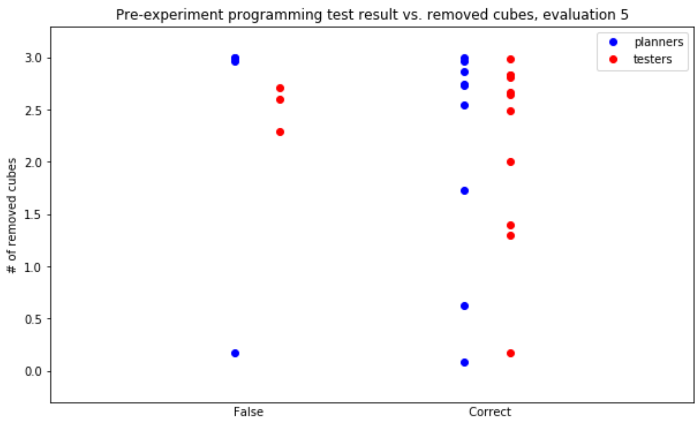


Figure 4.17: Programming test results related to performance, number of removed cubes in evaluation 5

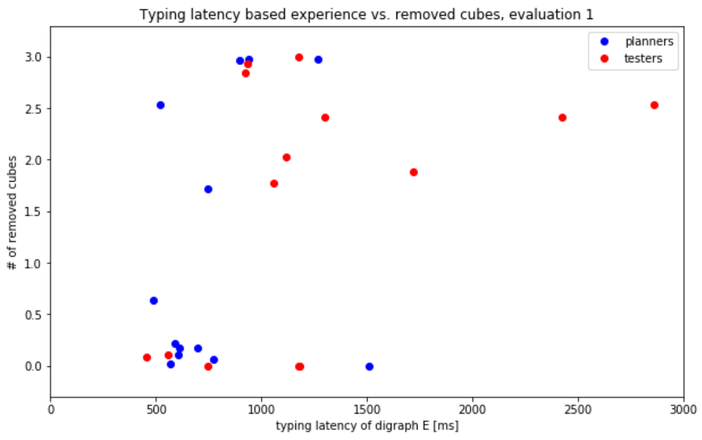


Figure 4.18: Influence of typing latency between special characters and alphabetical or numerical characters on performance (number of removed cubes) in evaluation 1. A shorter latency corresponds to higher experience [Thomas et al. 2005].

setup but it also comes at a cost.

The experimental setup aims to imitate actual CPS development to be externally valid but at the same requires a repeatable conditions and a task that can be finished within a single session of tolerable duration and allows for a quantifiable performance metric. This excludes the majority of actual real life CPS development tasks since most CPS do not remove objects with a Lego Mindstorms robot.

Real CPS development often is complex. The task is also intended to be complex and is indeed complex to some participants but not for all. One participant managed to successfully complete the task without testing and then correcting the solution. This means that the task, if approached by a sufficiently skilled programmer, is plannable and thus only complicated but not complex.

Each participant spends approximately three hours for the experiment. Mainly this time consumption make it difficult to find a large number of suitable participants and gaining sufficient statistical power with the measured effect size, standard deviation and number of participants. This is especially the case if one wants to minimize the differences between participants such as different study programs and seniority. In retrospect, with the knowledge of how large the differences between participants are it may have been worth to also include participants from other school years to gain more data points assuming that the difference between school years is small compared to the individual differences.

The number of removed cubes can only have discrete values between 0 and 3 and therefore the possible resolution is limited to four different values. From single tests the solutions can only be evaluated on this rough scale and may, apart from the also existing uncertainty, not represent the quality of the solution accurately. This can be mitigated by executing the solutions repeatedly and using statistical methods.

When the participants test their solutions the robot behaves non-deterministicly and chaotically. Therefore, it cannot be guaranteed that each participant can get exactly the same testing experience. Even if two participant execute an exactly same solution their observations and therefore the reflections may differ and thereby influence the further development. This effect would be minimized if every solution is tested and observed repeatedly but the time during the experiment is not sufficient for the participants to execute many tests.

In the experiment, as well as in the setup of Thomas et al. [Thomas et al. 2005], a keyboard with a Scandinavian layout was used. However, neither Thomas et al. nor I asked the participants whether this layout is their preferred and most familiar keyboard layout. Especially for programming the English layout is more suitable and allows for faster typing of code. It was observed that some participants were actively searching for keys while programming. Therefore it seems reasonable to assume that some participants were not fully familiar with the Scandinavian keyboard in the context of programming and this influenced their keystroke latency and biased the keystroke latency as a measure for programming experience. The information about the participant's most familiar keyboard layout was not gathered.

4.13 Qualitative results

In this section I provide a qualitative look at the behaviors of the participants and my personal evaluations on how it influenced the outcome of the experiment. In this qualitative assessment I do not attempt to compare events quantitatively and under equal circumstances but instead look at single qualitative cases that can give insights. I will also include participants who are not included in the quantitative results shown previously, i.e. participants from the trial runs (participant numbers smaller than 41) and the participants that are excluded previously due to malfunctions of the experiment setup or participants that withdrew from the experiment. Planners are labelled by the even participant numbers, testers by the odd numbers.

4.13.1 Time spent on planning

As already observed quantitatively in 4.10 the planners use statistically significantly more time before they start coding a solution. What are typical planning activities and how do they effect the results?

After the task has been announced many participants start the programming phase with reading the robot and library documentation again (p22, p24, p25, p26, p50, p59, p60, p68, p70). Now, with the task in mind, the documentation can be read with the intention of planning a solution. Some participants then take notes and write down ideas on paper or as comments in the programming environment (p25, p50, p60, p71). Another explorative behavior is to take a second closer look at the playground and the blinking lights (p46, p50) or push the robot around on the playground like a matchbox car to visualize the planned behavior (also p46, p50). Extensive planning behavior is predominantly observed in planners and to a lesser extend in testers.

4.13.2 Programming and debugging styles

A main factor for the performance of the solutions is how quickly the participants could find and repair errors in their codes. Some participants corrected errors quickly in less than 10 minutes (p24 after 4 minutes of testing, p47 after 9 minutes with multiple tests, p58 after 3 minutes and 1 test) and other participants need considerably more time - up to 50 minutes of testing (p26 after 65 minutes including 20 minutes with testing, p51 after 50 minutes of testing, p52 after 80 minutes including 20 minutes with testing, p53 after 20 minutes with testing, p63 after 40 minutes of testing). Localizing the error in the code certainly depends on the skill of the programmer, how well they understand their code and how confusingly the code is written. While some participants needed one test to observe the malfunction, reflect what causes it and find a solution other participants require extensive debugging. In these cases it is the debugging strategy and behavior of the participant that have a considerable effect on the time needed to repair the error. How quickly an error was repaired largely depended on how quickly the code that caused the malfunction was identified and not so much on the time needed to find a solution.

One way to discover errors quickly is to discover them as soon as possible after they are created. This is done by testing frequently and verifying that after changes were made the robot behaves still as expected (especially p47 and most testers). This is not always possible for planners because at the beginning they are not allowed to test their design changes. If the robot does not behave as expected the error must be in the last code that was written after the previous test. This method finds errors quickly but can be disrupting and time consuming if the participant writes correct code and uses time that was unnecessarily spent on testing because there is no error. An advantage of this approach is that it becomes unlikely that the newly added, untested code includes multiple errors. Finding errors becomes increasingly difficult when the tested code includes multiple, and often confounding errors.

Another strategy to localize errors is by singling out potential code snippets that cause the malfunction. This is either done by commenting out code parts and observing if the unintended behavior (and the intended behavior) is now missing, by copying code parts into a sidewindow (separate windows in the programming environment) and verifying that they perform individually as expected (p54, p55, p56, p63, p69) or by placing the robot on the playground in specific scenarios that test specific aspects of the solution (p46).

The last and often least efficient observed approach is to localize the error by simultaneously correcting it using a "trial and error" method. In this case the participants do not properly reflect on the test observations and do not want to or cannot understand what caused the error and therefore cannot localize the error first and then repair it but instead they change the code, test it and if it then works have consequently localized the error. Since analyzing and thinking through the code to target the error can also be time consuming this approach can be useful if the code section that causes the malfunction is known to be very small and the possible number of causes is low. For example if the last code change that causes the malfunction is of the type "if($a > b$)" then maybe "if($a \leq b$)" corrects the error. Some participants (p47) used this approach successfully. I assume this because the code changes were made very quickly and sometimes it did not solve the problem. Other participants, who had errors in larger code sections and only deployed the trial and error debugging strategy, needed horrendous amounts of time to find the error(s) and they may have used this strategy because they were not capable of localizing the error otherwise.

Writing well-structured code that uses confined and concise functions that serve one distinct purpose and combining these functions to a short main code makes it easier to segregate code parts for debugging. If the code is clearly formatted, has a low level of nesting (code blocks within code blocks) and uses simple conditions it is less confusing to read and understand and therefore simpler to locate the errors.

The participants discover errors through testing and sometimes by mentally visualizing the robot behavior; in some cases already while still not having implemented their intended solution. Patching their solution with workarounds, appending extensive code for edge cases and functionality add-ons without questioning the originally chosen approach often leads to unstructured and complicated code. One extreme example is the condition of a while loop that is already nested in another if condition and another while loop that makes the robot drive forward as long as the downward reflection value is above a certain threshold, the blink value is above a certain threshold for at least 1 second but no more than 5 seconds and the red light reflection value remains above 5. When the participant tested this complicated solution and observed a malfunction it was difficult to trace back the cause. I observed individual differences in code structure between participants regardless of the experimental condition. Although a larger number of planners (p24, p42, p46, p58) than testers (p57) designed a well-structured code (categorized by subjective judgement) it is not clear that planning leads to well-structured codes. Both planners (p26, p48, p52, p70) and testers (p51, p53, p63, p69) write poorly structured and complicated codes. In the few available cases of well-structured and poorly structured codes in this experiment the participants with the well-structured codes performed better than the participants with the complicated codes. Participants p46 and p58 both had well-structured codes and were the two best performing participants in the experiment. Participants p57 and p42 also showed well-structured codes and performed well above average (p57) while participant p42 performed well from evaluation 3 onward which is an average result.

Participants p48, p53, p69, and p70 were participants with poorly structured codes. While participant p69 performed well above average and participant p48 on average, participants p53 and p70 performed poorly. It can only be assumed from observations that the well-structured codes helped to find errors faster and in tendency perform better but in principle it is also possible that more experienced and capable programmers perform better in any case and also happen to write better structured code.

4.13.3 Poor reflections and priority choices

I observed several participants who tested and observed that the robot clearly did not behave as they had intended but then they did not attempt to find the error and instead continued programming other features (p55, p65).

Other participants had a strange choice of priorities as they, for instance, programmed the color detection for indicating the cube color before their solutions successfully removed any cubes at all (p43, p67, p69).

Participant p59 had solved the task 37 minutes before the first evaluation but then ”unsolved” it during evaluations 1 and 2.

4.13.4 Chapter résumé

Summarizing it can be said that generally a more extensive planning and a trend to better structured codes is found for the planners. However, this does not neccessarily lead to a better performance. The individual skill of the programmers and especially their debugging strategy and behaviour considerably determines the quality of the code and thus the performance regardless of the experimental conditions of testing or planning.

Chapter 5

Mixed-method discussion on the use of a prototype-driven approach in CPS development

In this chapter I will combine the insights from the case-studies and the results of the experiment to discuss possible explanations for the observations and to answer research question one. It asks how a prototype-driven approach can be applied and when it is appropriate.

A main idea of prototype-driven development and wayfaring is to build and test prototypes, reflect on the test results and then draw conclusions for redesigning the solution. This assumption is discussed based on the case-studies in the third chapter and from the quantitative results of the experiment described in chapter 4. I have presented how testing of prototypes impacted the CPS's development in the Fibo car example (3.1) and the ATLAS detector (3.2) and that a prototype-driven approach can lead to successful results especially in complex problems like the ATLAS detector. From the case studies I can give examples and argue that early building and testing of prototypes is beneficial for the development because it allows to find unknown unknown and interdependencies sooner and thereby make design and concept changes easier but I cannot quantitatively conclude that early prototyping is beneficial because I have no equivalent comparison case that was conducted with a planning based approach. Therefore, the controlled quantitative experiment uses the early testing versus planning stimulus and aims to compare the two extremes quantitatively under controlled conditions while still maintaining the external validity for CPS development. The following sections explain how to interpret the quantitative results and the last section of the chapter combines qualitative and quantitative insights into normative recommendations.

5.1 No statistically significant performances difference due to large differences between the participants

In the experiment I could not show a statistically significant performance difference when quantitatively comparing the performance outcome of a prototype-driven development approach to a planning-based spec-driven approach. This is mainly due to large performance differences between individual participants in the experiment regardless of their experimental condition. If I make the assumption that the mean performance difference between planners and testers as well as the standard deviation of the performance, hence the individually measured variations between participants, do not change with additional participants then I would have needed in total 54 participants to get a weakly statistically significant result with a p-value of 0.1 or 90 participants for a statistically significant p-value of 0.05 in the comparison of removed cubes. This calculation is already based on the results of the evaluation with the most statistically significant difference between planners and testers. The question is now if this performance deviation between participants is caused by imprecise and fluctuating measurements or if the performance difference between participants is actually this large due to the difference in the participant's programming and problem solving skills.

Although repeatedly executing the physical robot under seemingly same conditions can yield deviating results due to the chaotic nature of CPS it is shown in contribution 7 that already 20 manually executed iterations with the physical robot give a reliable performance measurement that can be replicated with the simulation. From this contribution I know that by using the simulation performance differences between two solutions as small as two percent can be resolved. This means that the performance analysis when comparing two solutions is much more precise than the measured deviations in the experiment, i.e. the uncertainty of the simulation has a negligible effect on the resolution of the performance outcome. There are some results that show a weak statistically significant difference in some measurements. For example do the manually observed results indicated that the testers removed the 3rd cube weakly significantly faster (p-value of $0.096 < 0.1$) than the planners in their individually best evaluation. This result was not replicated by the simulated results which show a p-value of 0.685 for the same comparison. This hints that the weakly statistically significant result is a statistical anomaly of the few manually measured data points in this specific evaluation. Firstly, the result is barely under the arbitrarily set p-value threshold of 0.1 for weak significance and secondly it becomes reasonably likely that some results show low p-values just by chance if a large number of different comparisons are made. In this case it was one of the several performance metrics in one out of the five evaluations. Furthermore, if the effect was genuine, I would expect that statistically dependent variable, the removal times of the 2nd cube, also show some indication of a difference, when the 3rd cube is removed faster. However, this was not found (p-value of 0.35 for the 2nd cube). Another similar example where the manually recorded results show a weakly statistically significant difference (p-value of 0.077) that is clearly not confirmed with the simulated results (p-value of 0.376) is that testers remove more cubes than planners in the 2nd evaluation. I dismiss this result as not genuine for the same reason. Additionally, this result does not occur in the first evaluation, which is the evaluation with the largest stimulus and where I would expect the largest difference. The result also does not show a trend over time that continues into the following evaluations. I therefore argue that the found differences are not genuine because they can neither be replicated by the larger data base from the simulation nor can they be supported by qualitative arguments such as parallelism of the 2nd and 3rd cube or do they indicate a trend that continues with the next evaluations.

In conclusion, if there is an influence of early testing on performance then the effect is smaller than what can be reliably concluded from the data. This does not mean that there is no effect of early testing but if so there are one or more other confounding factors that have a stronger influence on the performance. The resolution of the simulated results is accurate enough to say that it is the actual performance deviation between the different solutions of the different participants that account for the statistically insignificant difference between planners and testers. These individual performance differences overshadow the potential influence of early testing of prototypes in the quantitative results.

5.2 Potential causes for the large performance differences between participants

As the performance difference between participants cannot be statistically explained by the intentionally introduces stimulus of testing versus planning then the remaining question is how to explain the performance difference between the participants. Can I find quantitative and qualitative evidence that explains the large deviations in performance between different participants in a different way?

5.2.1 Programming experience

A confounding factors that was anticipated to influence the performance was the prior experience of the participants in programming and more specifically programming cyber-physical systems. Therefore, the experiment design aims to assess the participant's prior experience with a pre-experiment test, a self-reported experience questionnaire and by evaluating the participant's keystroke latency patterns as shown in subsection 4.11. None of the experience measurements correlates with the measured performance.

The pre-experiment programming test requires understanding the basic programming principles and serves as a filter for finding participants that do not have sufficient programming skills to meaningfully participate in the experiment. However, if a participant answered the test incorrectly I still allowed the participant to continue with the experiment and evaluated their programming skills by observing them. All participants used for the study were familiar with the basic programming principles and therefore included in the study even if they failed the pre-experiment test. One of the best performing participants (p46) returned the test incorrectly. The paper handout with the code of the test is attached in appendix A.1.

The questionnaire asks for programming experience in several languages or equipment. Python is the most taught language in the study program of the participants and therefore a good proxy for overall programming experience. In addition, C and Arduino are syntactically very similar to the language used in the experiment and Arduino and Lego Mindstorms programming experience is asked for because they are directly applicable to programming mobile robots and thus to the Lego Mindstorms task. None of those self-reported experiences correlates with the measured performance and statistically explains the performance differences. The programming experience questionnaire is presented in appendix A.1.

The last measure for estimating the programming experience is evaluating the typing behavior by the method of Thomas et al. [Thomas et al. 2005]. This method measures the habituality of the participants towards programming specific typing patterns. However, these measured typing patterns are not specific to programming of CPS and thus not representative for experience in developing CPS. Although Thomas et al. also used a Scandinavian keyboard layout many experienced programmers use English layouted keyboards as the programming specific keys are faster accessible on those. I did not ask for what keyboard layout the participants are most used to. This would mean that experienced programmers could in particular have long keystroke latencies. This could be a second reason why the typing behavior estimated programming skill does not correlate with the measured performance.

After all I would argue that the three methods mentioned above are a measure for the experience of typing codes and applying the programming principles in pure software projects. This means that at least the type of experience that I measured is not a parameter that linearly influences the performance.

An outcome of the study is that the strategies for organizing the programming flow, testing prototypes at meaningful times, reflecting on the test results and different concepts and debugging are important factors for performing well in this open-ended task. However, this experience cannot easily be captured by tests and questionnaires and thus not quantitatively measured and only evaluated subjectively by observing the participants as described in the qualitative results.

5.2.2 Concept choice

An open-ended task can be solved in several ways and by different concepts. As pointed out in subsection 4.9 the possible concepts perform differently and are unequally difficult to program. Is the performance difference between planners and testers insignificant because the testers made unfavourable concept choices that counter the performance difference potentially caused by the stimulus?

Since the testers chose the inferior performing and difficult to program blink light concept more often than the planners I can assume that the performance difference between the planners and testers would have been more pronounced if both groups had made more similar concept choices. When comparing planners and testers with the same concept choice only the comparison of the ultrasound-based approach has a meaningful sample size. In evaluation 1, where the planners had not tested yet and the stimulus is largest, the testers in average remove more cubes and do so faster than the planners but the difference is not sufficiently large to be statistically significant. In evaluation 4 the trend reversed. The planners remove more cubes ($p = 0.073$) and do so faster ($p = 0.058$) than the testers with weak significance. But with the small sample size a few outliers can influence the result towards a significant difference by chance. These results are also not confirmed when all concept choices are compared although the better concept choice of the planner should also be reflected in the overall performance. Possible reasons for the difference in concept choice are given in section 5.3.

5.2.3 Individual differences in error finding behavior

As described in the qualitative results on programming and debugging styles (section 4.13.2) the participants differed sometimes greatly in how many programming mistakes they made and far more crucially in how quickly the participants could recognize that an error was present, locate it and then repair it.

The seemingly crucial skill that distinguishes many of the well-performing participants from the poorly performing participants is knowing how to combine planning and testing. As described in the wayfaring approach in section 2.2.4 it is a continuous loop of planning and designing, building the code, testing the code and reflecting on the outcome and then planning and redesigning the solution according to what the developer has learned from the test. Taking sufficient time for planning is important as it helps to construct a well-structured code with fewer errors and less ambiguous codes where the errors can be found faster. During testing it is important to carefully observe the robot behavior and relate it to the code to find the malfunctions and recognizing which code part is currently performing correctly or failing. Since the debugging cannot be done step-by-step virtually in a debugger the participants need to be able to mentally understand the code in real time as the robot executes the code and then draw their design conclusions from the observation. For this it is important that newly written and so far untested code is easy to comprehend and does not comprise too many changes in one step such that a malfunction can unambiguously attributed to one location in the code. Tools like cause-dependent tones to mark events in order to know which part of the code is currently executed are very helpful and were used by some participants.

When to test depends on how complicated the code is and, thus, how likely it is to make errors and how quickly an error can be found as well as on the skill of the programmers to observe and relate the observation to the code. What may be a complicated task for highly skilled developers can be a complex task that requires more testing for others.

Although the effect cannot be quantified the study reveals that it was the individual differences in how the participants planned to avoid errors and their strategy for finding and correcting errors that explains most of the individual performance difference between participants. In the end it boils down to how well the design - build - test cycle from the wayfaring approach is executed that most considerably influences the performance.

5.2.4 Individual differences in skill and method preferences

Although all participants are recruited from the same year within the cybernetics study program at NTNU there are personal differences in programming skill and preferences on how to structure the coding process. Some participants are more used to a planning based approach while others use, and many depend on, a more prototype-driven approach. The performance difference can thus be a result of participants having to work with their favoured or unfavoured method.

Furthermore, it seems reasonable that testing is more required when the project is more complex and less predictable whereas a simple or complicated problem is predictable enough and can be planned with sufficient skills. Whether a problem is complicated or complex (according to [Snowden et al. 2007]) depends on if unknown knowns and especially unknown unknowns arise. This obviously depends on the project but also on the experience of the developers - unknown knowns and unknowns of less experienced developers can be known knowns for more experienced programmers. Although the open-ended task used in the experiment is meant to be and was indeed complex for the majority of participants it was merely complicated for some participants as one participant (p46) solved the task successfully at the first test and others solved it without hesitation after quickly observing an error.

The difference in performance is not explainable merely with the programming experience as discussed above, but is also influenced by individual skill and routines for foreseeing potential problems.

5.3 Does testing lead to design fixation on premature concept choices?

As described in section 4.6 there are four distinct concepts that the participants used to solve the task. The two most prominent ones were to detect the cubes with the ultrasound range finder or with the light sensor and blinking lights placed as beacons inside the cubes. The ultrasound solution is far easier to program and outperforms the blinking light solution.

Why did especially testers, who could have tested and compared the different concepts, choose the inferior blinking light solution?

I observed that testers used statistically significantly less time before starting to program their solutions (see section 4.10) and almost never tried out a second concept and remained fixated on the concept they started with. Planners also did not change their concepts before evaluation 1 but this makes sense because without testing they have no reference inducement to make changes. They could have prepared a second solution that they can test immediately after the first evaluation but none of the planners chose to do this.

The planners need to rely on the information about the robot and the library to make mental models of their solutions. The robot behavior of different concepts therefore needs to be imagined and compared mentally. Obviously, mentally exploring concepts does not require typing code and thus takes less time than actually executing and testing a concept. Thus, the wasted time when discarding a mental concept is less than discarding a concept after programming and testing it. Arkes and Blumer [Arkes et al. 1985] describe how continuing an endeavour, once an investment of effort or time has been made, can be explained by the desire to not appear wasteful when discarding the already made progress. With increasing effort and time investment the sunk cost effect intensifies.

While planners, even when making several competing mental models, use little time and no programming and testing effort, the testers invest more time and effort for coding and thus have a higher sunk cost when discarding a solution. I argue that this desire to minimize sunk cost leads to fixation on the initial concept choice.

Although this may explain why testers fixate on their initial concept choice it does not explain why they make inferior concept choices in the first place. Both planners and testers before starting to program have similar initial conditions apart from the instruction that the testers shall test often and early. Since I observe that the testers use statistically significantly less time before starting to program I assume that they think through fewer mental models and are more likely to start programming their initial concept idea. The blink lights are the last part of the task description and may have primed the participants to think about a solution that uses the blinking lights. Although the priming was the same for both experimental conditions the testers may have been more susceptible to it due to the ability and instructed urge to test promptly.

The fixation on the chosen concept may have been substantiated when the testers test the robot and see that their solution can potentially work. The participants rush to correct the errors and proceed with the current concept instead of questioning the overall concept. This leads to a fixation on the often prematurely made concept choice as described in more detail in contribution 10.

5.4 Conclusions on how to develop Cyber-Physical Systems

In this section I will use the previously described and discussed observations to answer the first research question on how and when to apply a prototype-driven approach during the early stages of cyber-physical systems development.

Let us first revisit the insights from the case studies and the experiment to find positive observations that shall be included and negative observations that shall be tackled with the proposed prototype-driven approach. In the case studies I have seen that unknown unknowns exist in complex projects and can be discovered through testing of prototypes. After reflecting on the test results and abductive reasoning the test results influence the design of the next prototype. A series of designing, building, testing and then redesigning lead to an explorative wayfaring-like development path guided by these probing cycles instead of executing a pre-planned design. The aim is to decrease uncertainty of the project by finding unknown unknowns of critical functions, and thus potentially surprising problems that terminate the project, as early as possible.

In both case studies unknown unknowns arose when merging interdependent solutions from different disciplines. Testing of prototypes is thus not only useful for testing individual parts of the solution but also prototypes from different disciplines need to be merged early on to test for interdependent unknown unknowns and reflect upon the entire concept and question whether the chosen concept is a good choice.

Since the aim is to reduce the uncertainty it is sufficient to build low-resolution, and thus low-fidelity prototypes that focus the critical functions to ensure that these criteria can be met before proceeding with optimizing the solution. The rapid prototyping with low resolution is needed to discover unforeseen problems early. From the ATLAS case study I know that results from testing prototypes can be used as arguments for or against design choices were often required by the design review panels. At the same time, using test results to justify a design to opposing proposals of other developers proved to be a useful tool for fostering interlaced knowledge between developers and allowed a bottom-up coordination of large-scale projects. I argue that this bottom-up coordination can react much quicker to design changes than a traditional top-down hierarchy and would hardly be possible without the constant justification of design choices based on prototypes.

From the robot experiment I have seen that just prescribing developers to test at certain times and thus enforcing building and testing of prototypes alone does not yield improved performance of the solutions. The mere quantity of prototypes built and tested does not matter and too extensive testing and redesigning can even have negative effects like an unreflected trial-and-error approach for fixing design errors and patch-work like solutions that treat the symptoms but does not question the chosen concept and can lead to unnecessarily complicated solutions. It showed that when and what prototypes were built and tested matters decisively. In the experiment I observed that some participants had problems prioritizing the critical functions and implemented other non-critical functions before the critical functions were solved. Additionally, I observed that the participants who were encouraged to test used statistically significantly less time before implementing their solutions and, despite being able to develop several concepts and compare them, fixated on the initially chosen and sometimes inferior performing concept. I suspect that this lack of questioning the concept results from avoiding the sunk cost of giving up already achieved progress.

In the following I will propose a method that provides agility for uncertain development projects to adapt to unexpected design changes that also nudges the developers towards a behavior that incorporates the above mentioned advantages and addresses the difficulties.

Assuming the development project to be complex the approach needs to be guided by prototype testing to react to unforeseen problems (unknown unknowns or unknown knowns) while at the same time be adaptive when the complexity of the development changes. The aim is to decrease the uncertainty as soon as possible during the development. This can be achieved by prioritizing critical functions, building low resolution prototypes that can be built quickly and merging these low resolution prototypes of the critical subsystems early. I suggest to define the critical functions and how they can be tested to be satisfied before exploring the solution space. These criteria for the critical functions are not design requirements. For example, some critical functions expected from a Mars rover are to land on Mars and be functional after landing while defining that it needs to have six wheels would be a design requirement. This leaves the solution space open for i.e. deformable tensegrity robots (see contribution 4)

By defining the critical functions first and then exploring the solution space one ensures that the development is driven by the needed functionality and not by a preconceived idea of a potential solution. The question becomes in how many different ways the critical functions can be fulfilled. The focus here lies on "how many" and "critical functions" because it encourages exploring and comparing multiple solutions while focusing on developing the minimum required to satisfy the critical functions to avoid unnecessary feature creep. Avoiding optimization beyond the necessary minimum for gaining the required insights (low resolution prototypes) and the development of unnecessary features reduces the sunk cost when abandoning explored solutions. This has the aim of providing enough time to explore further concepts and to decrease the risk of design fixation.

Unknown unknowns can emerge when combining multiple subsystems. To decrease this kind of uncertainty it is important to merge subsolutions to achieve a system that fulfills the complete set of critical functions as early as possible. This makes it necessary to quasi-simultaneously find solutions to different critical functions and doing so with low resolution prototypes allows the developers find well-working combinations of subsolutions earlier.

Consequently, it is common that the reflection upon the test results prompts design changes and in larger-scale projects this can influence many developers or even groups of developers. In these cases a mutual understanding (interlaced knowledge) combined with a bottom-up decision hierarchy is important for allowing these agile design changes. As shown by the ATLAS case, the early merging, testing and justifying enables this interlaced knowledge necessary for the bottom-up design coordination in large-scale projects.

Since the aim is to decrease uncertainty as early as possible one should not only focus on critical functions in general but prioritize those that are least likely achievable and thus most uncertain. In case these critical functions cannot be fulfilled and the project has to be abandoned it minimizes the resources spent on meaningless developments.

Another major challenge that arose from the qualitative findings of the experiment is that the ability to know when and what to test and how to reflect on the test results has a major impact on the development outcome. Having to define the critical functions and its tests before starting the development leads to a conscious choice when to test and to a preconceived expectation of the test outcome and thus it is easier to realize when the actual test results deviates. By having frequent self-prescribed tests through the critical functions and focusing on only developing the essentials the error can be found and isolated sooner. An additional problem specific to CPS is that errors may not be repeatable due to the non-deterministic behavior of CPS. In these cases errors need to be found and located by a single observation. Repeated execution, either of the actual physical system or by simulation, can give statistical insights to compare different concepts and where to look for improvements but cannot provide the same direct qualitative insights of directly observing a failure qualitatively.

The above described development method is also described in contribution 10.

Chapter 6

Designing and conducting research on the development of cyber-physical systems

This chapter will revolve around the second research question:

How can a controlled quantitative experiment for researching early-stage development of cyber-physical systems be designed and conducted?

The first section of this chapter is on how to enhance internal validity in a mostly quantitative experiment setup. I will name different threats to internal validity and give examples how it can be maintained.

The design of the experiment was not planned out in detail from the beginning and emerged from pilot tests. In the second section I will showcase how the wayfaring approach can also be utilized when designing experiments.

As mentioned in the previous chapter, evaluating the performance of chaotically behaving mobile autonomous robots requires statistical methods. In the third section I will describe the rationale and the methods behind developing the simulation and how it leverages the experimental results.

Last but not least, in the last section I will provide some lessons learned while designing and executing the experiment, give ideas for further improvement and discuss how focusing on internal validity influenced external validity and the statistical power. What compromise is most beneficial for research in this field?

6.1 Designing a repeatable and internally valid experiment setup

As described in the section on mixed-method research (2.1) internal validity is important for attributing measured effects to the intentionally introduced difference between groups of participants (stimulus). Any other difference between participants may knowingly or unknowingly influence the dependent variable and unless the influence of the confounding variable is precisely known the measured difference cannot be attributed to the stimulus. This is especially important for quantitative research as the results cannot be logically deduced but are only statistically inferred. This is why anything but the stimulus is kept as constant as possible for every participant and the confounding variables that cannot be kept constant need to measures such that their influence can be statistically be determined.

The following subsections give examples for possible undesired confounding variables and how their influence was kept constant in the robot development experiment or at least measured or minimized.

6.1.1 Participant selection

One source of differences are differences between participants. Therefore, the aim is to select participants that are presumably as similar as possible in their properties that have influence on the experiment result. In the case of the robot development experiment this means that I aim to find participants with similar programming skills and experiences and intellectual abilities. Thus, I chose participants from one course at one university. However, this means that most participants are in their early twenties, Caucasian and male. If these properties of the participants have an influence then the results are biased and thus not transferable to developers with other properties. Although selecting very equal participants makes the results comparable between participants (internal validity) it makes them less externally valid.

Using monetary rewards for motivating participants to join the study is common. However, if the reward is directly given to the participant it can introduce a selection bias towards participants that are more monetarily indigent but are otherwise less suitable for the experiment. This monetarily introduced selection bias can be mediated by not paying out the reward to the individual participant but to a group that the participants belong to. In the robot experiment the students from the course were fundraising for a course trip to Japan and the reward for participating in the experiment was paid out to the entire group. The individual share of this reward was therefore small since only some members of the group participated in the experiment. This introduces a bias towards intrinsically motivated participants and this may very well have an influence on the experiment results but I argue that this influence is much more equal distributed among the participants than the influence of a monetary selection bias would be.

6.1.2 Controlling the interaction with the participants

The behaviour of the participants is influenced by instructions. For the internal validity it is important that the instructions are given and understood equally for every participant. This means that the instructions themselves but also how they are presented need to be similar. If the experimenter gives the instructions directly to the participant the interaction cannot be controlled and the participant gets an impression of the experimenter that can consciously or unconsciously influence the participant's further behaviour. Therefore, the experiment was conducted without direct human to human interaction between the experimenter and the participant. Instruction were given by email, on paper, by audio and through screens.

The following paragraphs will give some detailed examples of how this can be achieved.

Invitation of the participants

One way that the participant can be influenced by the experimenter prior to the experiment is by already knowing the person or researching him or her, on for example, social networks. This can be avoided by inviting the participant through a middleman that is part of the group the participants are recruited from. This middle person sends out an invitation email and a reminder provided by the experimenter at a defined time before the experiment. The middle person is then responsible for scheduling the participants into time slots but does not assign the experimental condition.

Since the arriving participants will not be welcomed by a human they need detailed instructions on how to find the experiment room. The participants are also informed before the experiment that there will be no direct human-human interaction and that the experiment starts automatically when they enter the experiment room.

It is unavoidable that the participants find out who is responsible for the experiment when reading the consent form at the beginning of the experiment and then may recognize the experimenter by name or at least find out the gender.

Conducting the experiment at different times of the day or on specific weekdays may introduce a bias. For example the participants may have had an exhausting lecture before and are tired, the weather can influence the mood or at different times of the year the participants need to spend more or less time studying for exams. This is why it is important to vary the experimental condition (in my case testing vs. planning) often and without a pattern. I alternated the two conditions but paid attention that one group was not always taking part in the experiment during the morning and vice versa.

Providing standardized instructions

Ensuring that each participant receives identical instructions without human-human interaction can be guaranteed by predefining the instruction, the medium and the timing of delivery. During the experiment the instructions are provided by prerecorded computer generated voices, positioning of the cursor on the programming screen, exchange of paper through a slit that visually conceals the experimenter and text and video on a dedicated instruction screen.

The majority of instructions are provided through the **instruction screen** to the right side of the programming screen. When the participant enters the programming booth both screens are black and when the participant sits down the instruction screen turns on announcing that this is the instruction screen. The timing of this is manual and controlled by the experimenter by observing the participant through a camera. The content is controlled by a semi-automated Power Point presentation. All parts that are not dependent on the participant's behaviour or very critical to be identical for every participant are automated. This includes displaying timers and describing the robot and the task with an audio-visual sequence. Other instructions, like those for setting up the evaluations are situationally triggered by the experimenter following a predefined protocol that adapt to the participant's behaviour. When instructions are given while the participants are not supposed to work on their solution (i.e. during evaluations) the programming screen can be turned off.

The **oral instructions** could be kept identical by recording a human voice and then playing it back. The computer generated voice is chosen because it has no meaning inducing tonality and the tonality does not change between different instructions. It has a very neutral tone to avoid influencing the participant's mood. The voice instructions are either part of an automated procedure or situational and triggered by the experimenter according to a predefined routine. Automated voice instructions are part of the Power Point presentation used for giving instructions on the instruction screen. Situational voice instructions follow a if-then protocol. For example if the participant has not closed the door to the experiment room when entering the programming booth then a voice instruction will remind the participant to do so. A special case of situational voice instructions are given during debugging. The participant receives immediate help if the compiler discovers a syntax error while compiling. This requires that the experimenter pays constant attention to the programming on a duplicated screen in the experiment control area and has spotted the syntax errors already before the participant compiles the code. In this case a voice instruction specifies the cause of the syntax error and the experimenter uses the mouse cursor on the programming screen to indicate the position of the syntax error. For example the experimenter marks the word that has the spelling mistake or encircles the existing bracket whose counter part is missing. When the participant asks questions another voice instruction announces that the participants can only ask questions that can be answered with yes or no and then the question can be answered with a prerecorded "yes" or "no". Only procedural questions and no questions helping with finding a solutions are answered.

A problem with using the text to speech software is that some pronunciations sound unnatural and are difficult to understand. This sometimes limits how an instruction can be formulated and requires reformulation.

Voice instructions are also used during the evaluations. The setup of the playground is set up by the experimenter to ensure equal conditions between participants. To avoid human-human interaction the programming booth is visually separated from the playground area by a sliding door. While the experimenter sets up the playground the programming screen is turned off. When the experimenter has prepared the playground a voice instruction tells the participant to place the robot at the same location where a paper indicator of the robot was placed on the playground to indicate the starting position and orientation of the robot. The experimenter can observe the placement of the robot with a camera live feed. An additional voice instruction exists in case the participant misplaces the robot or does not remove the paper indicator underneath the robot. The participant starts the execution of the program on the robot following a voice instruction count down. This is necessary to synchronize the time taking with the start of the robot.

Instructions and information on paper offer the possibility to provide information that the participant can keep as well as to receive back well-documented information from the participants. Instructions on paper include the consent form, questionnaires, the pre-experiment programming test and the description of the robot and the task for the participant to keep. The papers are exchanged through a slit in the wall above the programming screen. The slit is positioned high enough that even if there is a visual gap to look through the participant could not see the experimenter and on the

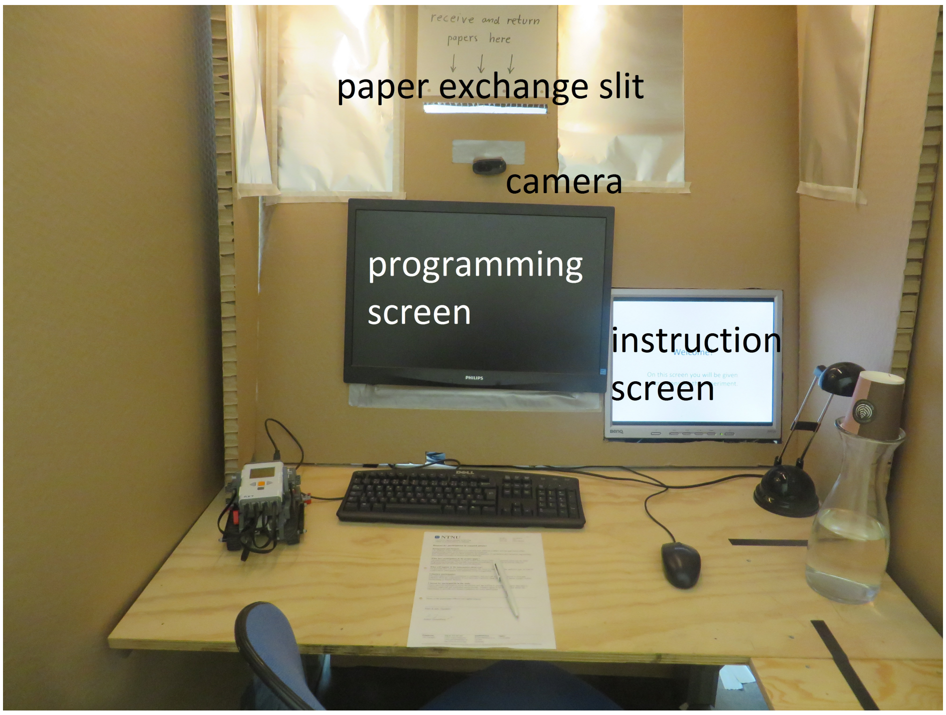


Figure 6.1: The state of the programming booth when the participant sits down for the first time. The aluminium foil is to reflect the ceiling lights for a more evenly distributed lightning.

experimenter's side is a paper sheet that covers the slit but allows to slide through the instruction papers (see figure 6.1). The timing of the exchange is predefined and initiated as a response to the participant's behaviour or with a voice instruction.

Verifying that instructions are understood correctly

In order to ensure that the instructions lead to the same behaviour for every participant it is important that the instructions are understood correctly and similarly.

In the majority of cases the correct understanding is evident from observing the participant's behaviour. Common misunderstanding can be avoided by redesigning clearer instructions or corrected by playing an additional voice instruction. In some rare cases it can be more important to clarify using the experimenter's own voice even if this compromises the restriction of direct human-human interaction.

Since it is important that the participants know the procedure for compiling their code, loading it onto the robot and executing it, this procedure is executed with a test code before the programming phase starts. This ensures that the participants have correctly understood this crucial procedure.

6.1.3 Automation and "foolproofing" of the setup and procedures

The experiment is controlled from the experiment control area as it is shown in figure 6.2. This area is separated from the programming booth and the playground area with cardboard walls. On the camera screen the experimenter can see a live video feed of the participant in the programming booth and a top down view of the playground area. This video feed can also be recorded for post-experiment analysis. This proved useful for verifying and reconstructing participant behavior in cases where the recorded data seemed faulty. In addition, it is useful for getting further qualitative insights for explaining participant behaviour as well as documenting the performance of the robot on the playground. Recording videos allows other researcher to codify the qualitative observations and increase construct validity by allowing for analysing inter-coder reliability [Cohen 2016].

The situational manually triggered voice instructions are played back through a loudspeaker controlled by the computer that processes and records the video feed.

On the duplicated programming screen the experimenter can continually follow the participant's code while the participant is programming. This is necessary to get a qualitative impression of the participants programming behavior as well as finding syntax errors in the code before the participant compiles the code in order to give immediate syntax help as described in the previous subsection. The experimenter also has a keyboard and a mouse to control the cursor on the programming screen. The keyboard is only needed while setting up the experiment and the mouse cursor is used to indicate syntax errors.

The instructions on the instruction screen are controlled from a separate computer that runs a Power Point presentation. This presentation consists of manually advanced slides with situation specific information and timers as well as automatically triggered slides that combine voice and displayed instructions in a predefined timed sequence.

The experiment control area is the only area with daylight and a heating unit to ensure a more controlled artificial lighting and uniform temperature in areas where the participant stays. Also all noise sources such as computers and the experimenter are located in the experiment control area.

In order to guarantee similar conditions to all participants it is not only necessary to predefine the content and timing of the interactions but also ensure that they are executed equally. Checklists and automation are tools to ensure and document consistent execution and minimize the possibility for human error and the ability of a biased experimenter to alter the experiment outcome by consciously or unconsciously introducing inconsistent behavior [Tuyttens et al. 2014, Mynatt et al. 1977].

Arduino keyboard controller and keystroke logger

The Arduino Leonardo (shown in figure 6.3) is connected to the keyboard the participant uses to type code and to the programming PC. By being in the middle between the participant's keyboard and the PC it can be used to manipulate and generate additional keyboard inputs. The Arduino Leonardo is used to:

1. pass on, record and timestamp the participant's keystrokes and save them on an SD memory card

2. restrict the keystroke (F6) for loading code onto the robot while the participant is not allowed to test it with the robot

3. automatically save the code with an incrementing filename when the participant loads it onto the robot or each five minutes for planners. It emulates the keystrokes needed to execute the necessary shortcuts.

4. load the code onto the robot, save it with a corresponding filename for each evaluation when the experimenter presses a button to advance to the next evaluation. This is important for keeping the setup time of evaluations low.

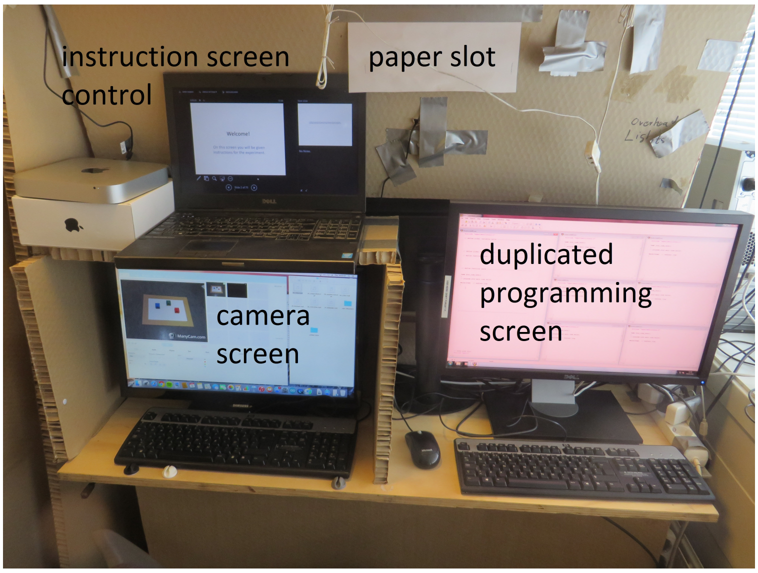


Figure 6.2: The experiment control area where the experimenter can control and influence data acquisition, provide instructions and receive feedback and observe the participant.

Apart from the obvious, namely to record the keystrokes of the participants for potentially finding behavioural patterns in this data, the Arduino helps with simplifying the experiment procedure. It reduces the chance for human error such as forgotten or ambiguous data capturing and it technically restricts the participants from behaving in undesired ways.

6.1.4 Controlling for confounding variables - measuring programming experience

Properties that cannot be controlled but that are strongly assumed to have an effect on the dependent variable need to be quantified because this allows for statistically analyzing the effect that this confounding variable has on the dependent variable. Thereby, the effect of the stimulus can still be quantified independently. It is important though to measure the confounding variable reliably and ideally in at least three independent ways to achieve triple modular redundancy. If these measurements agree, the experimenters can be confident that the measurement is correct and if one measurement disagrees the average of the two closer measurements can be used.

In the robot experiment an assumed confounding variable is the programming experience of the participants. It is therefore measured in three different ways (questionnaire, pre-experimental programming test and typ-

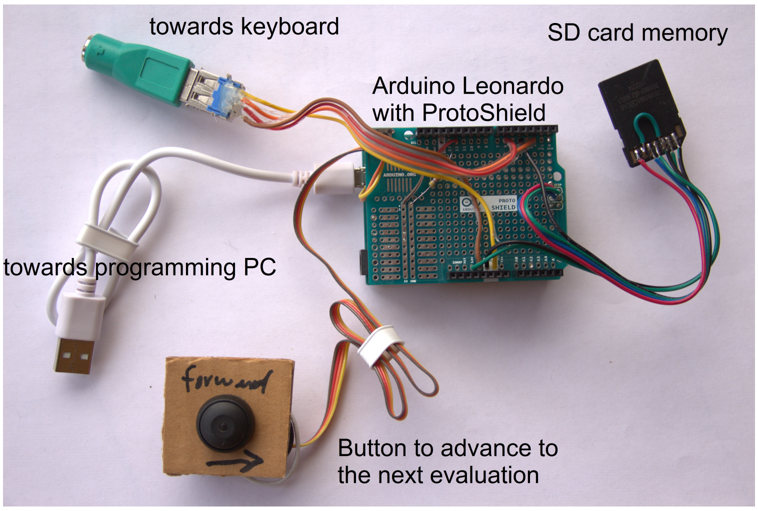

towards keyboard
SD card memory
Arduino Leonardo with ProtoShield
towards programming PC
forward
Button to advance to the next evaluation

Figure 6.3: Keylogger used for capturing and influencing the participant's keystrokes.

ing behavior). In this case none of the three measurements agreed. This means that the three measurements either do not measure the same quantity or are not adequate tools for measuring programming experience. Furthermore, none of the three measurements correlated with the dependent variable, also either because the measurements were not accurate or because the alleged confounding variable was not confounding the dependent variable after all.

6.2 Wayfaring for developing experimental setups

The conventional research approach starts with identifying a gap in literature, an under-researched area with potential for new discoveries, and then finding new research questions and matching hypotheses in this research area which are then tested in an experiment setup. The study design (experiment setup, participants, control variables,independent and dependent variables) is defined along with the data analysis before the data is collected. Initiatives like the Center for Open Science and its Open Science Framework [Sullivan et al. 2019, Nosek et al. 2018] provide opportunities for preregistering studies before data is collected such that the study design and data analysis cannot be altered based on the results. This decreases the likelihood of mistakenly reporting spurious findings that emerged by chance. This is a well-acknowledged approach for confirmatory studies. However, if research is entirely confirmatory it is limited to discoveries within the predefined research question and its hypothesis.

The research presented in this thesis followed this approach for the quantitative results of the final study. However, the earlier preliminary studies and qualitative research in this thesis was more opportunistic and exploratory and did not follow this approach.

The initial research question at the beginning of the PhD was inspired by my personal experiences as part of a product development team in a complex technical challenge. Updated research questions emerged as results of wayfaring while conducting research on the current research question. Wayfaring was not only used for evolving the research question but also to shape and refine the final experiment setup. From one point onward in time the research question, hypothesis and experimental setup was locked, the experiment was executed and the quantitative data was then analysed as previously defined.

In addition, the qualitative data describing the participant's behaviors was collected opportunistically. Therefore, the qualitative data may be biased by selective observation of the experimenter while the quantitative data was collected indifferently of any events and thus does not have a situational dependent observer bias.

However, using wayfaring to design the experimental setup may unintentionally cause inherent biases due to the way the experiment is designed. This cannot be avoided and may also occur in predefined studies with the exception that an experiment setup designed using wayfaring may emerge as more "tailored" towards a certain result. I argue that this is justifiable as long as the experiment setup is well documented because then the external validity of the findings can be related to the experimental setup and understood in this context. Even if the results are influenced by the context, at least the context is known and the results can be interpreted in this context.

The following subsection describes how the initial research question arose, and how it developed in the light of the preliminary findings and gives some examples of how wayfaring influenced the technical side of the experiment setup. The later subsection will put more focus on how wayfaring was deployed for designing the experiment setup.

6.2.1 The evolution of the research question

As part of doing a PhD at TrollLabs it is clear that the research will be in the realm of early-stage product development of complex and tangible technical products. Beyond this framing the topic of the PhD was not further defined at the start of the PhD.

While being part of an ME310 project I experienced how the interactions between team members in an egalitarian hierarchy influenced the intrinsic motivation and the outcome of the project. This notion was further supported and enhanced during a visit of Malte Jung presenting his PhD work [Jung 2011], visiting CERN and reading the reports on collaborations on the ATLAS detector project at CERN [Türtscher 2008] as well as the graduation research project of Joanna Pisarczyk [Pisarczyk 2015] on fablab-like grassroot peer-production communities in the Amersfoort region in the Netherlands. Malte's work looked at how group hedonic balance, which is influenced by the in-group interactions, influences the performance of engineering teams, the CERN reports mention that justification and a notion about caring for others in the project influences intrinsic motivation and cooperation and Joanna Pisarczyk points out the importance of face-to-face interactions on how and who collaborates in voluntary egalitarian projects.

This lead to the desire to quantify behaviors between team members and search for patterns that seem to have an impact on the team's engineering performance. Possible measures to look at include counting words indicating justification like "why", "therefore" or "because", proximetrics (who is how close to whom, body orientation, occupation of space), body posture, gestures, backchanneling, touch and eye contact. Time resolved quantification of these measures allows not only correlating frequency to performance outcome but also entrainment between the team members and comparing the effect sizes of different behaviors.

The main challenges that ultimately lead to giving up this research was that it is difficult to ground within engineering design research as well as psychology and it is therefore hard to publish. Another reason is the extensive scope of the project. Even if all the measurements can be developed and automated this research would still need a large number of data points to find statistically significant factors because the behaviors of the team members cannot be controlled and show a variety of behaviors. This could be improved by using virtual reality technology like Jeremy Bailenson does in his research on social interactions to eliminate, or at least reduce, confounding variables [Fox et al. 2009]. However, this would severely decrease the context in which the team members can be studied and thus limit the external validity and usefulness of the research for engineering teams.

For this reason the next objective was to design an experiment that addresses a more specific aspect and can be well controlled having only one independent variable, is grounded within engineering and less extensive. The resulting proposed experiment compares co-located vs. remote pair programming for solving a complex problem using an autonomous mobile robot. Although with this planned setup it was not possible to study which behaviors have an influence on the pair-programming performance, it was designed as a controlled study to see if there is an effect of co-located personal interactions at all and to quantify its magnitude. For this two identical programming stations need to be set up and connected to enable remote pair-programming. In this process I first set up one programming station to test the programming environment, the instructions, the robot and the task.

In this context, I developed a series of non-complex tutorial tasks to teach the participants how to use the robot with the library and bring all participants to a more even knowledge level. The question arose if a tutorial with purposely designed flaws in the provided example code will lead to better understanding and thus to better solutions while solving the complex task. Participants were invited and divided into groups to test and compare both approaches. The observations from testing with those participants were used to refine the setup and thus the results are not gathered under strictly similar conditions and cannot be compared quantitatively.

The main observation from comparing the two teaching approaches was that there was a large performance difference between participants but this difference was unrelated to the experiment condition. Instead I qualitatively observed that participants who tested their programming solutions with the robot more frequently and learned from these tests performed better. The influence of reflecting on test results seemed evidently larger than any possibly introduced effect caused by the stimulus (introducing flaws into the tutorial tasks). More information on this setup and the observations can be found in publication 6.

This qualitatively observed influence of early and frequent testing of prototypes for learning purposes was the predominant finding from the pre-studies. Thus, it lead to the final research question on how a prototype-driven approach can be applied to early-stage development of cyber-physical systems resulting in the study that is prominently described in this thesis.

6.2.2 The evolution of the task and the robot

The task must fulfill the following criteria:

1. be repeatable under controlled conditions

2. has a dependent variable that is measurable with high resolution

3. be representative for early-stage CPS development

4. be solvable by most participants within the provided time

The first is fulfilled by providing the participants with the same robot, library and setup of the playground. Since all iterations of the task design involved removing cubes, the dependent variables are number of removed cubes and the task completion time.

I want to study the development of uncertain and complex CPS and the task shall reflect this while still be solvable by most participants because otherwise not enough comparable data can be acquired. For more participants to solve the task I can either make the task less complex or provide a library of functions that takes away tedious and time consuming coding. For example, instead of setting the output voltages to motors and using the rotation sensor to program a feedback loop that ensures that the motor actually achieves the desired rotational speed the participants can use the provided library function that defines each motor speed directly. Thereby the participants gain time that they can use on conceptually finding a solution instead of solving technicalities. The aim of the experiment is not to test the participant's ability to apply known knowledge.

The initially intended task was to push out all cubes but in a given order of the colors. This task was incredibly complex, not only because the robot had to detect the color and reject pushing out the cube if it had the wrong color, but also because the robot had to verify that the correctly colored cube was indeed removed from the playground before proceeding with searching for the next cube. While pilot testing this experiment design I noticed that many participants did not solve the task even with a more and more extended and optimized library. This means that I had to make the task progressively less complex.

The next task design was to remove the blue and green cube but never the red cube. This task is considerably easier and already more test participants solved it but it still had the challenge that the red cube can be pushed out while being outside of the field of view of any of the sensors or with the side of the robot when the robot is rotating. Since this task was still too complex I ended up choosing the task that was eventually used in the experiment. The drawback is that this task is far more limited in the number of different concepts that solve the task and thus is far less complex and less representative for the development of real CPSs.

Also the shape of the cubes was changed after realizing in a test that the ultrasound sensor is not capable to detect cubes at an angle of 45 degrees, even if they are place immediately in front of the sensor. As a solution I tested cylindrical objects but those do not have a sufficiently large cross-section to be detectable from a distance larger than roughly 40 cm. As a third and working attempt I used retro reflectors, two wooden boards at right angles to each other, to always reflect a signal back parallel to the incoming beam.

Another feature that was rejected after testing was the "gotoposition" function in the library that allowed participants to tell the robot to drive to a specific location and turn into the programmed robot orientation. This function relied on the rotation sensor in the motors to integrate the driven path and then calculate the direct path from the current location to the desired location. The precision of the robot reaching this location decreases as the robot drives because any measurement error also integrates and increases. The participants expected and relied on consistent accuracy regardless of how far the robot had previously driven and got frustrated if the robot could not provide this precision consistently. Therefore, I removed this function from the library.

The Lego Mindstorms color sensor measures reflection of blue, green and red light and can therefore detect those three colors and I included functions for detecting these colors in the library. After testing I realize that even the red cube reflects more green and blue light than the green and blue cube. Therefore the detection of green and blue cubes is only possible by first measuring if they reflect some green and blue light and then eliminate that they are red by checking if the red reflection value is low. Finding these combination of thresholds to conclude the cube color caused too much confusion among the test participants that they spend too much time to understand and thus not having sufficient time for solving the task. Therefore the green and blue color detection was removed from the experiment.

As one can see it was not possible to plan ahead the interplay between robot and physical setup design, library functionality, task design and the skill of the participants and these parameters had to be tested and adapted repeatedly. This process very much resembles the wayfaring approach.

6.2.3 Wayfaring for instruction testing

Not only the robot and task design needs to be tested and adapted but also the semi-automated interaction with the participants. This includes finding the correct timing for the instructions, adding missing instructions and testing if the instructions are understandable.

The computer generated voice has an unnatural pronunciation that sometimes makes understanding the voice difficult. In these cases the instruction needs to reformulated differently. More commonly an instruction is missing. The behavior of the participants is difficult to predict and therefore special situational instruction are necessary. An example for this was that a test participant sat down in the programming booth without closing the door to the experiment room. As a consequence I had to generate a voice instruction that reminds the participants to close the door.

Another good example of unpredictable participant behavior is when a test participant used his own chalk to take notes on the blackboard that was hanging on the wall after I had already removed the existing chalk to avoid this.

6.2.4 Resolution of the measurement

Initially I assumed that almost all participants would solve the task already before the first evaluation and that the task completion time is a good smooth metric to measure the performance. As this was not the case also the number of removed cubes was introduced as as performance measure. Difficulties partly resulted from an unexpected degree of non-deterministic behavior of the robot. This weakens the findings when based only on the result from a single evaluation. It can be overcome by a deterministic simulation which can run the solution numerous times and thereby enable statistics. Needing statistics and the simulation was thus not anticipated and is an example for an unknown unknown.

6.3 Leveraging quantitative results with a simulation

While designing the robot and the task I assumed that the robot behavior is, if not fully deterministic, at least sufficiently consistent that a single execution of the code is enough to evaluate the performance of the solution. While this may be the case for simple well-designed solutions, for intricate solutions the performances differed greatly between evaluations and are influenced severely by small perturbations in the setup. This made it evident that the manually recorded results during the experiment cannot provide a realistic performance assessment by themselves and a statistical method was required to differentiate the performance of different solutions. Although it is theoretically possible to do this assessment with numerous runs manually with the physical robot it is firstly very time consuming and tedious. Secondly it is difficult to maintain similar conditions for every evaluation. A detailed description of the development and validation process can be found in contribution 7.

6.3.1 The benefits of using a simulation to leverage results of a human subjects robot programming experiment

The simulation is deterministic, meaning that if rerun under the same conditions the results will be exactly equal. This is important because then the start conditions can be kept equal for testing every solution and thus the simulation provides the possibility to compare the solution fairly under the same circumstances and thus with higher internal validity. The physical robot for example is subject to battery drainage, inconsistent levels of friction due to wear and tear and inaccuracy in the initial placement of the cubes and the robot and thus a manual evaluation with the physical robot cannot guarantee similar conditions. Achieving this would require frequently exchange of the batteries, calibrating the driving speed and multiple executions of the robot from one starting configuration to minimize the statistical error of inaccurate robot and cube placement. Since the simulation is deterministic it only needs one run per starting position. Not only can the simulation run faster and automated, it also requires no repeated runs from a starting position to gather useful data.

Thus, with the simulation the experiment can be evaluated within a day instead of months and from a larger variety of starting positions. This increases internal validity because more starting positions reduce the risk of favouring certain solutions by chance.

An additional benefit is that the simulation can purposefully and in a controlled manner introduce perturbation to for example starting positions, motor speeds or sensor readings and evaluate the robustness of the solution to these perturbations. This would evaluate how accurately the performance of the solution for each run can be predicted. This was not the criterion for evaluating the performance in this experiment but it may very well be a meaningful criterion to look at when evaluating the performance in other settings.

6.3.2 The development of the simulation

The development of the simulation was unique in the sense that usually a simulation exists before the physical robot is constructed and it is used to predict the behavior of the physical robot before it is physically implemented. In this case the physical robot was developed and refined using physical tests and constructed without a prior simulation and then the simulation was developed to reproduce the behavior of the physical robot as accurately as possible.

As input to the simulation the measurements of the environment and from the robot data sheet are used and each property is modelled separately (i.e. driving speed, angular speed when turning, sensor values, ...). When the simulated behavior did not match the actual behavior of the physical robot firstly the digital representation was made more accurate and only as a last resort heuristic correction were introduced. For example did the simulated ultrasound sensor show too short distances to the cubes when the cubes were simply modelled as squares. After implementing the retro-reflector configuration and using ray casting to emulate the ultrasound beam the simulated ultrasound distance was accurate. On the other hand, the simulated color reflection sensor values were too small for short distances between the sensors and the cubes even when using the ray casting method. Although digitally adding an additional reflecting circle around the center of the cube does not represent the geometry of the physical setup, adding this artefact made the simulated color sensors accurate. This method is acceptable since the aim of the simulation is not to model the real world in order to explain the occurring phenomena (like for example atomic simulations in theoretical physics) but to reproduce the robot behavior as accurately as possible such that it is useful for evaluating the robot performance. The simulation is written with C sharp in Unity, a game development software that includes a physics engine that simulates the physics interaction between the robot and the cubes. The simulation uses a two dimensional top-down view because the third dimension is not necessary for describing the physical setup. The participant codes can be entered into the simulation after syntactic adaptation.

6.3.3 Validation

The aim of the simulation is to provide a tool for evaluating the solutions written for the physical robot in an exact, repeatable and timely manner. We know that the simulation can evaluate the solutions quickly and repeatedly (deterministic) but it also needs to be an accurate representation of the physical robot behavior. This needs to be validated before the simulation can be used as an evaluation tool.

The validation is done in three ways: Firstly, by comparing each property individually. This is already done as described above during the development. Secondly, by qualitatively comparing the robot behaviors. Does the robot interact with the cubes in a similar way and is the simulated trajectory qualitatively indistinguishable from a set of observed trajectories of the physical robot under similar starting configurations?

The last, and for the performance evaluation most critical litmus test is if the simulated performance matches the manually recorded performance statistically. For this example codes are both repeatedly executed on the physical robot and simulated and the means of the performance measures are compared. Three example codes are used. They are selected such that they cover a wide spectrum of the functionalities and are representative of the solutions developed by the experiment participants. From the three example codes two used the ultrasound sensor for cube detection and the other solution used the blink light beacons. All three solutions solved the task reliably.

Before it is possible to judge if the simulated results are representative for the physical robot's performance it is necessary to know how precisely the performance of the physical robot can be determined. The variance in performance of the physical robot is the limit to how precise the simulated performance can match. For this the three solutions were repeatedly executed with the physical robot from the same starting configuration each time to gather statistics on the relative variance of the task completion time between reruns. To ensure that the chosen starting configuration was not biased this procedure was repeated from two additional starting configurations.

For the actual comparison between the physical robot performance and the simulated performance the physical robot is executed once with each example solution from 20 different starting positions. The same 20 starting positions are then programmed into the simulation and executed as well. The result shows that the simulated task completion times are between 70 and 78 percent of the manually recorded task completion times of the physical robot. Although the simulated times do not match the manually recorded times the relative difference between the solutions is sufficiently similar regardless if determined manually or by simulation and the relative variance of the differences between the solutions is smaller than the variances determined as described previously for the repeated reruns of the physical robot from the same starting position. The relative inaccuracy of the simulation is thus smaller than the inaccuracy caused by the non-deterministic behavior of the physical robot which is the limit of how precisely the simulation can be validated. For the comparison of performance as it is needed in the experiment the relative comparison between solutions is sufficient and thus the simulation can be used for this.

Lastly, it was verified that the 20 starting positions for the manual recorded task completion times were unbiased by comparing their average result to the average result of the simulation from 99 randomly chosen starting positions. An interesting question is how small performance differences between solutions can be resolved with statistical significance. It turns out that a 2 percent slower solution can be distinguished. This was done by comparing a solution to the same solution with 2 percent reduced motor speed. This resolution is more precise than the variance caused by the non-deterministic behavior of the physical robot and hence this remains the limiting resolution factor.

6.3.4 The simulation as part of a well-defined experiment setup

When replicating the robot experiment setup to reproduce the study described in the previous chapters it is important that the conditions are as similar as possible to the original study. The simulation can be executed by other researchers under equal conditions and thus produces comparable results. This cannot be guaranteed if the evaluation is done manually because small differences in the physical setup can easily occur and influence the evaluation of the solutions.

6.4 Lessons learned from the robot experiment

In this section I summarize aspects and details of the experiment that worked out optimally as well as of those that, in hindsight, could have been improved upon.

6.4.1 Timing of surveys

Surveys are a useful way of collecting data, however, they appeared to be annoying for some participants, take time and take mental capacity away from the rest of the experiment. They interrupt the experiment flow and therefore no written questionnaires were used during the programming phase but either in the beginning or at the end.

The programming experience questionnaire had to be placed at the beginning of the experiment in order to avoid that the participant's judgement is influenced by the programming used in the pre-experiment programming test.

Whereas it proved beneficial to collect all questions that cannot be influenced by what happened previously in the post experiment questionnaire. Thereby, asking for this data may annoy the participant but this cannot alter the experiment results anymore.

6.4.2 Selecting participants for pilot studies

As mentioned before, it is very difficult, if not impossible, to design a human-subject experiment without testing the study setup and adapting it to unforeseen problems. In the end I used roughly as many participants for testing different iterations of the experiment setup as participants used for the actual study. In order to collect internally valid data it is necessary to not make any changes while collecting the data that is used to derive quantitative, comparative conclusions. Therefore, the study design needs to be thoroughly tested in pilot studies and the question arises whom to use as pilot participants.

On the one hand, one wants to test with participants that are as similar as possible to the final participants but at the same time it is often difficult to find sufficiently many suitable participants for the final study and the same person cannot participate twice. It can therefore be meaningful to use pilot participants that could not be part of the final study anyway to have more available final participants. Another consideration for choosing pilot participants is what you can learn from them. This includes the usefulness of the feedback they can give as well as their abilities. In my case I early on used pilot participants that had experience with designing human-subject experiments themselves in order to get a well-qualified feedback on the general study design from a scientific point of view. Their feedback was useful for determining what research question and which hypothesis are interesting and valid, how they can be tested, what confounding variables to pay attention to and learning from the experiences they had made in their studies.

After having determined the purpose and principle objective of the experiment, I used pilot participants with a background in programming to better design the task and the library as well as to experience unexpected behaviors to adapt and add instructions. I recruited these pilot participants from students who took the mechatronics course and from the same study program as the participants for the final study but from one year higher. I also included extreme cases like professional programmers with several years of working experience.

During the last tests before the final study I used some participants from the pool of suitable participants for the final study because I judged that it is more important to have a well-calibrated experiment setup that works with the target group and loose some potential participants than executing the study with a task that is either too easy or too difficult and then in the end provides less useful data because the performance data is saturated. The pilot participants received a cinema ticket as a sign of appreciation.

I noticed that pilot participants that have an emotional connection to the experimenter and study designer, like friends and colleagues, are more engaged with testing the experiment and give much more, critical and thus useful feedback and suggestions for improvement. They tend to spend more mental capacity and time helping and can also more easily be contacted again and asked for their opinion after having made changes to the setup. This proved very useful especially during earlier stages of the study design.

This stepwise approach - first pilot participants experienced in human studies, then those with good programming experience, then those at the level of the final experiment - proved to be very successful to define the design and all details of the final experiment and ensures its validity.

6.4.3 Using a robot with an existing and well-validated simulation

The Lego Mindstorms robot and the library used in the experiment were deliberately designed for the task and adapted and tailored to the needs of the experiment. Building the robot from the ground up meant a large degree of freedom and flexibility but also consumed a large amount of time, especially building and verifying the simulation.

This time could have been reduced by adapting an existing mobile autonomous robot with an approved, well-established and verified simulation. Even if the self-built robot executed equally repeatable as an existing and established robot and the simulation was equally accurate, well-calibrated and verified the self-built robot and simulation requires extensive description for making the system replicateable for other researchers. Moreover it makes it more difficult to publish the robot and simulation design than referencing a well-known and acknowledged robot-simulation combination.

If I was to design such an experiment again I would rely more on an existing robot-simulation combination.

6.4.4 Trade-off between internal and external validity

When designing the robot experiment the main priority was a high internal validity while still maintaining external validity. However, this often comes as a trade-off between the two. To answer the research question to which degree early prototyping increases the development performance the task needed to have a quantifiable performance metric. In addition, it had to be solvable within the given time frame. This limits the task design that is aimed to represent complex nature of developing mobile autonomous robots. In the real world the development is much more complex than finding and pushing objects outside a rectangular area. In this sense, designing a task that is solvable under controlled conditions, which is necessary for internal validity, restricts the external validity because it limits the answer to the constrains set in the experiment and, thus, weakens the conclusions that can be drawn from the results.

The same argument applies to the selection of participants - the very homogeneous group of participants (early twenty year old Norwegian males in the middle of their cybernetics studies). They may be not representative in all aspects for other groups of programmers, e.g. professionals and thereby limit the external validity. In case of reproducing the experiment this selection of participants makes the results between experimental conditions internally comparable but limits the applicability of the finding to developers that are very similar to the study participants.

Making the experiment internally valid was time consuming both in terms of time needed to design the experiment (setting up the programming booth, designing and testing automated instructions, developing the simulation) as well as time consuming while the participant is present (multiple experience measurements, setting up evaluations following automated instructions, answering questionnaires and tests).

The long experiment duration meant that it was challenging to find many participants. A study with more participants could have been more statistically powerful and thus may have provided a higher resolution for determining effect size.

Another trade-off between internal and external validity arises in terms of team work and cooperation. Most development projects are conducted in teams and for higher external validity the experiment could have been conducted with several participants developing in cooperation simultaneously. I deliberately chose to conduct the experiment with individual developers for two reasons. In teams of two pair-programming developers the human-human interaction needed for cooperation introduces a difficult to control or measure confounding variable that would significantly decrease internal validity and the amount of data points halves. Finding sufficiently many suitable participants was a limiting factor.

Having a well-defined and documented experiment setup provides other researchers with detailed information to quickly replicate the experiment setup with less effort. They can either re-run the experiment with very similar participants in order to check if the findings are reproducible and thereby increase internal validity or with participants with different backgrounds to check if the findings also apply in different contexts and extend the applicability of the findings. Thus, providing a detailed description of a well-defined setup with a naturally highly restricted external validity may overcome these limitations by making it easier to rerun the experiment in new contexts and thereby extend external validity. Another example of a well-defined experiment setup that can enhance external validity through enabling other researchers to run the experiment in different contexts is described in contribution one [Kriesi, Steinert, Aalto-Setaelae et al. 2015]. In this case, the experiment was performed in several different laboratories, i.e. in different locations and even different countries. The equipment needed for the experiment was distributed together with precise descriptions on how to conduct the experiment so that a third party could execute the experiment unambiguously. If the hypothesis of the experiment can be concealed to the executing experimenter such a distributed experiment can thereby become a double-blind setup.

The pros and cons of predefined interactions

Useful tools for internal validity are predefined interactions as they create similar conditions for all participants. But the downside is that the experiment executions becomes rigid and less able to react to unforeseen events. Furthermore, predefining reasonable interactions needs extensive pilot studies and is therefore time consuming in the experiment development phase.Despite comprehensive preparations there were unexpected events during the robot experiment, such as the participant that brought his personal chalk to write on the blackboard (see 6.2.3), people knocking on the door and entering the experiment room unexpectedly, unpredictable participant behavior or rare failures of the robot that require context adapted interactions. These events are so rare and context dependent that a suitable instruction cannot be predefined. Thus, predefined interactions can enhance internal validity in the absense of these unexpected events but in return make experiment design more time consuming and execution less able to react to unforeseen events.

Does the anonymous observation influence the participant behavior?

Although avoiding direct human-human interaction between the participants and the experimenter increases internal validity it is unclear if this in itself effects the outcome of the experiment. The Hawthorne effect describes the change in participant behavior due to being observed in the experiment [Parsons 1974, McCambridge et al. 2014]. The question arises how does this change if the experiment is conducted without direct human-human interaction. Some participants reported the experience as "weird" when asked after the experiment but also that they quickly got used to it and were not thinking about it anymore. Anonymous observation decrease external validity because in most work environments the interactions are not automated and the work is not monitored by an unknown and hidden observer. As the participants of the robot experiment got quickly used to these conditions and felt only slightly deranged the effect is estimated to be minor and the profit of increased internal validity outweighs this drawback. This evaluation is based on qualitative observations and post experiment interviews. Further research is needed into this direction.

When is it meaningful to optimize for internal validity?

In a retrospective view on the study design it can be said that the experiment indeed has a high degree of internal validity which by itself is an important property especially of quantitative experiments where the explanatory power inherently comes from inductive reasoning. However, the research might have been more efficient and meaningful if more focus would have been on gathering results from more participants and potentially conducting multiple consecutive experiments with less internal validity but also with less effort for each study.

Collecting data with a larger number of participants who are less equal introduces confounding variables and may increase the standard deviation of the dependent variable and thus decrease the statistical power but may nonetheless be useful. The participants have to be randomly assigned to the experimental conditions so that the introduced confounding variables contribute equally to both experimental conditions and the effect averages out. If the effect of introducing confounding variables, and thus adding more deviations between the participants, is less significant than the gain in statistical power due to the larger number of data points then it may be beneficial to use a larger set of less equal participants. Another benefit is that the result then becomes more externally valid.

The hypothesis of the robot experiment (testing vs. planning) resulted from observations and pattern seeking during a pilot study and thus the research was rather exploratory than aimed at confirming a well-established, previously researched hypothesis. Therefore I now assume in hindsight that more insight would have been generated in the same time with several simpler, faster and less internally valid experiment setups using a larger number of participants. The used approach using the highly internally valid experiment design was to a certain extent inadequate for the partly explorative research.

6.4.5 Discussion on using wayfaring for developing highly controlled experiments

Is wayfaring a useful approach for developing highly controlled human-subjects experiments? Yes and no! Yes in the sense that I believe that constructing the robot experiment the way it was done required the discovery of many unknown unknowns and reacting opportunistically to these unexpected incidents. Testing was especially needed for exploring possible participant behaviors and how to deal with them and it was crucial for designing an automated experiment sequence.

Setting up research questions and hypothesis while wayfaring and adapting the experiment setup based on observations of pilot studies bears the danger of unknowingly introducing or enhancing a bias that is then confirmed in the final study. The wayfaring approach may therefore lead to a Feyerabend-like attitude of "anything goes" [Feyerabend 1993] that may be suitable for exploratory studies. In the robot experiment case I selected the stimulus of testing versus planning after observing that testing prototypes early seemed to make subjectively observed difference. I did not use this pilot data that indicated the importance of early testing for the final conclusion but it may be that the experiment setup I used is inherently sensitive to early testing and thus not representative and externally valid in this regard. In this particular case I can conclude from the final results that there was no strong bias since the results did not support the hypothesis that early testing of prototypes makes a statistically significant difference but in general this danger of introducing a bias through using wayfaring must be considered.

If wayfaring is applied for finding patterns and generating hypothesis then the study has to be exploratory whereas the hypothesis of a confirmatory study needs to be planned and stipulated with the aim of falsifying the hypothesis [Popper 1935]. Wayfaring in this case is useful to fine-tune the interaction with the participant but not for adapting the hypothesis. In summary, wayfaring is useful for exploratory studies and for developing the automated interaction with participants but not the general study design in confirmatory studies.

6.5 Thoughts on the future - a fully automated experiment setup?

The robot experiment showcases how confirmatory studies could meaningfully use automated participant interaction. In confirmatory studies it is applicable and especially useful because the anticipated observable pattern is already known but one wants to eliminate as many confounding variables as possible to become more certain that the introduced stimulus caused the observed effect.

An experiment setup that is entirely automated is effectively a double-blind setup because even though the experiment designer knows the experimental condition that the participant takes part in, the experimenter cannot alter the ongoing experiment in any way. Since data capture is then also automated the collected data and data format is known and the data analysis can also be predefined and automated. The experiment can then only be biased in the setup itself.

Not only is developing a fully automated experiment setup incredibly time consuming, it also comes with some other problems that need to be further researched. Firstly, although it may become possible to give fully automated instructions, there needs to be a much better understanding of how these instructions are perceived and understood by the participant. Secondly, it is not well researched how such automated instructions and data capturing influence the participant's behavior and may lead to an altered

Hawthorne effect.

On the other hand a fully automated experiment setup is by definition well-defined. The experiment setup needs to be unambiguously documented and can therefore also be unambiguously setup at a different time or location. This can allow for distributed experiments like the one in contribution one [Kriesi, Steinert, Aalto-Setaelae et al. 2015], which in return means that participants can be recruited at many different locations and thus in larger numbers. Potentially even the time effort for conducting the experiment is minimal if no experimenter needs to be present for conducting the experiment.

Chapter 7

Summary

This thesis presents research in the realm of early-stage product development of cyber-physical systems. It includes research that looks at how to use a prototype-driven development approach to better develop cyber-physical systems and also research on a meta level about how to conduct such research. The research on the development approach answers the first research question and is based on case studies and a quantitative human subject experiment. It leads to normative recommendations that are useful to developers. The meta level research reviews the design and execution of the same quantitative human subject experiment from a researchers perspective leading to insights on how to design and carry out such experiments and answers research question two.

RQ1: How can a prototype-driven approach be applied to the early-stage development of cyber-physical systems and when is it appropriate to use this approach?
Prototype-driven development approaches use prototyping iterations to advance the design. Testing of the prototypes provides the insights to abductively reason how to change the design in the next iteration. This leads to a wayfaring-like design process that cannot be planned and hence the design evolution can only be explained in hindsight. This helps in discovering and responding to unforeseen problems (unknown unknowns and unknown knowns) quickly and as soon as possible. From one case study it becomes evident that parallel prototyping of different disciplines and domains is necessary to discover unforeseen interdependencies earlier. The other case study of the development of the ATLAS detector at CERN shows that a wayfaring-like prototype-driven approach can also work in large-scale projects where it is impossible for a single person or a small group to oversee

the entire project. For this a high degree of design justification through for example prototype testing creates interlaced knowledge between domains which was used to coordinate design decisions in a decentralized bottom-up approach. Testing prototypes then does not only provide insights for further designs but serves also as means of design justification in large-scale projects. From the case studies one can qualitatively observe and assume that early testing of prototyping is essential and leads to better performing solutions. The quantitative human subject experiment studies the influence of early testing of prototypes on the performance outcome of the development of a mobile autonomous robot. The experiment does not quantitatively confirm a statistically significant influence of early testing on performance. This is mainly due to large individual differences between the participants. From qualitative observations during the experiment I observed large behavioral differences between the participants. Most influential are differences in debugging and error finding behavior and the ability to reflect on observations while testing the prototypes. Mobile autonomous robots, although programmed with a determinsitc code, nonetheless behave non-deterministic due to the interaction with the environment. Correlating the observed robot behavior with the context of the environment and the code in real time seemed to be a major difficulty for many experiment participants. The ability to reflect depends on the individual reflection skill as well as on the coding, debugging and testing behavior.

From the mixed-method analysis, I can recommend early testing of prototypes in complex cyber-physical systems development. However, an unreflected and enforced early testing of prototypes does not necessarily lead to better results and can also bring disadvantages. These are for instance confusing patchwork-like solutions that are difficult to reflect upon and fixation on premature concept choices due to the sunk cost effect. Meaningful selection of what kind of prototypes to build depends on the complexity of the project and individual preferences of the developers. Making good choices for what, when and how to build and test is not trivial.

RQ2: How can a controlled quantitative experiment for researching early-stage development of cyber-physical systems be designed and conducted?

The second research question is answered through a meta-level reflection of the quantitative robot experiment. In quantitative experiments insights are generated from statistical inference by introducing one stimulus and quantifying the dependent variable. This is important to ensure the internal validity that any observed effect can only be contributed to the intentionally introduced stimulus - confounding variables need to be avoided or at least controlled. I present many examples and some lessons learned on how internal validity can be improved in such quantitative human-subject development methodology experiments. Possible confounding variables that are addressed are individual differences between participants, direct human-human interaction between the experimenter and the participant, inconsistent presentation and understanding of instructions, imprecise rules for conducting the experiment and in the case of the robot experiment technical uncertainties with the robot. For these issues I presented an uncommon participation reward and invitation scheme, measures to avoid confounding direct human-human interactions and ways of "fool proofing" the execution of the experiment to minimize errors in conducting the experiment.

The presented setup shows possible steps towards a setup where the experimenter is aware of the experimental condition but the experiment execution becomes so well-defined and automated that the experiment becomes effectively a double-blind setup because the experimenter has little to no possibility to influence the experiment. This may however pose challenges to the external validity of the study due to added limitations caused by controlling the setup beyond externally realistic contexts.

As a development approach for designing the experiment I used the wayfaring approach to iteratively design and improve the study design. I present the design journey and explain why using this approach is useful for designing explorative studies and for developing the automated interactions with the participants and why, on the other hand, it has limitations for the design of very controlled confirmatory study designs.

The results answering the second research question can be seen as inspiration for researchers in development methodology research for their study designs by studying the lessons learned and taking over useful ideas.

Bibliography

Abad, Zahra Shakeri Hossein and Guenther Ruhe (Aug. 2015). "Using real options to manage Technical Debt in Requirements Engineering". In: *2015 IEEE 23rd International Requirements Engineering Conference (RE).* 2015 IEEE 23rd International Requirements Engineering Conference (RE). ISSN: 2332-6441, pp. 230–235. DOI: 10.1109/RE.2015.7320428.

Arkes, Hal R and Catherine Blumer (1st Feb. 1985). "The psychology of sunk cost". In: *Organizational Behavior and Human Decision Processes* 35.1, pp. 124–140. ISSN: 0749-5978. DOI: 10.1016/0749-5978(85)90049-4.

Baldwin, Carliss Young and Kim B. Clark (2000). *Design Rules: The power of modularity.* MIT Press. 508 pp. ISBN: 978-0-262-02466-2.

Blessing, Lucienne TM and Amaresh Chakrabarti (2009). *DRM: A design reseach methodology.* Springer.

Blindheim, Jørgen (2019). "On Solid-State Deposition of Metal Structures: Conceptualization of a New Additive Manufacturing Method based on Hybrid Metal Extrusion & Bonding". doctoral thesis. Trondheim, Norway: Norwegian University of Science and Technology.

Boehm, Barry (2000). "Requirements that handle IKIWISI, COTS, and rapid change". In: *Computer* 33.7, pp. 99–102.

Brede, Jostein Rødseth et al. (2019). "Resuscitative endovascular balloon occlusion of the aorta (REBOA) in non-traumatic out-of-hospital cardiac arrest: evaluation of an educational programme". In: *BMJ open* 9.5, e027980.

Burks, Arthur W (1946). "Peirce's theory of abduction". In: *Philosophy of science* 13.4, pp. 301–306.

Cetina, Karin Knorr (2009). *Epistemic cultures: How the sciences make knowledge.* Harvard University Press.

Cohen, Jacob (2nd July 2016). “A Coefficient of Agreement for Nominal Scales:” in: *Educational and Psychological Measurement.* DOI: `10.1177/001316446002000104`.

Cooper, Robert G. (May 1990). “Stage-gate systems: A new tool for managing new products”. In: *Business Horizons* 33.3, pp. 44–54. ISSN: 00076813. DOI: `10.1016/0007-6813(90)90040-I`.

Craig, Adam et al. (June 2015). *See. Feel. Trust Your Autonomous Car.* Stanford University, p. 167.

Creswell, John W. and Vicki L. Plano Clark (2007). *Designing and Conducting Mixed Methods Research.* 1st ed. THousand Oaks, California: SAGE Publications. 275 pp. ISBN: 1-4129-2791-9.

Edelman, Jonathan et al. (2012). “Understanding radical breaks”. In: *Design Thinking Research.* Springer, pp. 31–51.

Eisenhardt, Kathleen M. (1st Oct. 1989). “Building Theories from Case Study Research”. In: *The Academy of Management Review* 14.4, pp. 532–550. ISSN: 0363-7425. DOI: `10.2307/258557`.

Eppinger, Steven and Karl Ulrich (2011). *Product Design and Development.* 5th. McGraw-Hill Education. 432 pp. ISBN: 978-0-07-340477-6.

Feyerabend, Paul (1993). *Against method.* 3rd ed. Verso.

Fox, Jesse, Dylan Arena and Jeremy N Bailenson (2009). “Virtual reality: A survival guide for the social scientist”. In: *Journal of Media Psychology* 21.3, pp. 95–113.

Herstatt, Cornelius, Christoph Stockstrom et al. (1st Mar. 2006). “”fuzzy front end” practices in innovating japanese companies”. In: *International Journal of Innovation and Technology Management* 03.1, pp. 43–60. ISSN: 0219-8770. DOI: `10.1142/S0219877006000703`.

Herstatt, Cornelius and Birgit Verworn (2004). “The ‘Fuzzy Front End’ of Innovation”. In: *Bringing Technology and Innovation into the Boardroom: Strategy, Innovation and Competences for Business Value.* London: Palgrave Macmillan UK, pp. 347–372. ISBN: 978-0-230-51277-1. DOI: `10.1057/9780230512771_16`.

Ingold, Tim (2008). *Bringing Things to Life: Creative Entanglements in a World of Materials (Part 5).* URL: `https://www.youtube.com/watch?v=VYT-7_qlURw`.

— (2016). *Lines: a brief history.* Routledge.

Jensen, Matilde Bisballe, Christer Westum Elverum and Martin Steinert (2017). “Eliciting unknown unknowns with prototypes: Introducing prototrials and prototrial-driven cultures”. In: *1-31.* Accepted: 2018-03-

22T08:56:58Z Publisher: Elsevier. ISSN: 0142-694X. DOI: 10.1016/j.destud.2016.12.002.

Jung, Malte Friedrich (Aug. 2011). "Engineering Team Performance and Emotion: Affective Interaction Dynamics as Indicators of Design Team Performance". Dissertation. Department of Mechanical Engineering at Stanford University.

Kahnemann, Daniel (2011). *Thinking fast and slow.*

Khaitan, Siddhartha Kumar and James D. McCalley (June 2015). "Design Techniques and Applications of Cyberphysical Systems: A Survey". In: *IEEE Systems Journal* 9.2, pp. 350–365. ISSN: 2373-7816. DOI: 10.1109/JSYST.2014.2322503.

Kriesi, Carlo (2018). "Wayfaring in the Biomedical Sector: A Call for Re-Introducing the Toolmaker". doctoral thesis. Trondheim, Norway: Norwegian University of Science and Technology.

Kriesi, Carlo, Jørgen Blindheim et al. (1st Jan. 2016). "Creating Dynamic Requirements through Iteratively Prototyping Critical Functionalities". In: *Procedia CIRP*. 26th CIRP Design Conference 50, pp. 790–795. ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.04.122.

Kriesi, Carlo, Martin Steinert, Laura Aalto-Setaelae et al. (2015). "Distributed Experiments in Design Sciences, a Next Step in Design Observation Studies?" The Design Society - a worldwide community. In: ISBN: 9781904670650 ISSN: 2220-4334 Library Catalog: www.designsociety.org Pages: 319-328.

Kriesi, Carlo, Martin Steinert, Anastasios Marmaras et al. (2019). "Integrated Flow Chamber Device for Live Cell Microscopy". In: *Frontiers in bioengineering and biotechnology* 7, p. 91.

Leikanger, Kittil Kittilsen, Stephanie Balters, Martin Steinert et al. (2016). "Introducing the wayfaring approach for the development of human experiments in interaction design and engineering design science". In: *DS 84: Proceedings of the DESIGN 2016 14th International Design Conference*, pp. 1751–1762.

Lipset, Seymour Martin, Martin A Trow and James Samuel Coleman (1956). *Union democracy: The internal politics of the International Typographical Union.* Vol. 296. Free Press.

McCambridge, Jim, John Witton and Diana R. Elbourne (1st Mar. 2014). "Systematic review of the Hawthorne effect: New concepts are needed to study research participation effects". In: *Journal of Clinical Epidemiology* 67.3, pp. 267–277. ISSN: 0895-4356. DOI: 10.1016/j.jclinepi.2013.08.015.

Misty West - about webpage (2019). URL: `https://www.mistywest.com/about/`.

Mynatt, Clifford R., Michael E. Doherty and Ryan D. Tweney (1st Feb. 1977). "Confirmation Bias in a Simulated Research Environment: An Experimental Study of Scientific Inference". In: *Quarterly Journal of Experimental Psychology* 29.1, pp. 85–95. ISSN: 0033-555X. DOI: `10.1080/00335557743000053`.

Nehmzow, Ulrich and Keith Walker (Dec. 2003). "The Behaviour of a Mobile Robot Is Chaotic". In: *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour* 1.4, pp. 373–388.

Niedziela, Kornel et al. (Dec. 2014). *Augmenting the Passenger-Car Team Dynamic using Ambient Communication.* Palo Alto: ME310 course at Stanford University, p. 42.

Nosek, Brian A et al. (2018). "The preregistration revolution". In: *Proceedings of the National Academy of Sciences* 115.11, pp. 2600–2606.

Pahl, Gerhard et al. (6th Aug. 2007). *Engineering Design: A Systematic Approach.* Springer Science & Business Media. 629 pp. ISBN: 978-1-84628-319-2.

Parsons, H. M. (8th Mar. 1974). "What Happened at Hawthorne?: New evidence suggests the Hawthorne effect resulted from operant reinforcement contingencies". In: *Science* 183.4128, pp. 922–932. ISSN: 0036-8075, 1095-9203. DOI: `10.1126/science.183.4128.922`.

Pidić, Almir et al. (2018). "Low-Cost Autonomous Underwater Vehicle (AUV) for Inspection of Water-Filled Tunnels During Operation". In: *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* American Society of Mechanical Engineers Digital Collection.

Pisarczyk, Joanna (June 2015). "Peer Production in physical world: a look at today's community-based infrastructures". Graduation research project. Leiden University Media Technology Department.

Popper, Karl (1935). *Logik der Forschung.* Springer.

Rook, Paul (1986). "Controlling software projects". In: *Software engineering journal* 1.1, pp. 7–16.

Royce, Winston W (1987). "Managing the development of large software systems: concepts and techniques". In: *Proceedings of the 9th international conference on Software Engineering*, pp. 328–338.

Sanchez, Ron and Joseph T. Mahoney (1996). "Modularity, flexibility, and knowledge management in product and organization design". In: *Strategic Management Journal* 17 (S2), pp. 63–76. ISSN: 1097-0266. DOI: `10.1002/smj.4250171107`.

Schrage, Michael (1993). “The Culture(s) of Prototyping”. In: *Design Management Journal* 4.1, pp. 55–65.

— (1999). *Serious play: How the world’s best companies simulate to innovate.* Harvard Business School Press.

Sjöman, Heikki, Juuso Autiosalo et al. (2018). “Using Low-Cost Sensors to Develop a High Precision Lifting Controller Device for an Overhead Crane—Insights and Hypotheses from Prototyping a Heavy Industrial Internet Project”. In: *Sensors* 18.10, p. 3328.

Sjöman, Heikki, Jani Kalasniemi et al. (2018). “The Development of 1Balance: A Connected Medical Device for Measuring Human Balance”. In: *Technologies* 6.2, p. 53.

Sjöman, Heikki, Nazare Soares et al. (2018). “The Breathing Room: Breathing Interval and Heart Rate Capturing through Ultra Low Power Radar”. In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–4.

Snowden, David J and Mary E Boone (2007). “A Leader’s Framework for Decision Making”. In: *Harvard Business Review* 85 (November), pp. 1–9.

Srikanth, Kannan and Phanish Puranam (2011). “Integrating distributed work: comparing task design, communication, and tacit coordination mechanisms”. In: *Strategic Management Journal* 32.8, pp. 849–875.

Steinert, Martin (2020). *Course - Fuzzy Front End - TMM4245 - NTNU.* URL: `https://www.ntnu.edu/studies/courses/TMM4245#tab=omEmnet` (visited on 23/03/2020).

Steinert, Martin and Larry J Leifer (2012). “’Finding One’s Way’: Re-Discovering a Hunter-Gatherer Model based on Wayfaring”. In: *International Journal of Engineering Education* 28.2, p. 251.

Sullivan, Ian, Alexander DeHaven and David Mellor (2019). “Open and reproducible research on open science framework”. In: *Current Protocols Essential Laboratory Techniques* 18.1, e32.

Thomas, Richard C., Amela Karahasanovic and Gregor E. Kennedy (2005). “An Investigation into Keystroke Latency Metrics As an Indicator of Programming Performance”. In: *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42.* ACE ’05. eventplace: Newcastle, New South Wales, Australia. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., pp. 127–134. ISBN: 978-1-920682-24-8.

Türtscher, Philipp (21st Nov. 2008). “The Emergence of Architecture in Modular Systems: Coordination across Boundaries at ATLAS, CERN”. Dissertation. University of St. Gallen.

Türtscher, Philipp, Raghu Garud and Arun Kumaraswamy (1st Dec. 2014). "Justification and Interlaced Knowledge at ATLAS, CERN". In: *Organization Science* 25.6, pp. 1579–1608. ISSN: 1047-7039. DOI: 10.1287/orsc.2013.0894.

Tuyttens, F. A. M. et al. (1st Apr. 2014). "Observer bias in animal behaviour research: can we believe what we score, if we score what we believe?" In: *Animal Behaviour* 90, pp. 273–280. ISSN: 0003-3472. DOI: 10.1016/j.anbehav.2014.02.007.

VDI (1997). *VDI 2222 Methodic development of solution principles - original title in German: Konstruktionsmethodik - Methodisches Entwickeln von Lösungsprinzipien.*

Wolf, Alan et al. (July 1985). "Determining Lyapunov exponents from a time series". In: *Physica D: Nonlinear Phenomena* 16.3, pp. 285–317. ISSN: 01672789. DOI: 10.1016/0167-2789(85)90011-9.

Yin, Robert K (2017). *Case Study Research and Applications: Design and Methods.* SAGE Publications.

Appendices

Appendix A

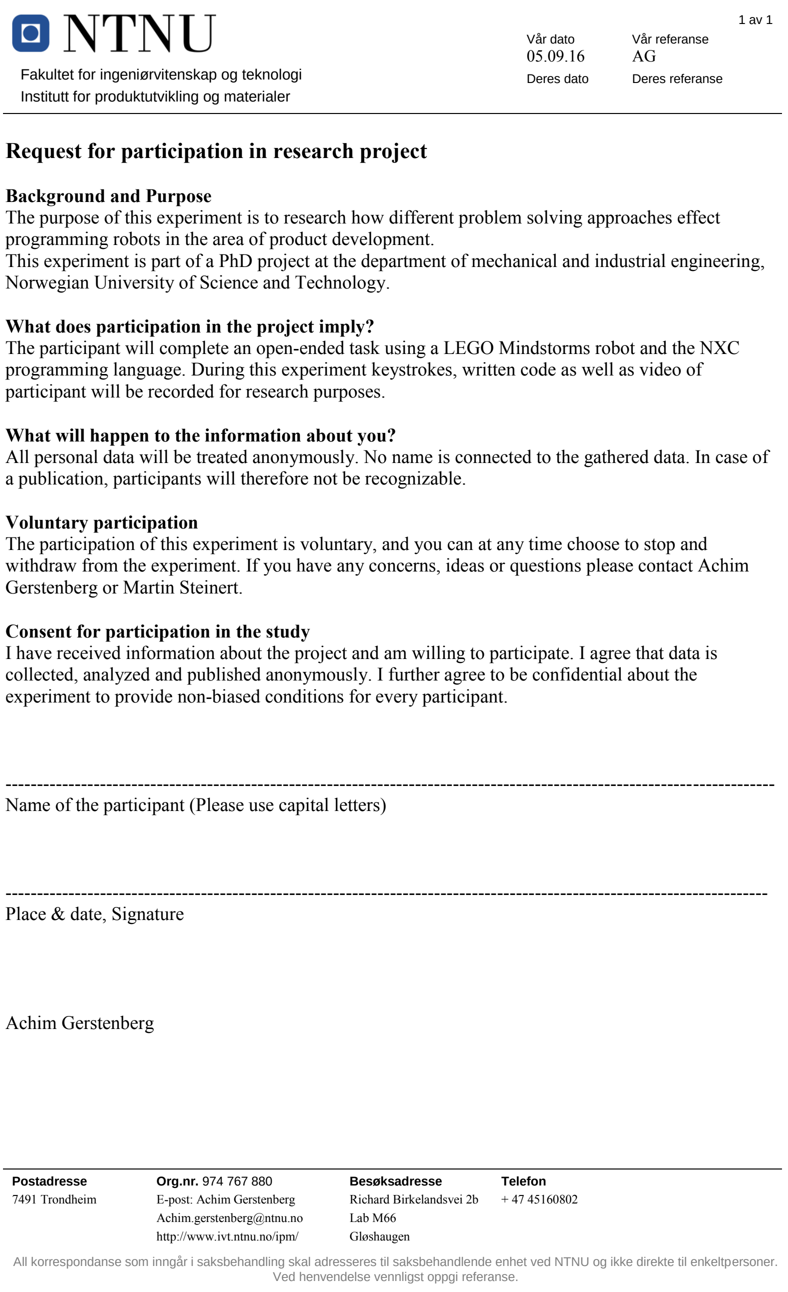
Experimental setup information

This chapter in the appendix provides additional resources used while conducting the experiment. The codes used for evaluating the solutions with the simulation and analysing the data can only be found digitally under the links mentioned below. All this information and more can be found digitally on Github. There are three relevant repositories:

- Content for running the experiment:
  `https://github.com/AchimGerstenberg/RoboExpSetup`
- Content for evaluating the solutions using the simulation:
  `https://github.com/AchimGerstenberg/RoboExpSimulation`
- Python script for data analysis:
  `https://github.com/AchimGerstenberg/RoboExpDataAnalysis`

A.1 Printed templates

Consent Form



1 av 1

NTNU

Fakultet for ingeniørvitenskap og teknologi
Institutt for produktutvikling og materialer

Vår dato	Vår referanse
05.09.16	AG
Deres dato	Deres referanse

Request for participation in research project

Background and Purpose
The purpose of this experiment is to research how different problem solving approaches effect programming robots in the area of product development.
This experiment is part of a PhD project at the department of mechanical and industrial engineering, Norwegian University of Science and Technology.

What does participation in the project imply?
The participant will complete an open-ended task using a LEGO Mindstorms robot and the NXC programming language. During this experiment keystrokes, written code as well as video of participant will be recorded for research purposes.

What will happen to the information about you?
All personal data will be treated anonymously. No name is connected to the gathered data. In case of a publication, participants will therefore not be recognizable.

Voluntary participation
The participation of this experiment is voluntary, and you can at any time choose to stop and withdraw from the experiment. If you have any concerns, ideas or questions please contact Achim Gerstenberg or Martin Steinert.

Consent for participation in the study
I have received information about the project and am willing to participate. I agree that data is collected, analyzed and published anonymously. I further agree to be confidential about the experiment to provide non-biased conditions for every participant.

Name of the participant (Please use capital letters)

Place & date, Signature

Achim Gerstenberg

Postadresse	**Org.nr.** 974 767 880	**Besøksadresse**	**Telefon**
7491 Trondheim	E-post: Achim Gerstenberg	Richard Birkelandsvei 2b	+ 47 45160802
	Achim.gerstenberg@ntnu.no	Lab M66	
	http://www.ivt.ntnu.no/ipm/	Gløshaugen	

All korrespondanse som inngår i saksbehandling skal adresseres til saksbehandlende enhet ved NTNU og ikke direkte til enkeltpersoner.
Ved henvendelse vennligst oppgi referanse.

Checklist

PRE-EXPERIMENT CHECKLIST participant number:

Print Documents (this checklist and evaluation form, consent form, CTest, questionnaires, expl. of functions, datasheet, task description
Write the participant number on all prints
Create/open a participant folder on the usb stick
Update the info.txt file on the SD card

Turn on mock-pit lights
Turn on ceiling lights (not the spots)
Put your phone in flight mode and silent
Empty floor from experiment equipment
Put in new batteries into the LEGO robot
Set the “shut down” time of the robot to “never”
Connect and reset the robot
Prepare programming windows
Load the “checkcompile” program onto the robot and have it open on screen 1
Set keyboard language to EN
Make sure that checkcompile is visible on screen 1
Turn off screen 1
Load presentation with instructions on 2nd screen

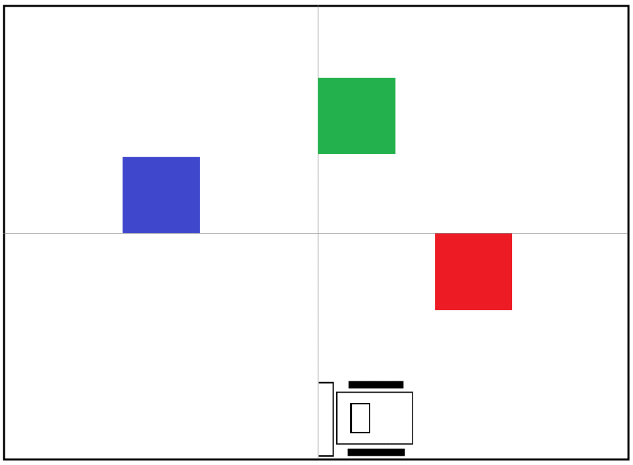
Put the prepared SD card into the keylogger
Load keylogger program onto Arduino, open serial monitor, check if it works
Synchronize stopwatch, programming computer, keylogger, and video rec. computer
Make voice instructions available (open folder)
Set volume to max

Get fresh drinking water
Put up “experiment in progress” sign on the door
Close off the experimentor’s space with cardboard walls

After participant has signed the consent form
Start screen capture
Start video recording

Evaluation protocol

Evaluation 1 participant number:

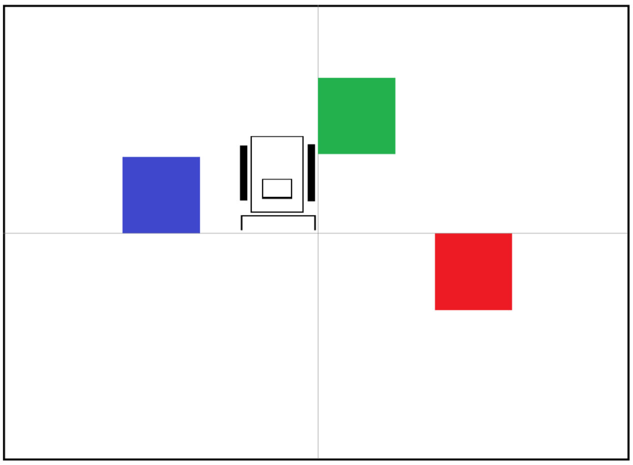


Code compiled	Robot fell off cardboard	
Robot moved	Robot stopped	
Robot touched a cube	Program manually aborted	
1st cube removed Time:	t bonus	t penalty
2nd cube removed Time:	t bonus	t penalty
3rd cube removed Time:	t bonus	t penalty

Overall time including bonus and penalties: ___________________________

Notes:

Evaluation 2 participant number:

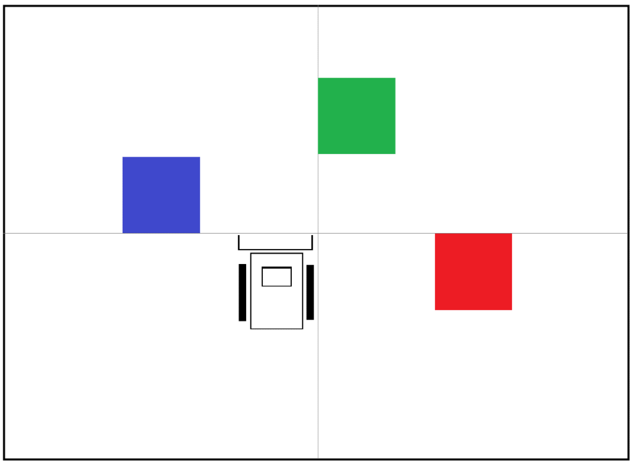


Code compiled	Robot fell off cardboard	
Robot moved	Robot stopped	
Robot touched a cube	Program manually aborted	
1st cube removed Time:	t bonus	t penalty
2nd cube removed Time:	t bonus	t penalty
3rd cube removed Time:	t bonus	t penalty

Overall time including bonus and penalties: ___________________________

Notes:

Evaluation 3 participant number:

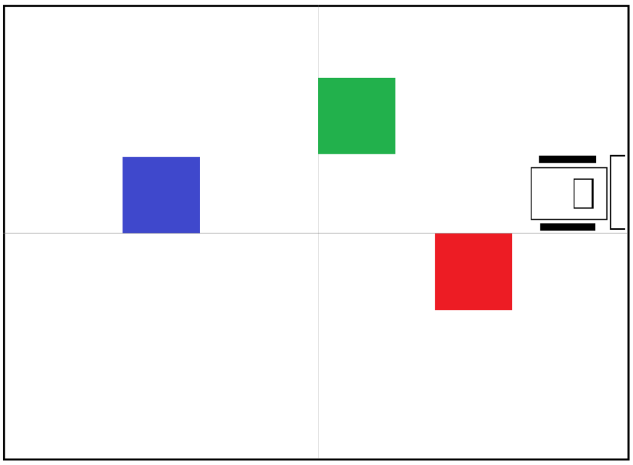


Code compiled	Robot fell off cardboard	
Robot moved	Robot stopped	
Robot touched a cube	Program manually aborted	
1st cube removed Time:	t bonus	t penalty
2nd cube removed Time:	t bonus	t penalty
3rd cube removed Time:	t bonus	t penalty

Overall time including bonus and penalties: ___________________________

Notes:

Evaluation 4 participant number:

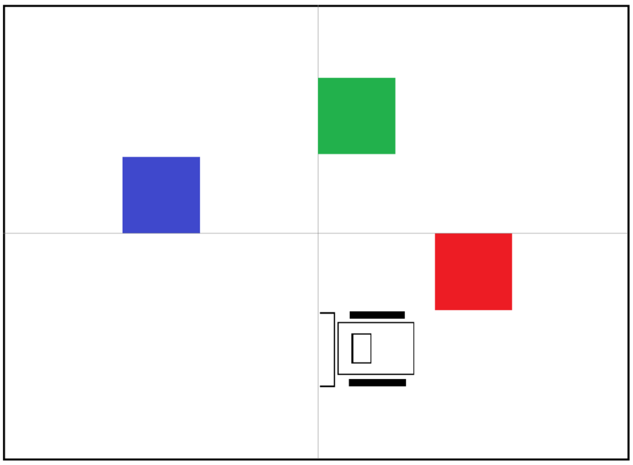


Code compiled	Robot fell off cardboard	
Robot moved	Robot stopped	
Robot touched a cube	Program manually aborted	
1st cube removed Time:	t bonus	t penalty
2nd cube removed Time:	t bonus	t penalty
3rd cube removed Time:	t bonus	t penalty

Overall time including bonus and penalties: ___________________________

Notes:

Evaluation 5 participant number:



Code compiled	Robot fell off cardboard	
Robot moved	Robot stopped	
Robot touched a cube	Program manually aborted	
1st cube removed Time:	t bonus	t penalty
2nd cube removed Time:	t bonus	t penalty
3rd cube removed Time:	t bonus	t penalty

Overall time including bonus and penalties: ___________________________

Notes:

Questionnaires

Programming experience questionnaire participant number:

How would you grade your programming knowledge in C or C++:
(0 = no experience, 6 = professional (for money) programmer)

0 1 2 3 4 5 6

How would you grade your programming knowledge in programming Arduino:
(0 = no experience, 6 = professional (for money) programmer)

0 1 2 3 4 5 6

How would you grade your programming knowledge in programming LEGO Mindstorms:
(0 = no experience, 6 = professional (for money) programmer)

0 1 2 3 4 5 6

Please give a ranking of all the programming languages and how much experience you have in each. (0 = no experience, 6 = professional (for money) programmer)

1.	________________________	0	1	2	3	4	5	6
2.	________________________	0	1	2	3	4	5	6
3.	________________________	0	1	2	3	4	5	6
4.	________________________	0	1	2	3	4	5	6
5.	________________________	0	1	2	3	4	5	6
6.	________________________	0	1	2	3	4	5	6

Post experiment questionnaire participant number:

Thank you for participating in our experiment!
This really helps us in doing research about learning behavior in product development.

This experiment is anonymous but before you leave, we would like to know some things from you:

Specify your gender:

Which year were you born:

What is your study program:

Which semester are you in in your current field of study:

How pleased are you with the result of your programming?
(0 = not at all, 6 = very pleased)

0 1 2 3 4 5 6

How much did you enjoy this experiment? (0 = not at all, 6 = it was very pleasurable)

0 1 2 3 4 5 6

Do you have a study colleague in mind that you will recommend to participate in this experiment? Yes No

Will you actually recommend it to him or her? Yes No

One final (but very important) remark:
Please do not tell anyone about the content of this experiment.

Pre-experiment programming test

```
// What is the value of x when the program ends?

int x = 0;
bool y = false;

int function(int parameter1, int parameter2)
{
  return parameter1 + parameter2;
}


while(x <= 5)
{
  if(x >= 2 && y)
  {
    y = false;
    x = function(x,1);
  }
  else
  {
    x = x + 2;
    y = true;
  }
}

// END OF THE PROGRAM


// What is the value of x?
// Your answer:



// you can use this table to help remember the values

//   x   |   y   //
//---------------//
//   0   | false //
//       |       //
//---------------//
//       |       //
//       |       //
//---------------//
//       |       //
//       |       //
//---------------//
//       |       //
//       |       //
//---------------//
//       |       //
//       |       //
//---------------//
//       |       //
//       |       //
//---------------//
```

Voice instruction content

In this appendix section I list the manually triggered voice instructions and when they are used. The computer generated voices that accompany the instructions on the instruction screen correspond to the text that is displayed in the power point slides. The slides used on the instruction screen can be found on Github:
`https://github.com/AchimGerstenberg/RoboExpSetup/tree/master/instructions`

Welcome, this experiment is conducted without direct personal interactions. You will be guided through the experiment by prerecorded voice and video as well as text instructions.
Triggered when the participant enters the experiment room for the first time.

Please close the door.
Triggered if the participant enters the programming booth without closing the door to the experiment room.

Please go to the programming booth.
Triggered if the participant does not enter the programming booth after entering the experiment room and awaits further instructions.

You have received a new instruction on the instruction screen.
Triggered if the participant has received a new instruction on the instruction screen but has not noticed it.

Please do not forget to load your code onto the robot and test it regurarly.
Triggered before evaluation 1 for participants in the testing condition that have not tested within the last 5 minutes. "Regurarly" was misspelled on purpose to make the computer generated voice more easily understandable.

We recommend to test your code soon.
Triggered one minute after the previous test reminder if the participant had not yet tested recently.

Please connect the robot.
Triggered if the robot was not connected to the programming PC and the experimenter needed to load a code onto the robot. This usually happened before evaluations.

Do you want help with the syntax?
Triggered after the participant tries to compile a code that includes a syntax error. If the participant answers "yes" one of the following voice instructions is used:
A semicolon is missing. Watch the mouse cursor for a hint.

A bracket is missing. Watch the mouse cursor for a hint.

There is a spelling mistake. Watch the mouse cursor for a hint.

Watch the mouse cursor for a hint.
Triggered if none of the above applies.

The robot is placed incorrectly.
Triggered during an evaluation if the participant has not placed the robot correctly at the assigned position and orientation.

Are you ready to start the robot?
Triggered when the robot is placed correctly and the experimenter is ready to document the robot performance.

Start the robot in three, two, one, now.
Used to synchronize the time keeping by the experimenter with starting the robot's code execution by the participant.

I can only answer to questions with yes or no.
Triggered if the participant asks a question that cannot be answered with yes and no. Yes and No questions are answered with yes and no by a computer generated voice.

Please wait a moment and do nothing.
Used when the experimenter needs additional time. Usually needed if something unexpected occurred.

You can now carry on.
Used when the problems have been resolved and the participant can continue with the experiment.

Explanation of programming library functionality

```
// Explanation of functions


// Table of contents:

// OUTPUTS:

   dispNum(x,y, value);

   dispText(x,y,"example text" );

   PlayTone(frequency, duration);

// TIMING & RANDOMNESS

   startTimer1();

   readTimer1();

   wait(time);

   random(minimum, maximum);

// SENSORS

   reflectionDown();

   reflectionRedLeft();

   ultrasound();

   blink        // is an int variable

// MOVEMENT

   motor(speed_left_belt, speed_right_belt);

   turn(speed, degrees);


// EXAMPLE SYNTAX CODE
// Output for FEEDBACK

void dispNum(int x, int y, int value)
{
 // displays a numerical value on the screen of the robot at the
 // screen coordinates x (horizontal) and y (vertical).
 // In x the display is 90 pixels wide and in y it has 8 lines that can be adressed
 // with LCD_LINE1, LCD_LINE2, LCD_LINE3, ..., LCD_LINE8.
}
 //example:
 dispNum(0,LCD_LINE1, 42);



void dispText(int x, int y, "example text" )
{
 // displays text on the screen of the robot at the
 // screen coordinates x (horizontal) and y (vertical).
 // In x the display is 90 pixels wide and in y it has 8 lines that can be adressed
 // with LCD_LINE1, LCD_LINE2, LCD_LINE3, ..., LCD_LINE8.
}
 //example:
 dispText(0,LCD_LINE1, "Hello World" );



void PlayTone(int frequency, int duration)
{
 // plays a tone with the frequency specified in variable "frequency" in Hz
 // for the time specified in "duration" in ms.
}
 //example:
 PlayTone(440,200);
```

```
// TIMING and RANDOMNESS

void startTimer1()
{
 // starts or restarts a timer when this function is called.
 // start_timer2() and start_timer3() work analogously.
}
 //example:
 startTimer1();



unsigned long int readTimer1()
{
 // returns the time in milliseconds that has passed after starting the timer.
 // readTimer2() and readTimer3() work analogously.
}
 //example:
 int time = readTimer1();



void wait(int time)
{
 // pauses the execution of the program for the time specified
 // in the parameter "time" in milliseconds.
}
 //example:
 wait(1000);



int random(int minimum, int maximum)
{
 // returns a random number between minimum and maximum
 // if minimum is greater than maximum, it returns 0
}
 //example:
 int x = random(-20,20);
```

```
// SENSORS

int reflectionDown()
{
 // This sensor emits red light downwards and measures how much is reflected
 // It returns values between 0 and 100, where higher numbers mean higher reflectivity
 // of the surface below
}
 //example:
 int refl = reflectionDown();



int reflectionRedLeft()
{
 // The color reflection sensors send out red light and detect how much of
 // the emitted red light is reflected back into the sensor.
 // Red objects reflect more red light than green or blue objects.
 // Smaller values correspond to less reflectivity of red light.
 // This function return a reflection value for the left reflection sensor.
 // A similar function exists for the right reflection sensor and is called by
 // reflectionRedRight()
}
 //example:
 int reflRight = reflectionRedRight();



int ultrasound()
{
 // The ultrasound sensor measures the distance between the sensor and an object
 // that reflects ultrasound by detecting the time of flight of an ultrasound pulse.
 // It returns a number that correlates with the distance in cm.
}
 //example:
 int ultra = ultrasound();


int blink;
// The reflection sensors in the front can also sense a blinking light source.
// This process runs continuously in the background.
// The average value from the last one and a half seconds is stored in the
// global integer variable "blink" that can be used throughout the code.
// example:
   if(blink > 2)
```

```
// MOVEMENT


void motor(int speed_left_wheel, int speed_right_wheel)
{
 // the speeds can be values between -100 (full backwards) and 100 (full forward).
 // Speed of 0 stops the motor.
}
 // example:
 motor(10,-10);



void turn(int speed, int degree)
{
 // Sets one belt to positive speed and the other to negative speed and thereby
 // turns the robot with the motor speed defined in the first parameter.
 // The turning direction and stopping angle is defined in degrees in the
 // variable degree. A positive degree value turns the robot clockwise.
}
 // example:
 turn(20,-90);
```

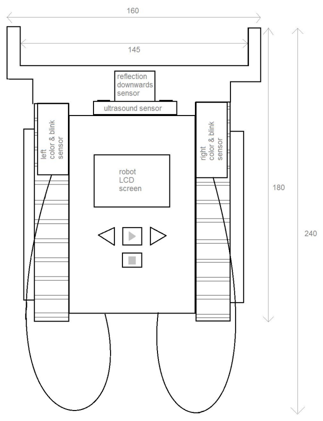
```
// EXAMPLE OF SYNTAX

int x = 7;
int y = 42;

// decleration of a function
int function(int parameter1, int parameter2)
{
 return parameter1 + parameter2;
}


while(true)  // infinite loop
{
 if(x == y || x != 7)    // "==" equal, "||" logical or, "!=" not equal
 {
  x = 42;
 }
 else
 {
  y = 7;
 }

 while(x >= y && x > 7) // "<=" less than, ">=" greater than, "&&" logical and
 {
  x = x - 1;             // you could also write "x--;"
 }
}

/* this is
a multi-line
comment */
```

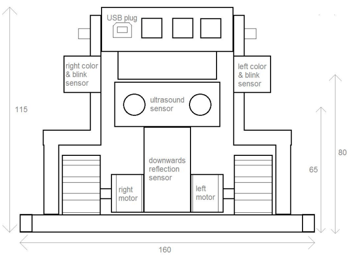
Robot datasheet

Data Sheet

Table of contents:

Dimensions:


160
145
reflection downwards sensor
ultrasound sensor
left color & blink sensor
right color & blink sensor
robot LCD screen
180
240


USB plug
right color & blink sensor
left color & blink sensor
ultrasound sensor
115
downwards reflection sensor
right motor
left motor
80
65
160

Front view

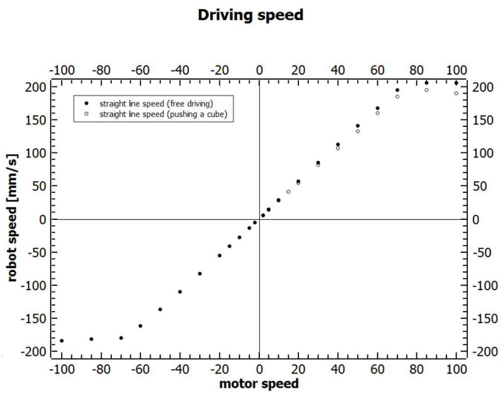
Top view

Weight including batteries: 690 g

Movement:

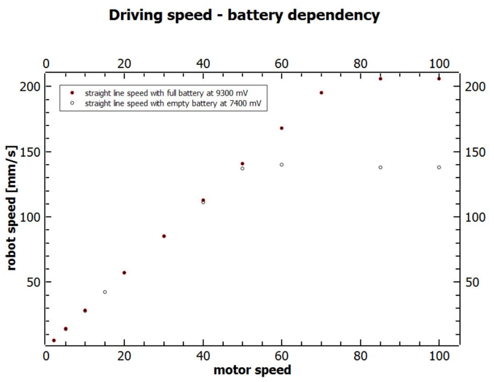
The robot has one rubber belt on each side which move over a front and a back wheel. The front wheel is driven by an electric motor. Included in the motor is a rotation sensor that senses position of the motor shaft. This allows the motor speed to be PID controlled. The motors can be controlled individually of each other and can turn forward and backwards.

Straight Line Speed:



Measured straight line driving speed as a function of programmed motor speed with fully charged batteries.

At motor speeds of 85 and above the robot starts to drift into a left turn.

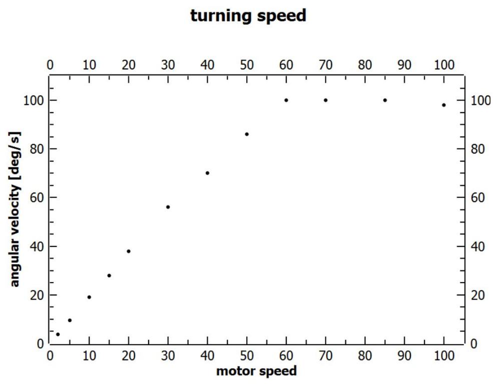


Forward straight line driving speed as a function of motor speed at different battery voltages.

At the lower battery voltage the robot starts a left turn at motor speeds of 60 and above.

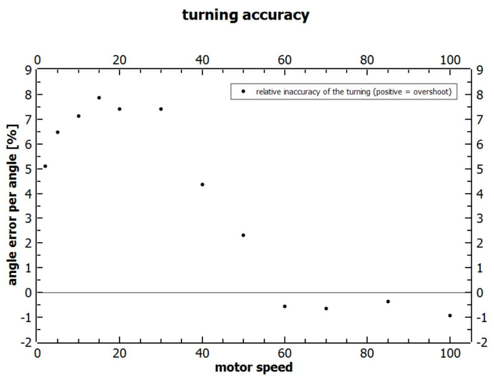
Turning Speed and Accuracy:

Turning the robot by using the motor command:



Turning speed using the motor function with motor(-x,x) where x is the programmed motor speed. This means the robot is turning counterclockwise on the spot with equal absolute motor speeds on both belts in opposite directions.

Accuracy of turning the robot with a programmable speed by a programmable angle using the turn command:



Relative deviation from the desired turning angle. A positive value means "overshooting" the desired turning angle.
A motor speed of i.e. 10 means that the robot turns with motor(-10,10) or motor(10,-10).

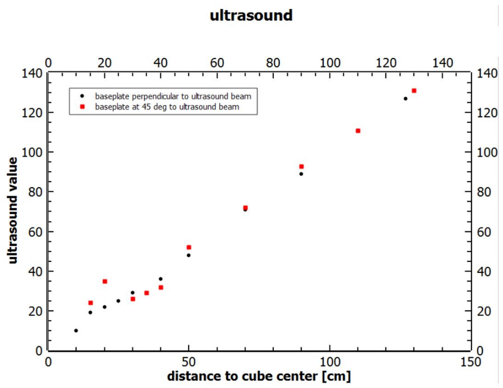
Sensors:

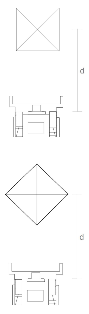
The robot is equipped with 4 sensors.

- Ultrasound distance sensor: points forward and measures the time of flight of a sound signal emitted from the robot and reflected off an object. This time can be converted into a distance measurement.
- Downwards reflection sensor: points downwards and measures the reflectivity of the surface under the robot
- Color and blink sensor: The robot has one of these sensors on each side. They point forward and measure the colored light reflectivity of objects and the light intensity of a blinking light with a frequency of 0,93 Hz.

Ultrasound accuracy:

Ultrasound value as a function of distance from the ultrasound sensor to the center of the cube:



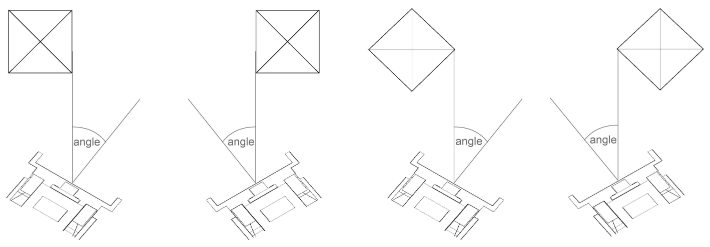


Base plate perpendicular to the sensor beam

Base plate at 45 degrees to the sensor beam

Ultrasound cut-off angles:

The ultrasound distance measurement remains accurate until the object is outside the field of view of the sensor. At the cut-off angle the object is no longer in the field of view of the ultrasound sensor. The cut-off angles are shown below:



Cut-off angle:	Cut-off angle:	Cut-off angle:	Cut-off angle:
35 ± 3 degrees	35 ± 3 degrees	22 ± 2 degrees	26 ± 2 degrees

Downwards reflection sensor:

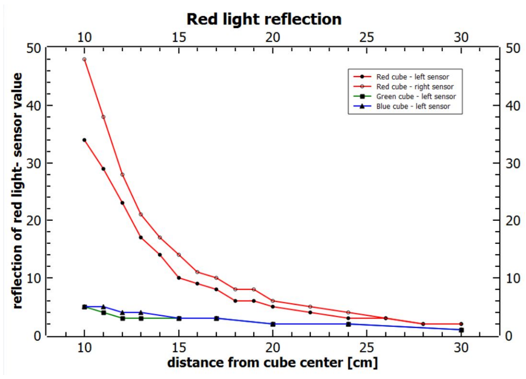
The reflection sensor sends out pulses of red light and detects how much of this light is reflected.

Sensor value on white paper: 47

Sensor value on cardboard: between 36 and 38

Sensor value when overhanging over the cardboard edge: 15

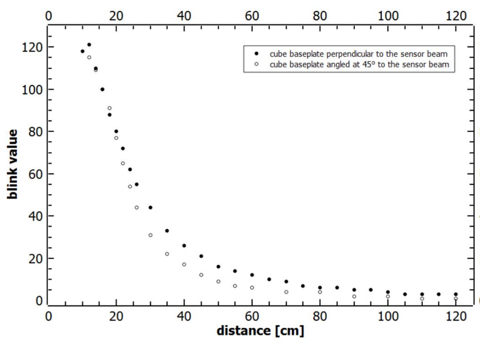
Red light reflection sensor sensitivity:

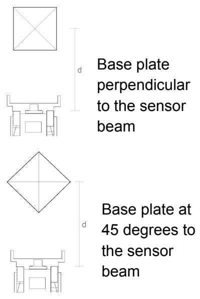


Reflectivity of red light off a red, green and blue cube dependent on the distance between the sensors and the center of the cubes.

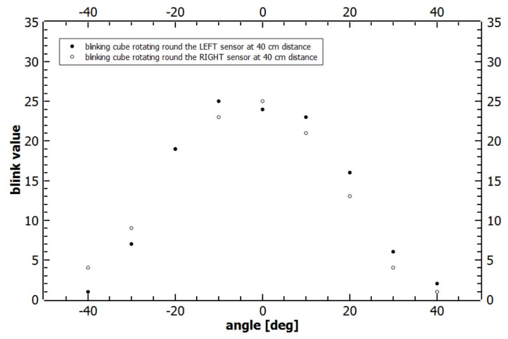
Blink sensor sensitivity:
The blink sensor uses the same sensor as the color sensor except that it is responsive to a blinking light source at 0,93 Hz.
The following values were measured for a blink light placed upside down in the ceiling of a green cube.

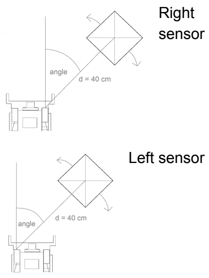
blink - distance dependency



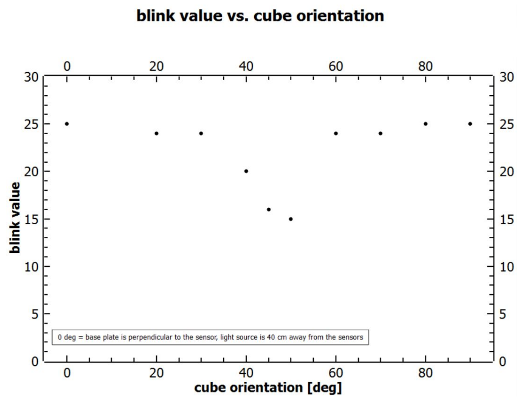

d
Base plate perpendicular to the sensor beam
d
Base plate at 45 degrees to the sensor beam

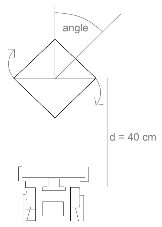
View angle of the blink sensor





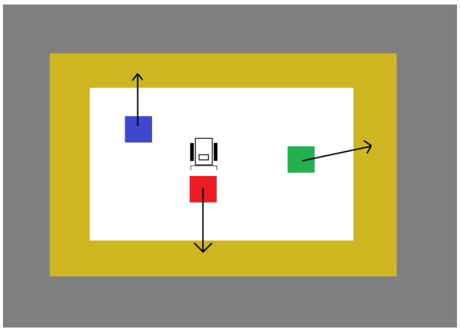
The color sensors have a similar view angle pattern.


blink value vs. cube orientation
blink value
cube orientation [deg]
0 deg = base plate is perpendicular to the sensor, light source is 40 cm away from the sensors


angle
d = 40 cm

Task description

The task is to use the robot to remove the three cube objects from the white area in the shortest time possible.



The robot must stay on the cardboard area.
You may use any of the three blinking light sources and place them wherever you like before executing the program. They fit into the top side of the cube objects. The robot shall not be influenced by you after the program is started.

Objects that are fully outside the white area on the cardboard must be removed from the “playground” by you.

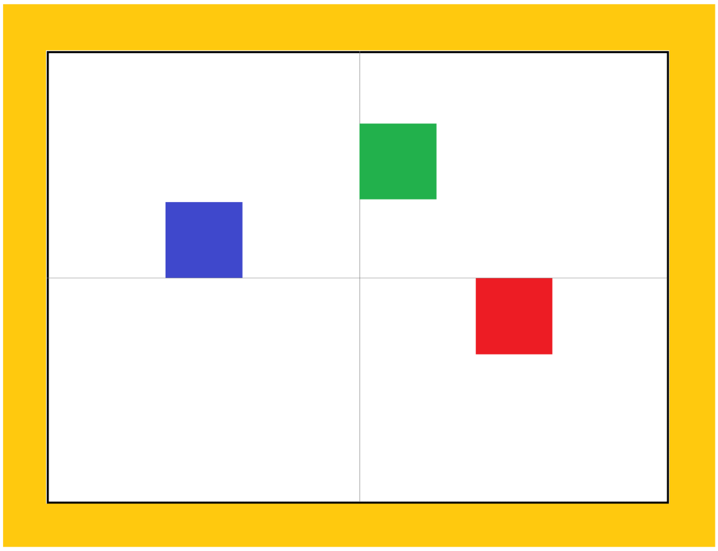
You can earn a 10 second time bonus if you display the correct tone frequency while pushing out a cube. You will get a 10 second time penalty if you play the wrong tone frequency.

Red cube:	400 Hz
Green cube:	800 Hz
Blue cube:	1600 Hz

You are given:

- The LEGO robot
- This task description with the starting setup (backside)
- Explanation of useful functions
- A data sheet about the robot’s behaviour
- Time to complete the task
- Three blinking light sources

This is the setup of the cube objects for each evaluation. The starting position and orientation of the robot will be different each time.



Appendix B

Additional data

This appendix chapter includes data from the robot experiment that was left out in the main part of the thesis because it would make it too long to be easily readable.

B.1 Manually recorded results

In this section I present additional data that was recorded manually by the experimenter while the participant was participating in the experiment.

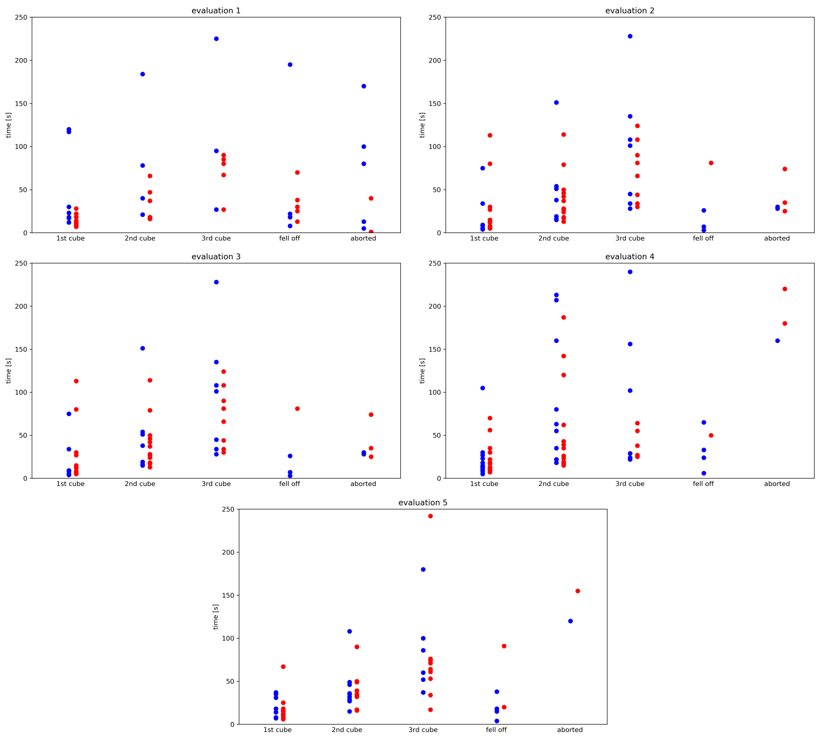


Figure B.1: Cube removal times, blue dots: planners, red dots: testers

B.2 Simulated results

The data shown below results from 99 iterations for each solution from 99 different randomly chosen (but similar for each solution) starting positions. The raw data contains 13365 rows. It is therefore added in the Github repository as an html and a tab separated txt-file under
`https://github.com/AchimGerstenberg/RoboExpDataAnalysis`
the filename "simulation_rawdata".

The labelling of the columns is explained in subsection B.2.1. This raw data is then averaged over the 99 iterations for each solution in every evaluation. The resulting data is published as an html table and a tab separated txt-file in the same Github repository under the file name "simulation_evaldata" and the corresponding labbeling is explained in appendix B.2.2.

B.2.1 Simulation raw data labelling

datetime

Actual time when the simulation of this iteration completed and the result was saved in the file

participant
participant identification number

evaluation

codename
Filename thecode of the solution was initially saved under

StartConfig
Incrementing number for counting up the starting positions from 1 to 99

startX, startY and startPhi
X, Y coordinate and orientation of the robot's starting configuration in the simulation

red, green and blue blink
Set to true if the participant used the blink light beacons inside of the red, green or blue cube.

red, green and bluecube
Removal times of the red, green and blue cube in seconds

bonus
Sum of bonus time for detecting the correct cube color in seconds

penalty
Sum of penalty time for incorrectly detecting the cube color in seconds

lastcube
processed data: time of removing the lastcube in seconds. This is equal to the task completion time. The value is "NaN" if the task was not completed.

overalltime
The time in seconds for completing the task including subtraction of the bonus time and adding the penalty time. The value is "NaN" if the task was not completed.

boxestouched
Counter for how often the robot touched a cube before completing the task. For compatibility reasons the value is ”NaN” if the task was not completed.

edgetouches
Counter for how often the robot reached the edge of the white area on the playground before completing the task. For compatibility reasons the value is ”NaN” if the task was not completed.

distance
The distance the robot travelled before completing the task in arbitrary units. For compatibility reasons the value is ”NaN” if the task was not completed.

removedcubes
processed data: number of removed cubes

fallofftime
The time in seconds when the robot fell off the playground if the robot fell off. Otherwise the value is ”NaN”.

lastmovedtime
The time in seconds when the robot lastly changed its x or y coordinate. This can be indicative if the robot was stuck spinning or stopped.

condition
Experimental condition. The value is either 0 for participants that could not test their solutions before the first evaluation or 1 for participants that could.

first, second and thirdcube
processed data: Time in seconds when the robot removed the first, second and third cube. ”thirdcube” is redundant with ”lastcube”. The values can be ”NaN” if the cubes were not removed before the robot fell off or the simulation timed out.

B.2.2 Averaged simulation data for each evaluation

The averaged data for each evaluation contains the mean values, standard deviations and sample size N as well as the medians for the same column labels as mentioned in the previous subsection. In many cases the sample size is $N = 99$ for all 99 starting configurations and in some cases $N < 99$ if the raw data contains "NaN" values.

Additionally, the table consist of the following column labels:

taskcomplete
Share of runs that complete the task from the 99 starting positions

tests
Number of tests after the previous and before the current evaluation

keystrokes
Number of keystrokes after the previous and before the current evaluation

searchmethod
manually entered data: Method used for detecting the cubes. 1 = ultrasound, 2 = blink, 3 = combination of ultrasound and blink

B.2.3 Influence of testing

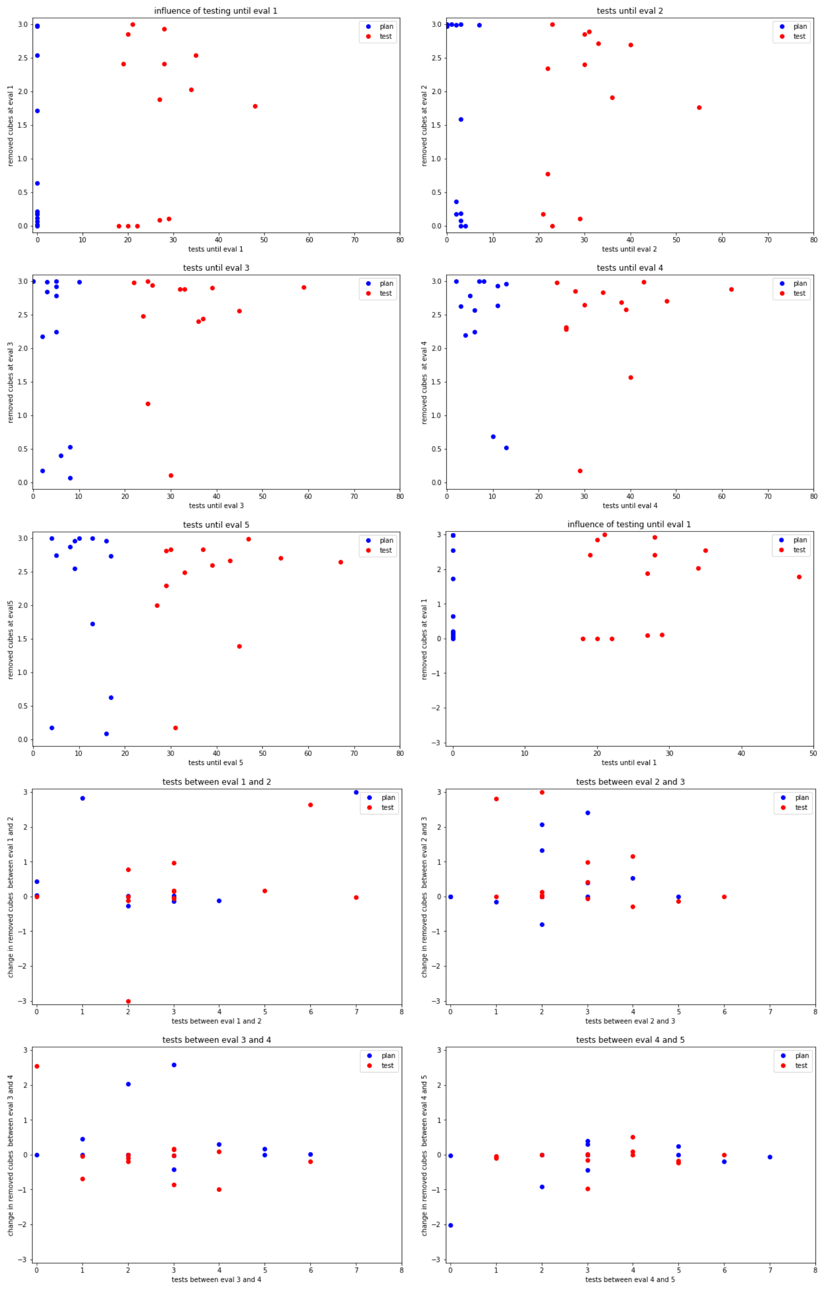


Figure B.2: Influence of total amount of tests on number of removed cubes and number of tests between evaluation on change in number of cubes.

B.2.4 Influence of typing between tests

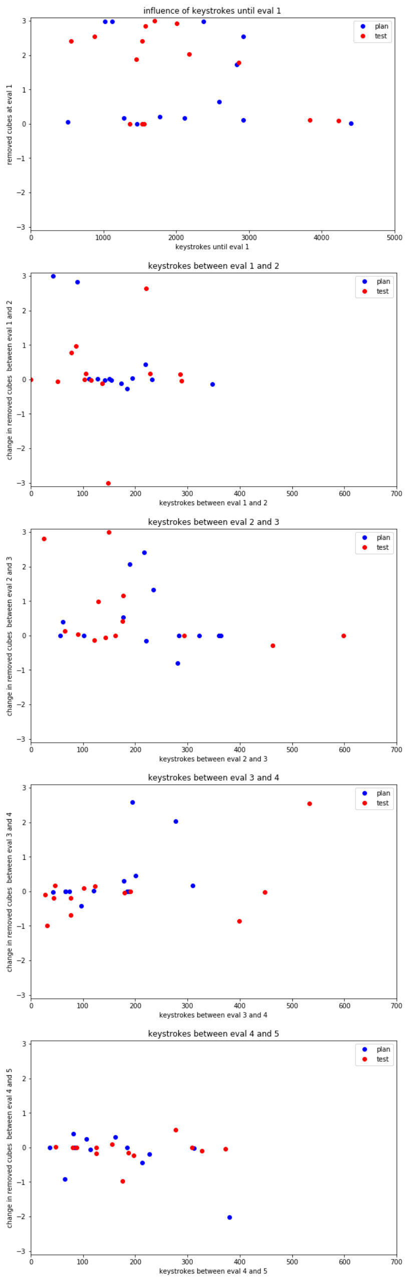


Figure B.3: Influence of amount of keystrokes between evaluations on the change of number of removed cubes in the same time interval.

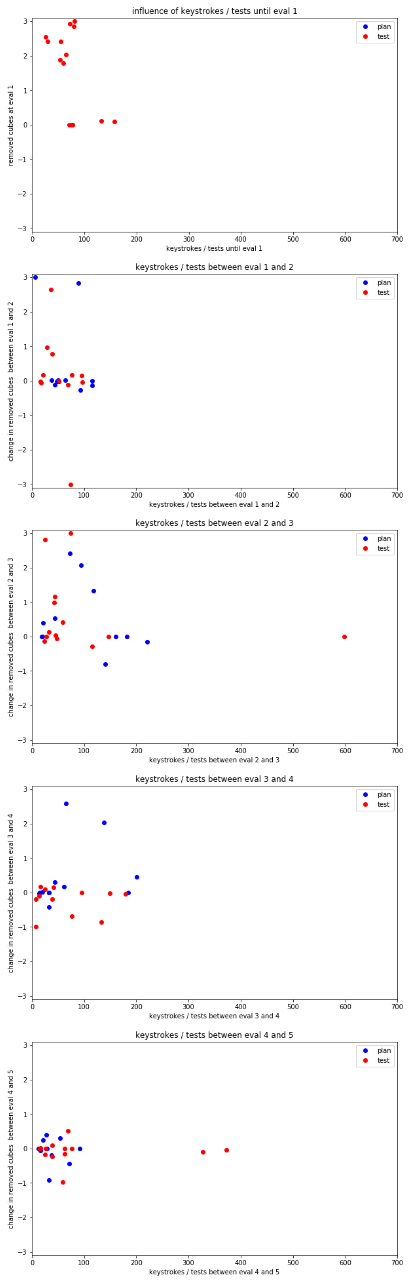
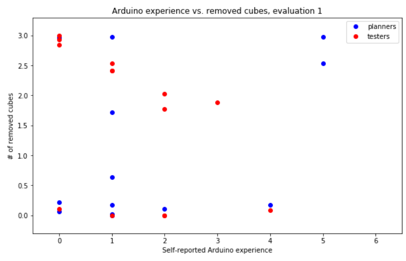


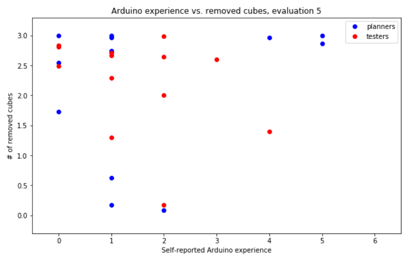
Figure B.4: Influence of amount of keystrokes per test between evaluations on the change of number of removed cubes in the same time interval.

B.2.5 Influence of experience on performance

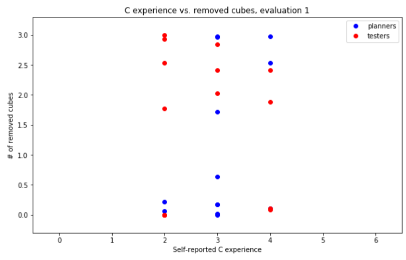
Self-reported experience



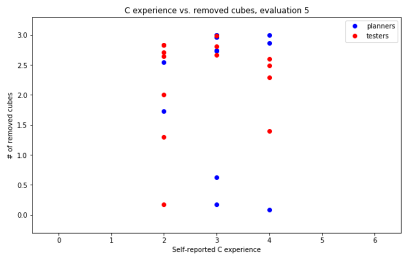
(a) Arduino experience, evaluation 1



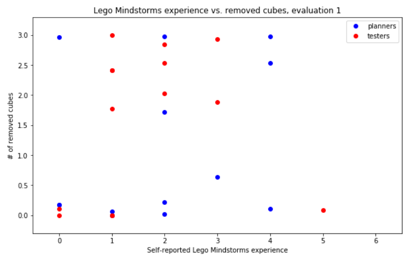
(b) Arduino experience, evaluation 5



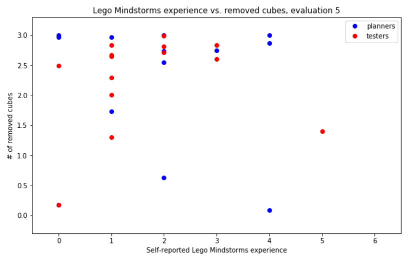
(c) C experience, evaluation 1



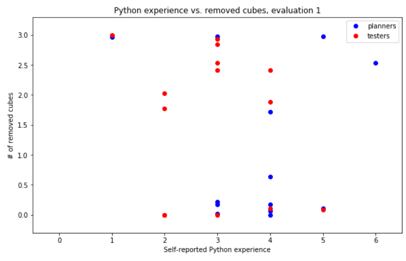
(d) C experience, evaluation 5



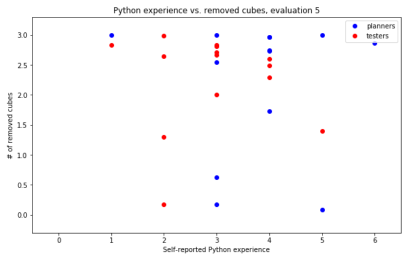
(e) Lego Mindstorms exp, eval 1



(f) Lego Mindstorms exp, eval 5



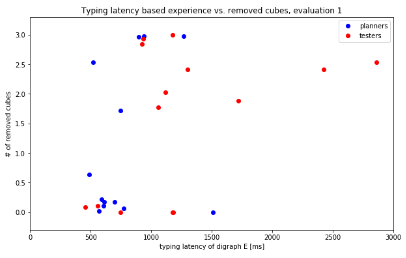
(g) Python experience, evaluation 1



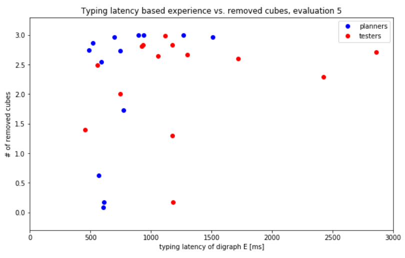
(h) Python experience, evaluation 5

Figure B.5: Influence of self-reported programming skills on the mean number of removed cubes (performance).

Keystroke latency



(a) Keystroke latency, evaluation 1



(b) Keystroke latency, evaluation 5

Figure B.6: Influence of typing latency on the mean number of removed cubes (performance) in evaluation 1 and 5.

Appendix C

Publications included in the PhD work

C.1 Contribution 1: Distributed Experiments in Design Sciences, a Next Step in Design Observation Studies?

C.2 Contribution 2: Bridging Tangible and Virtual Interaction: Rapid Prototyping of a Gaming Idea

Bridging Tangible and Virtual Interaction: Rapid Prototyping of a Gaming Idea

Thov Reime[1], Heikki Sjöman[1], Achim Gerstenberg[1], Pekka Abrahamsson[2], and Martin Steinert[1]

[1] Department of Engineering Design and Materials, NTNU, Richard Birkelands vei 2B, 7492 Trondheim, Norway
{Heikki.Sjoman,Achim.Gerstenberg, Martin.Steinert}@ntnu.no, Thov@stud.ntnu.no
[2] Department of Computer and Information Science, NTNU, Sem Sælands vei 9, 7491 Trondheim, Norway
pekkaa@ntnu.no

Abstract. The Fibo Car is an example for a game interface that allows a user to modify a virtual car in a racing game through assembling tangible car parts. This paper describes the 6 week development journey towards a fully functional proof of concept prototype, reflections on the process as well as the technical details of the prototype.

1 Introduction

The basic idea of the Fibo Car game project is that the player can construct a real world car model out of tangible building blocks. The structure of the model is digitally recognized and it influences the properties of the virtual model in the car racing game.

In this paper we focus solely on the development of the tangible objects, the structure recognition, and a virtual representation without any gameplay. The game idea is related to games like Kerbal Space Program [1] or Besiege [2] except that the constructing takes place tangibly like in LEGO Mindstorms [3].

The solution presented here has one central part that can detect the attached neighboring parts. The identification is realized by measuring and identifying part-specific resistances with an Arduino Uno microcontroller board [4] wired to a PC. A virtual representation of the identified tangible model is then shown on a screen. The latest version of the prototype is shown in figure 1.

The upcoming section describes the development journey of the first 6 weeks. It includes failures, dead ends, and gives reasons for the actions taken. We concentrated on development speed using a process with rapid iteration cycles in favour of fast learning and quick improvement without project control by predefining requirements and a priori budgeting.

© IFIP International Federation for Information Processing 2015
K. Chorianopoulos et al. (Eds.): ICEC 2015, LNCS 9353, pp. 523–528, 2015.
DOI: 10.1007/978-3-319-24589-8_50

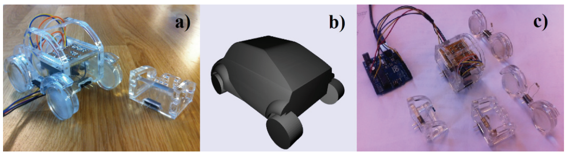


Fig. 1. Latest proof of concept prototype showing the tangible model in a) and the corresponding virtual representation in b). All available tangible car parts of the latest prototype are shown in c).

2 Development Journey

The project started with a presentation of the basic game idea by the problem owner to the developers. The aim was to reach a common ground on the project vision and the reasons behind the idea. This initiated a brainstorming about possible solutions. The main challenge was perceived to be the structure recognition. Therefore, we started by exploring possible technical solutions on paper. When considering radio communication and information through light signals, measuring resistances turned out to be the easiest to develop and cheapest alternative. Concerning the algorithm for structure recognition, we realized that it is simpler if each car part is only detecting its nearest neighbor instead of all parts detecting the entire structure. Therefore, resistance identification and neighbor detection were chosen to be pursued. The resistor solution was tested with resistors on a solderless prototyping board measured with ohmmeters. The principle was confirmed as functional. The idea here was to make sure as early, simple and fast as possible that principles worked with components already available in the lab in order to minimize the amount of time wasted in case it did not work. In the beginning of the **second week** of the development journey, we determined that we require three electrical connection points for connecting two car parts. This fit with the already existing BitSnap connectors [5] that had three electric connection points. They used magnets and their own shape to ensure a consistent and non-ambiguous electrical connection. Those BitSnap connectors were designed to be soldered onto a circuit board, and solderless prototyping boards were too large to fit into the car parts. Knowing that the principle worked, we decided that spending time for manufacturing a soldered circuit board version was a safe investment. The result is shown in figure 2 on the left. From this prototype we learned that the BitSnap connectors were mechanically not rigid enough to support the weight of the car parts. Furthermore, the connectors were not symmetrical, meaning that only matching pairs were combinable. This was in conflict to the fundamental idea of allowing any given car piece to connect. Anyhow, the structure recognition technically worked. Therefore, we continued developing the remaining critical functions such as measuring the resistors with a microcontroller, sending this data to a PC and displaying the measurement on a screen. We decided to not bother with improving the connectors at this point in time to save time towards achieving the critical functions of our envisioned game idea. The microcontroller measurement was prototyped using an Arduino Uno because it is easy to develop, immediately accessible in the lab, and already offers the software to display the results of the measurement on a computer screen. After merging the existing development stages and fulfilling the critical functions as early as possible (digitally recognizing a structure of mechanically attached objects, transferring this data to a PC, and displaying it) we could now focus on improving the existing solution. During **week three**, we intended to focus on shrinking the Arduino microcontroller solution to a size that is suitable for embedding in a car part. Light Blue Beans [6] appeared to be a suitable solution that is already available in the lab. They also had the advantage of replacing the wire connection between the Arduino and the PC by wireless Bluetooth communication. One upcoming problem with Light Blue Beans was that they only have two analogue inputs. This lead to the use of shift registers to channel many measurements through few input pins on the microcontroller. The shift register also work in combination with the Arduino Uno and the Arduino was kept because it is more convenient to program.

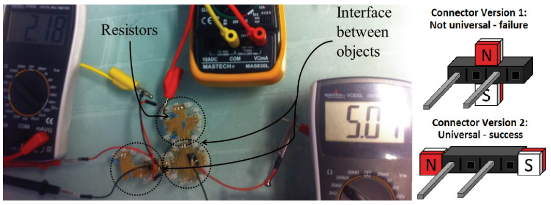

Resistors
Interface between objects
Connector Version 1: Not universal - failure
N
S
Connector Version 2: Universal - success
N
S

Fig. 2. Left: the resistances on the connected circuit boards in the middle are unique for each connection and measured by ohmmeters. Right: two sequent designs of a mechanically more stable connector.

Week four started with developing mechanically stronger and symmetrical connectors. This was implemented by using larger and stronger magnets and using pin connectors to further stabilize mechanically. The first design is shown in figure 2 in the top right. However, this design turned out to be impossible to connect to an identical connector because the magnet orientation would not match. It required a matching counter piece and was thus no improvement to the previous solution with the BitSnaps. The bottom right design solved this problem. This design flaw was discovered by building and testing the design in a very rough way instead of technically drawing and machine producing the parts. This decreases the risk of design errors and thereby saves more costly resources at a later development stage when such errors have more profound implications.

All electrical components were now on two large breadboards that required a lot of space. The components had to be merged on one platform so that they would all fit safely inside a physical shell. This was accomplished by soldering all components (transistors to control a shift register, reference resistors and header connectors) compactly onto a custom circuit board.

The next issue to be tackled was to advance the virtual representation from a line of text to a car look-a-like representation. We took two approaches into account: The first was a photograph based version where the PC would display a corresponding picture for every possible combination of parts. The second was using 3D models for representation of the car structure. We decided to develop the second option because the number of pictures needed for the first was inconveniently large when scaling up the number of car parts. We used Processing [7] to process the data coming from the Arduino, determining the structure and displaying the models on the PC screen. The system was first tested by displaying a rocket and a chair as substitutes for the virtual car model. Only after verifying the concept, we continued to make virtual representations of the car parts using a CAD software and importing those models to Processing. After confirming that Processing was a reasonable option for displaying a digital representation, **week five** began by drawing the car model parts and implementing them within Processing. At the same time, we also pursued the implementation of Light Blue Beans to make the physical model wireless. However, we experienced that Windows 8.1 did not allow importing serial data via Bluetooth. We could not instantly resolve this problem with the resources at hand and therefore decided to move back to the proven technology to not lose more time with this issue. During **week six** we explored switches, buttons and potentiometers as extra tangible inputs to alter the car parts. Since the gameplay did not yet exist, the visual representation was the only possibility to make adjustments to. We showed that this extension was technically functional and could also be used to change non-visual properties in the game later on. But there was no meaningful reason to develop something further that had no use at the current development stage. Therefore, we stopped after the proof of concept and continued to make a laser cut physical car model in acrylic. The acrylic car model was combined with the existing technology and combined all aspects from physical model to structure recognition and virtual representation. Figure 1 shows the prototype after these six weeks of development.

3 Reflections on the Development Process

It turned out that our process was very similar to the wayfaring process described by Steinert and Leifer [8]. Both processes are largely based on rapid iteration cycles of design, build and testing ideas as early and quickly as possible. We tested the most critical functions with the resources that are readily available in the lab to fail early and mitigate the risk of losing advances that become unusable due to a later design changes. The early testing lead to learnings that shaped the development journey; the design emerged over time.

4 Detailed Description of the Latest Prototype

The final prototype consists of one central part connected to a PC, and four external objects that can be attached to the central part. The central part has four connectors, one on each vertical side, on which external parts can be attached; each external part has only one connector. When no external parts are attached to the central part, a 3D model resembling the central part is displayed on the connected PC screen. Upon attaching an external part to the central part, a virtual 3D model resembling the attached part is automatically updated.

The identification of the neighboring car parts is achieved by the measurement of resistors through the connectors on the sides of the car parts. All connectors are made from 4 pin headers where two alternating pins are pulled out (see figure 2, bottom right). In the external parts' headers, a resistor is placed with one pin hole between its two legs. In the central part headers, two wires are connected to the female pins that the external connectors will fit into. Thus, when an external part connector is connected to a central part connector, we get a closed loop that runs through one wire into one of the central part header pins, through the male pin on the external part header, through the resistor, and back across to the other wire. This design is made with the intention of having multiple 'central parts' in the future that can measure each other's resistors. So far in this prototype, the central part pins that connect to the external header serve only for structural integrity.

The connector wires are connected to an analogue gate and ground on an Arduino Uno. The Arduino is able to calculate the resistance between ground and the analogue gate by comparing it to a reference resistor between its 5 volt supply and ground. Because there are four connectors and we use only one analogue gate, a shift register is used to control which connector has current at any time. The shift register is placed on a custom made circuit board along with the reference resistor, four transistors, two rows of headers for the connector wires, and a series of headers for easy connection of wires from the Arduino. Three wires connect three digital pins on the Arduino to the shift register. The shift register is connected to the gate pins on the transistors which open the current through the various connectors. Thus, the loop through ground, resistor, and analog gate is controlled. The circuit board is placed inside the central part and connected to the Arduino through a total of six wires (three digital, 5V, ground, and analogue).

When measuring the resistors, the Arduino uses as sequence of North, West, South, and East when the central part is seen from above. For each measurement, the value is serial printed, and a semicolon is added between the values. Processing 2.2.1 imports the string through the COM port on a PC. Before Processing can use the data for anything, it must convert the string into integers and store them in an array. The semicolons act as delimiters for the values. Processing then takes each value in the array and compares it to a set of thresholds.

Processing displays a rotating 3D model resembling the central part in a window. Depending on which interval between thresholds a certain measured value is, Processing displays a corresponding 3D model next to the central part. The correct position is acquired by the position of the value in the array, thus the reason for the compass sequence in the Arduino.

All 3D models are made in Autodesk Inventor [9] and converted into an .obj format. Processing loads the models in the setup of the script, and only displays them when receiving not NULL values from the Arduino.

The physical objects are made from laser cut pieces of 5mm thick acrylic plastic sheets. All pieces are modeled and assembled in Inventor before converting to a 2D format fit for cutting. The pieces are then assembled together with circuit board, central part connectors, and external connector. The pieces are held together with hot glue and clear tape so that broken pieces can be removed and retrofitting is easier.

5 Future Plans

In the near future, we will focus on improving the existing prototype by including wireless communication, universally orientable connectors, alternate modes of inter-object communication, more than one 'smart part', and how to merge our tangible programming prototype with actual gameplay. We will continue to use a wayfaring mind set as we are satisfied with the results it has yielded so far. Looking further ahead, developing and testing of real gameplay is needed before we can undergo user testing and subsequent reiterations.

Acknowledgments. We extend our most sincere gratitude to project owner Leonore Alexandra Nilsen, CEO of Metis Productions AS, for giving us this challenge and introducing us to the Fibo Car Project. This research is supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30.

References

1. Web page about "Kerbal Space Program", https://kerbalspaceprogram.com/en/?page_id=7 (retrieved April 28, 2015)
2. "Besiege" sandbox game, http://www.besiege.spiderlinggames.co.uk/ (retrieved April 28, 2015)
3. lEGO Mind Storms EV3, http://www.lego.com/en-/mindstorms/?domainredir=mindstorms.lego.com (retrieved April 28, 2015)
4. Arduino Uno microcontroller board, http://www.arduino.cc/en/Main/HomePage (retrieved April 28, 2015)
5. Little Bits Electronics, BitSnaps, http://littlebits.cc/accessories/bitsnaps (retrieved April 28, 2015)
6. Lightblue Beans, https://punchthrough.com/bean/ (retrieved April 28, 2015)
7. Processing programming language, https://processing.org/ (retrieved April 28, 2015)
8. Steinert, M., Leifer, L.: 'Finding One's Way': Re-Discovering a Hunter-Gatherer Model based on Wayfaring. Int. J. Eng. Educ. 28(1), 251–252 (2012)
9. Autodesk Inventor, http://www.autodesk.com/products/inventor/overview (retrieved April 28, 2015)

C.3 Contribution 3: A Simultaneous, Multidisciplinary Development and Design Journey - Reflections on Prototyping

A Simultaneous, Multidisciplinary Development and Design Journey – Reflections on Prototyping

Achim Gerstenberg[1], Heikki Sjöman[1], Thov Reime[1],
Pekka Abrahamsson[2], and Martin Steinert[1]

[1]Department of Engineering Design and Mat.,
[2] Department of Comp. and Info. Sc.
NTNU, Høgskoleringen 1, 7491 Trondheim, Norway
{Achim.Gerstenberg,Heikki.Sjoman,Thovr,Pekkaa,
Martin.Steinert}@ntnu.no

Abstract. This paper proposes a wayfaring approach for the early concept creation stage of development projects that have a very high degree of intended innovation and thus uncertainty. The method is supported by a concrete game design example involving the development of a tangible programming interface for virtual car racing games. We focus onto projects that not only have high degrees of freedom, for example in terms of reframing the problem or iterating the final project vision, but are also complex in nature. For example, these can be projects that allow for the exploration and exploitation of **unknown unknowns** and serendipity findings. Process wise we are primarily focusing onto the early stage that precedes the requirement fixation, which we see as more dynamic and evolutionary in nature. The core conceptual elements that we have derived from the development experiences are: **simultaneous prototyping** in multiple disciplines (such as computer science, electronics and mechanics and engineering in general, **abductive learning** based on the outcome of rapid cycles of designing, building and testing prototypes (**probing**), and the importance of **including all the involved disciplines** (knowledge domains) from the beginning of the project on.

1 Introduction

To innovate incrementally is hard, to innovate “radically” harder still. Many an engineering project is fixating their requirements very early and then focus onto executing these predefined (and often unproven) specs as fast, as good, and as cost effective as possible. The usual outcome is a cost and/or time overrun if the innovative specs are to be met or a decrease in result quality. In a sense people perceive the innovation game often as a game under certainty with fixed variables and attribute values, fixed rules and thus predictable outcomes, hence it can be modeled, simulated and optimized. We argue that the innovation game is a game under uncertainty, with unknown unknowns that need to be discovered, evaluated and then discarded or embodied. The game is also played in a dynamic environment (opponents may counter and react) and even the rules are technically not fixed - take the Kobayashi Maru test situation as an example.

© IFIP International Federation for Information Processing 2015
K. Chorianopoulos et al. (Eds.): ICEC 2015, LNCS 9353, pp. 409–416, 2015.
DOI: 10.1007/978-3-319-24589-8_33

We argue that the development of highly innovative/uncertain and products is rather like an exploration journey. You have a vision where you want to end up and a general idea where your project is heading. However, neither can you know all the “moves” required to get there, nor can you accurately anticipate the effects and responses that one move will have in the future. Your expertise is your toolbox and it greatly helps in “playing your way through the project”. Nevertheless the project is dependent on many unforeseeable events. In fact **unknown unknowns** (variables that are part of your problem/solution that you are neither aware off nor do you know their value) arise, serendipitous events present themselves, turning a complicated problem into a complex one - too complex to be planned out beforehand. We subsequently argue that sequential process models are not fitting for any innovative projects. [1,2].

The reference case [3] of designing a tangible game interface for racing games is used to extract reoccurring patterns during the design process and propose a method based on the experiences [4,5]

Our proposed method is based on abductive learning [6,7,8] and includes all involved disciplines from day one. This wayfaring model based on Steinert & Leifer [9] aims to allow the rapid requirement dynamics that become necessary during the development process.

2 Use of Wayfaring in the Example Case

Our example case is based on the **vision** of developing a physical car model as a tangible interface for manipulating/shape a digital car model in a virtual car racing game. A description of the project and the technical solution can be found in [3] This vision as an overarching goal was given to the developers instead of a precise list of requirements on how the technical solution is supposed to look like and the project architecture was allowed to emerge. This meant that the space of possible solutions is open, ambiguous and uncertain. In our example case, the problem became to identify car parts in the physical model that are attached to each other and to recognize the assembled structure. We explored the solution space by trying to come up with as many possible solutions to the problems as possible (divergent thinking). Possible ideas for solutions included measuring resistance, power dissipation of wireless communication devices or pulsed light communication for identifying connected pieces. For determining the structure we looked into a centralized structure with one central part that collects the data from all assembled parts and a decentralized structure that only required the detection and identification of neighboring parts. However, with no, or only little, experience it is unknown to us which of the suggested ideas was feasible to pursue. We call these unsolved uncertainties **unknown unknowns** because these open questions emerged during the development process and were in itself unknown to us before engaging the problem. We argued that resistors were the cheapest, simples and most reliable proposition and that just detecting neighboring car parts simplified the algorithm. Furthermore, having to use a specific center part restricted the liberty of freely using any car part separately in the virtual game. However, these are only arguments based on limited experience and in order to converge on the most promising proposal one has to build and test ideas to gain new knowledge. This repeating cycle of divergent and convergent thinking with designing, building and testing ideas is called **probing**. The probing cycles lead to **abductive learning** where the test result leads to design requirement changes and ideas for the next probing cycle. In our case, we realized that the measurement of the resistors fluctuates significantly. This lead to changes in the programming of the microcontroller that processes the measurement. The abductive learning from repeating cycles of probing leads to a **wayfaring** of opportunistically finding one's way through the project. This means that the test results of the last probing cycle shape the future development. Figure 1a) shows the first test of the resistor principle. There we discovered that the idea is feasible and that three electrical connections between car parts are needed. This lead to the development of the setup shown in figure 1b) that uses BitSnap connectors that serendipitously already had exactly three electrical connections and allow the user to easily manipulate the physical car model. Testing these BitSnap connectors revealed that these are mechanically not sufficiently robust and not genderless, thus limiting the combinations of mountable car parts. This learning resulted in the development of the customized connectors shown in figure 1 c) and d) where the first version in figure 1c) turned out to be also not genderless and subsequently lead to the development of the second version in figure 1d). This train of subsequent probing cycles showcases the wayfaring journey that can be successive or dead ended.

Progress is achieved by the emergence of new ideas as a result of previous probing cycles. Therefore, it is important to minimize the time spent and to maximize the learning outcome for each probing cycle. This accomplished by concentrating on just testing the **critical functions** by building a **low resolution prototype** that is reduced to the properties that are necessary to only test the critical function. An example for this is the testing of the resistor principle as it is shown in figure 1a). The critical function was to find out how resistors can be used to identify connected parts unambiguously. To save time we compromised robustness, automation of the measurement, looks and compactness of the system to focus only on the critical function and thus used prototyping boards, header wires and ohmmeters that were readily available in the lab.

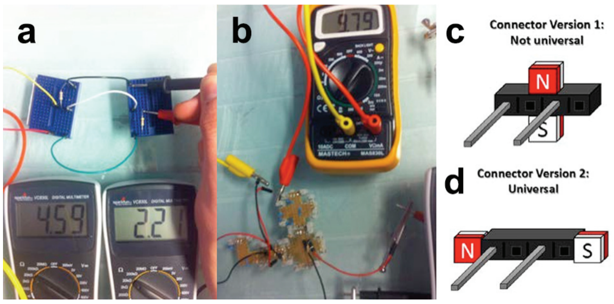


Fig. 1. a) first test of the resistor concept, b) testing with the BitSnap connectors, c) failed version of universal connector, d) successive version of a universal connector.

Imposing this train of thought to the entire project yields that prototypes that fulfill critical functions within different disciplines are merged as soon as they are available to test the system at large. The aim is to test and discover **interdependencies**. In our case, we combined the resistance measurement with a microcontroller, the information transmission to a PC and the virtual representation on the PC screen as soon as they were available in their most rudiment form. This means that all components from possibly different disciplines need to be **prototyped simultaneously**. Testing the entire system creates an **interlaced knowledge** between different disciplines. The structure recognition algorithm for example influenced the shape of the connectors and these changes had to be made in agreement with mechanical design of the car parts. This was possible because the developers of all disciplines were integrated from day one.

3 A Wayfaring Approach to Early Stage Concept Creation

In this part we describe a method that we derived from the project described above. The method has potential when finding and tackling previously unsolved engineering design problems that have no known existing solution. These problems are not necessarily complicated but rather complex according to Snowden and Boone [10]: they cannot be solved by asking experts to plan the final solution because they require the use of previously unproven and maybe even unknown concepts. In this context the development process becomes a wayfaring journey where the path towards fulfilling the vision emerges from making educated guesses and testing concepts, rather then a navigation journey along predefined waypoints. An optimum solution cannot be predicted when doing things that have never been done before. This method concerns only the early part of product development, the fuzzy front-end of concept creation, where the requirements of the product are not yet fixed. Figure 1 depicts such a wayfaring-inspired product development journey. This is a systematic and heuristic approach to developing something radically novel. The path to the end result will only be explored and discovered during the project. The journey consists of many probes. A probe is a circle of designing, building and testing of an idea or a prototype. In the figure 2, probes are depicted as multiple circles and may contain branching of ideas and prototypes on a multidisciplinary level or even dead ends. Each circle level corresponds to a role or a discipline in the project. At first, the team takes the best-guess direction based on the initial vision. Through multiple probing and prototype cycles the team then tries to find the big idea worth implementing. This journey can be long or short, but the main point is to learn fast with low-resolution prototypes. Through these prototypes one develops the requirements dynamically as perception of the problem and the vision of the solution will change during the journey. In a nutshell, we increase the degrees of freedom in the early design phase, develop requirements dynamically, and only then switch into classical engineering/project management mode.

While researching radical innovation projects, our chess analogy is lacking because in chess it is theoretically possible to calculate the move with the highest probability of winning the game. However, in the product design “game” the possible future moves, players, even the boundary conditions are often neither comparable nor foreseeable. There are **unknown unknowns** that create opportunities for extremely innovative solutions but also prevent us from predicting or simulating an optimum solution. In this analogy, the rules of the chess game can change without notice and we can only provide a journey overview in hindsight, roadmaps do not apply. The Hunter-Gatherer model by Steinert and Leifer [9] and Ingold [11] inspired this wayfaring concept.

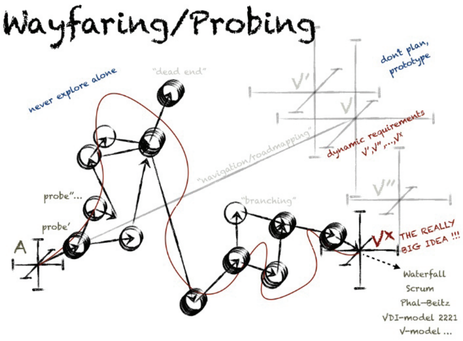


Fig. 2. Wayfaring journey in product development.

Many of our engineering problems are **multidisciplinary** and require interdependent knowledge between disciplines that cannot be covered by individuals or homogeneous teams. Two or more disciplines of the project are interdependent when design changes in one discipline lead to requirement adaptation in at least one other discipline. We argue that including team members or at least domain perspectives from all involved disciplines early in the project helps to reveal desirable and undesirable **interdependencies** already in early decision making phases. Even if actual deliverable input from every member is dispensable early on, the benefit of learning early overcomes the cost of participation. One of the greatest threats in new product development is the fear of failure [12]. According to Snowden [10] safe failing is identified as one of the cornerstones while innovating in the complex domain. The **interlaced knowledge,** developed through sense-making and justification of ideas to the other involved disciplines, is also beneficial when designing within one discipline while having the entire system in mind and thereby knowing when the other disciplines need to be taken into account and their input is needed [13]. This is a skill that can only be learned when combining all involved disciplines from the first day of the project.

The nature of trying out new concepts entails that outcomes cannot be guaranteed and some problems, opportunities and interdependencies are difficult, if not impossible, to foresee. When trying out something never attempted before we can no longer base our assumptions on past experiences and unexpected discoveries can arise. Snowden calls these discoveries **unknown unknowns** because we unknowingly discover something previously unknown [10]. In order to achieve these unexpected discoveries new experiences must be created from **probing ideas**. One of the ideas of probing is therefore to build and test prototypes that create completely new knowledge – knowledge that is impossible to accurately anticipate regardless of what our expectations may be. The concept of probing is depicted in Figure 2. Each probe is a prototype where new knowledge is deductively, inductively and/or abductively created and tested. The vision and requirements are then evolving dynamically until they are locked. The development cycle is executed through different roles of disciplines. Each probe is ideated through divergent thinking where open questions are asked in order to stimulate the creative process followed by convergent thinking, that evaluates and analytically benchmarks the ideas through proof-of-concept prototypes. The interesting interlaced knowledge lies in the boundaries of the different disciplines and presents the potential for serendipity discoveries.

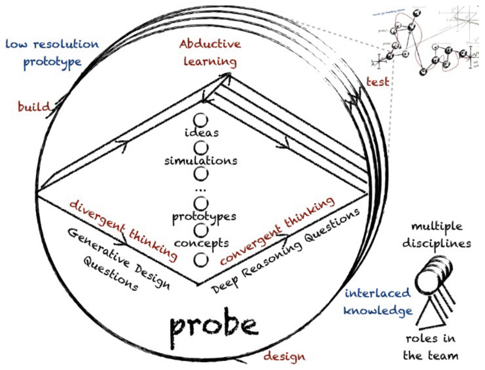


Fig. 3. Probing cycle

To continue with the chess analogy, we do not expect to win if we must plan all our moves (and anticipate the opponent's) in the beginning. However, if allowed to experiment and revert moves a thousand times during the game, it will quickly become a game of probing (or prototyping) multiple moves. Through not following an optimal game strategy, this will eventually lead to overall winning the game in case of a complex game scenario. Because the cost of probing is minimal, it allows us to explore opportunities that are not immediately perceived as profitable. It leads to moves that would normally not be taken, to discoveries that are normally not found, and may potentially lead to surprising and highly innovative ways of winning the game. Therefore, the aim must be to make the probing and the learning of ideas as low-risk (i.e. fast and cheap) as possible in order to create the experience needed to reflect, to understand the outcome, and then **abductively reason** and opportunistically choose the next step [14].

The notion is to put the focus on testing the **most critical functions**, thus leaving the development of the "nice to have" add-ons for later. It is preferable to utilize the resources for discovering the essentials and preferably fail there early. The probing removes uncertainty and an undiscovered problem is revealed before it forces undesired **requirement changes** at a later stage [15]. The testing usually involves building a **low resolution prototype** with the intention to either find the critical function or to build a prototype for user testing in order to avoid developing into an unnecessary direction. Low-resolution prototypes can be anything from cardboard models to Arduino hacks to proof-of-concept prototypes. Often developers have major problems in failing. Low-resolution prototypes in very fast iteration rounds do not resemble the finished object and are thus one way to allow and speed up experimentation. It seems to be inherent to human nature to fear failure, thinking it will cost too much. This can lead to a non-willingness to take risks and make cooperation hard with people from other disciplines. This skill of creative competence [16,12] does not come naturally. This is why changing the mindset into one that favors building prototypes with the option of failing safely before planning is critical while developing new concepts. Hence, despite the natural fear of failing, the mindset should be biased towards building low-resolution prototypes in order to gain experience instead of thinking the idea through and remaining with doubt.

Another finding is to **merge system components** as soon as possible in order to tackle potential integration issues very early on. This follows the same line of thought as aiming to discover unknown interdependencies as early as possible. Whenever a component individually fulfills its critical function, it ought to be integrated with other components to test its critical function in the context of the whole system. So, even when the system can and is divided into modules, integration should be tested while changes to the system are still easily possible. We believe that there is no point in fully developing one component and then risking requirement changes in other components that would endanger the previous development. This requires quasi-**simultaneous prototyping** to ensure that components can be merged. Thus in our context, simultaneous prototyping means understanding and probing ideas from multiple disciplines at the same time.

The **main purpose of probing** is to find solutions to the evolving problem by abductive reasoning and to continuously update the understanding of the problem. While probing different paths for the project one of the most important mindsets is to be opportunistic, to find, recognize and take chances that present themselves. Another benefit is the possibility to abandon disadvantageous concepts, "dead ends", in an early stage at the lowest cost and involvement possible. All in all, the wayfaring model calls for a bias towards action and learning in action.

4 Conclusions of Wayfaring

We propose a method suitable for developing new products with a high degree of uncertainty. It is largely based on including all disciplines related to the product from the beginning on and iterative cycles of probing ideas by designing, building and testing prototypes. The intent of this approach is to discover **unknown unknowns** and unexpected interdependencies early in order to minimizing losses due to failure and to spot opportunities and hitherto unknown potentials. Both, the initial problem statement and the targeted project vision remain in flux much longer than usually. The relatively early requirement fixation stage becomes a delayed dynamic requirement evolution process. The decisions to fix the dynamic requirements are made based on gained and tested information, based on learning cycles trough low-resolution prototyping and probing. We believe the headway and learnings, both in terms of breadth and depths have been superior to pre-planned or more traditional process models. We thus invite the community to deploy and test this approach in the early, pre-requirement definition phase and to share their insights.

Acknowledgments. We thank the project owner L. Nilsen, CEO of Metis Productions AS. This research is supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30.

References

1. Sanchez, R., Mahoney, J.T.: Modularity, flexibility, and knowledge management in product and organization design. Strategic Management Journal 17(S2), 63–76 (1996)
2. Baldwin, C., Clark, K.: Design Rules: The power of modularity. MIT Press (2000)
3. Reime, T., et al.: Proceedings of the 14th International Conference on Entertainment Computing, ICEC 2015, Trondheim, Norway, September 29-October 2, 2015. Springer (2015)
4. Eisenhardt, K.M.: Building Theories from Case Study Research. The Academy of Management Review 14(4), 532–550 (1989)
5. Yin, R.K.: Case Study Research: Designs and Methods. SAGE Publications (2013)
6. Burks, A.W.: Peirce's theory of abduction. Philosophy of Science 13(4), 301–306 (1946)
7. Eris, O.: Effective inquiry for innovative engineering design. Springer Netherlands (2004)
8. Leifer, L.J., Steinert, M.: Dancing with ambiguity: Causality behavior, design thinking, and triple-loop-learning. Information, Knowledge, Systems Management 10(1), 151–173 (2011)
9. Steinert, M., Leifer, L.J.: "Finding One"s Way': Re-Discovering a Hunter-Gatherer Model based on Wayfaring. International Journal of Engineering Education 28(2), 251 (2012)
10. Snowden, D.J., Boone, M.E.: A Leader's Framework for Decision Making. Harvard Business Review, 69–76 (2007)
11. Ingold, T.: Lines: a brief history. Routledge (2007)
12. Bandura, A.: Perceived self-efficacy in the exercise of control over AIDS infection. Evaluation and Program Planning 13(1), 9–17 (1990)
13. Türtscher, P., et al.: Justification and Interlaced Knowledge at ATLAS, CERN. Organization Science 25(6), 1579–1608 (2008)
14. Schön, D.A.: The reflective practitioner. Basic Books, New York (1983)
15. Kriesi, C., Steinert, M., Meboldt, M., Balters, S.: Physiological Data Acquisition for Deeper Insights into Prototyping. In: DS 81: Proceedings of NordDesign 2014, Espoo, Finland, August 27-29 (2014)
16. Kelley, T., Kelley, D.: Creative confidence: Unleashing the creative potential within us all. Crown Business (2013)

NTNU – Trondheim
Norwegian University of
Science and Technology

A Simultaneous, Multidisciplinary Development and Design Journey of Bridging Tangible and Virtual -Reflections on Prototyping

Achim Gerstenberg[1*], Heikki Sjöman[1], Thov Reime[1], Pekka Abrahamsson[2], Martin Steinert[1]

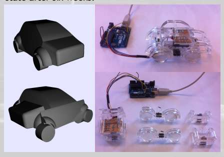
* Achim.Gerstenberg@ntnu.no
1: Department of Engineering Design and Materials, TrollLABS, NTNU
2: Department of Computer and Information Science, NTNU

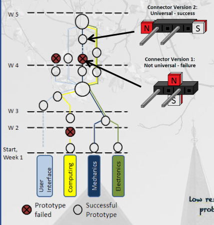
Abstract

*This paper proposes a **wayfaring** approach as a method suitable for the early concept creation phase of development projects that have a very high degree of innovation. The method is supported by a concrete game design example focusing on the development of a tangible programming interface for virtual car racing games. Process wise we are solemnly centered onto the early stage that precede the requirement fixation, which we see as more dynamic and evolutionary in nature. The core elements that we have derived from the development experience are: **simultaneous prototyping** in multiple disciplines, **abductive learning** from rapid cycles of designing, building and testing prototypes (**probing**), and the importance of **including all involved disciplines** (knowledge domains) from the beginning of the project.*

Example Case

The Fibo Car is a tangible toy interface for car games. The prototype's technology is based around an Arduino Uno [1] that measures unique resistors placed inside car-like physical parts. The exterior parts are attached to a central part by header connectors that provide electrical connection and structural integrity. A circuit board containing a shift register inside the central part allows for the measurement of resistors in four parts at the same time. The measured values are sent to a computer where Processing 2.2.1 [4] analyses the data and displays 3D models representing the physical parts, thus giving a real-time virtual representation of the physical assembly. The prototype did not include input from the gameplay domain because we assumed that the resulting prototype would have too low resolution to be included in actual gameplay: a high resolution prototype with gameplay would be made in the future. The prototype presented in the figure below shows the state after six weeks.



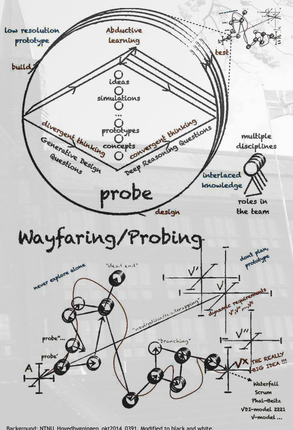


Examples of wayfaring in case

- **Interdependencies:** Business model and game design affected the domains of mechanics, computation, electronics, and virtual representation: representants from all disciplines should be involved in the early phase.
- **Unknown unknowns** and **abductive learning:** Resistors as identifiers had initially promising results, but its instability made it a non-viable solution for the end product. New magnetic connectors led to the discovery of how symmetry affects design of genderless connectors (see figures above).
- **Low resolution prototypes** and **probing:** The principle of the electrical connectors was prototyped at a low resolution level with minimal time and resource investment before making actual connectors. This formed the basis for the **critical function** of part identification. Existing 3D models from Processing's example library were used to prototype the critical function of displaying 3D models based on signals from Arduino.
- **Merging and simultaneous prototyping:** Electronics and mechanics were already merged by combining LittleBits Bitsnaps [3] and the three pin electrical connector principle on soldering boards during week two.

Proposed method

The Hunter-Gatherer model by Steinert and Leifer [6] and Ingold [2] inspired this wayfaring concept. We believe this method has potential in unsolved engineering design problems with dynamic requirements and high level of interdependencies between related engineering disciplines. These problems are complex according to Snowden and Boone [5]: they require the use of previously unproven or unknown concepts; optimum solutions cannot be predicted. We suggest that the path shall emerge from **probing** in multiple domains simultaneously: designing, building, and testing of an idea or prototype related to preferably all knowledge domains. This quasi-**simultaneous prototyping** ensures that components can be merged in the future. Including perspectives from all domains help reveal **interdependencies** and build **interlaced knowledge**. Risk taking and failing is used to encourage testing a wide range of opportunities and yield discovery of **unknown unknowns**. Through multiple probing and prototyping cycles, the team tries to **abductively reason** and find the big idea worth implementing. The main point is to learn fast with **low-resolution prototypes** and **merge** these as soon as possible to validate or falsify **critical functions**. Another benefit is to abandon "dead ends" quickly. As the vision of the solution and problem perception changes, the requirements develop dynamically. After the idea has been found, it is passed forward to more traditional approaches like the waterfall model or the scrum model. All in all, this wayfaring model calls for a bias towards action and learning by doing.



Background: NTNU_Hovedbygningen_okt2014_0391. Modified to black and white.
Foto: Gunnar K. Hansen, NTNU Komm.avd.

Conclusion

We derived a method for developing new products of a high degree of uncertainty. It is based on including all disciplines from the beginning and using iterative cycles of probing ideas by designing, building and testing prototypes. The described example has covered a project time of six weeks of a single Full-time equivalent (FTE) position. We believe the headway and learnings have been superior to traditional process models. We thus invite the community to deploy and test this approach in the early, pre-requirement definition phase and to share their insights.

Acknowledgements

Thanks to Leonore A. Nilsen, CEO of Metis Productions AS, and to TrollLABS. This research is supported by the Research Council of Norway through its user-driven research (BIA) funding scheme, project number 236739/O30.

References

[1] Arduino Uno Microcontroller Board, 2015. Retrieved April 28, 2015: http://www.arduino.cc/en/Main/HomePage
[2] Ingold, T. (2007). *Lines: a brief history*. Routledge.
[3] Little Bits Electronics, BitSnaps, 2015. Retrieved April 28, 2015: littlebits.cc/accessories/bitsnaps
[4] Processing programming language. Retrieved April 28, 2015: https://processing.org/
[5] Snowden, D. J., & Boone, M. E. (2007). A Leader's Framework for Decision Making. *Harvard Business Review*, 69 - 76
[6] Steinert, M., & Leifer, L. J. (2012). "Finding One"s Way': Re-Discovering a Hunter-Gatherer Model based on Wayfaring. *International Journal of Engineering Education*, 28(2), 251.

C.4 Contribution 4: Large-scale Engineering Prototyping - Approaching Complex Engineering Problems CERN-style

C.5 Contribution 5: Open-ended Problems - A Robot Programming Experiment to Compare and Test Different Development and Design Approaches

Not included due to copyright restrictions

C.6 Contribution 6: Testing the Effect of Desirable Difficulties on Teaching Robotics

1
CERN IdeaSquare Journal of Experimental Innovation, 2020; X(X): X-X 1
DOI: https://doi.org/10.23726/cij.2020.???

Testing the effect of desirable difficulties on teaching robotics

Achim Gerstenberg,[1*] Martin Steinert,[1]

[1] Department of Mechanical and Industrial Engineering, NTNU, 7491 Trondheim, Norway
*Corresponding author: achim.gerstenberg@ntnu.no

ORIGINAL ARTICLE

ABSTRACT

Desirable difficulties such as generating one's own solution instead of replicating a provided solution is associated with improved long-term memory. Disseminating misleading information has shown improved learning in science education over consuming concise and clear learning instructions. We aim to quantify if a tutorial about programming a mobile autonomous robot that requires having to correct misleading instructions leads to better problem solving capabilities than providing correct and clear tutorial instructions when asked to solve a complicated open-ended robotics task. We present an experimental setup for a controlled comparative human-subject study that compares the effect of desirable difficulties on participant's performance in solving a complicated open-ended task after completing an introductory tutorial. We explain the experiment timeline, the tasks of the tutorial, as well as the open-ended task including the robot and how this experiment can be executed under very controlled, repeatable and as unbiased as possible conditions. We share and discuss some observed problems in this setup from early trials.

Keywords: Desirable difficulty, Human subject experiment, Problem solving

Received: August 2020. Accepted: Month Year.

INTRODUCTION

Desirable difficulties such as generating a solution instead of being told a solution (Bjork 1975; Bjork 1994; Bjork 2011) and using tests instead of presentations (Roediger & Karpicke 2006) have been shown to improve long-term retention of the learning content. In addition, Muller found that noticing and correcting false information in science education videos significantly improved the understanding (Muller 2007).

We present an experimental setup for a comparative study that quantitatively investigates if introducing desirable difficulties improve short-term performance in open-ended problem solving. In the learning phase, we use flaws in an example solution that the participants need to correct as a generative desirable difficulty. We furthermore describe how such quantitative experiments can be conducted under controlled conditions to minimize biases introduced by inconsistent interaction with the participant.

EXPERIMENT AGENDA

During a series of tutorial tasks that familiarize the participants with the programming of a robot, the participants are either presented a working solution to the task or a dysfunctional, flawed solution that they need to correct. In addition, all participants receive an explanation of the library functions needed. The flawed or unflawed solutions are presented on a separate screen so that participants cannot automatically copy and paste the solution into the programming environment. This ensures that they need to either create their own solution or need to fully read and then retype the provided solution. After completing nine tutorial tasks, which introduce the participants to the functionalities of the robot necessary for this experiment, all participants are presented an open-ended task. The time from when the robot is started until it has completed the open-ended task is used as a measure of performance. The evaluation takes place after the programming phase. We suggest a programming phase of eighty minutes.

This experimental setup aims to research if desirable difficulties increase problem-solving performance in an open-ended task. Accordingly, the hypothesis to be tested is:

Participants that correct flawed solution suggestions in tutorial tasks afterwards generate solutions that solve the open-ended task faster than participants that received unflawed solutions during the tutorial tasks.

The following paragraphs describe the robot used, the tutorial tasks and the open-ended task.

Additional information about the physical setup, a more precise description of the robot and the library, the standardized interaction with the participants and other experiment examples this setup can be used for are described in (Gerstenberg & Steinert 2018).

THE ROBOT

The robot is based on Lego Mindstorms NXT 2.0 and has two motors to drive forwards, backwards and turn, and sensors to detect the reflectivity of the surface underneath the robot, an ultrasonic distance sensors and light sensors that can detect the color of an object and recognizes a blinking light. It is programmed in the NXC (not exactly C) programming language. A library that simplifies the programming of the robot is provided to the participants. This library includes functions for sensors, outputs and movement. For example, a function that separately defines the PID controlled motor speeds on each side in percentage values from negative 100 to positive 100, spares the participant the need to program a closed feedback loop for the voltages to each motor. Other functions simplify turning of the robot, reading out sensor values, generating random numbers, play tones and display characters on the robot's screen. Figure 1 shows a front view of the robot.

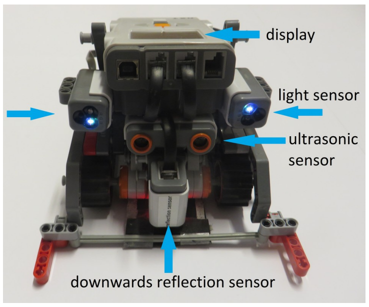


Fig. 1. Detailed front view of the robot with its four sensors. The motors and belts for moving the robot are positioned on the sides and are not visible.

TUTORIAL TASKS

The aim of the tutorials is on the one hand to introduce the participants to the robot and its capabilities and on the other hand to provide the differentiation between the flawed and unflawed condition. The tutorial consists of 9 separate tasks where each tasks introduces one new functionality from the library. While working on the tutorial the participants can execute the codes that they have created. They are presented the next tutorial task when the robot fulfils the current task. In the case that the experimental group, that needs to find the flaw in the solution, does not solve the task within the time limit they are shown the solution for 1 minute. The following table lists the nine tutorial tasks with a short description of the flaw used for the experimental group and the provided time limit.

#	Task description	Flaw description	Time limit
1.	Write a program that each second plays a sound for 100 ms at 440 Hz.	Sound has 4400 Hz and the time between sounds is 1 ms instead of 1 second	3 min
2.	Show the elapsed time in seconds on the display.	Time variable starts at 42 seconds and the display position changes each second instead of the value	3 min
3.	Drive straight forward as slowly as possible (but visibly moving) for 5 seconds then turn 90 degrees counterclockwise with speed 10 using the turn function. When the turn is completed, drive straight backwards as fast as possible for half a second and then stop both belts for two seconds. Spin the robot clockwise for 2 seconds at half speed using both belts and stop.	Robot turns in the opposite direction in both cases and uses a forbidden speed value.	5 min
4.	Place the robot on the aluminum foil and drive forward as long as the robot is on the aluminum foil. Use the downwards light reflection sensor to find out when the robot (or more precisely the sensor) reaches the edge of the aluminum foil and stop the robot.	Too low threshold for detecting the edge of the aluminum foil. The robot keeps driving after reaching the edge.	5 min
5.	Place the robot in front of the red or green cardboard wall. Drive towards the wall. If the wall is green stop in front of the green wall. Keep driving and crash into the wall if it is red.	Robot only compares the green light reflection between the sensors on each side and does not compare to the red.	5 min

6.	Place a blinking light 20 cm behind the robot. Rotate the robot on the spot until the robot detects a blinking light. Stop the robot when it detects the blinking light.	Robot turns so quickly that the light sensors are not facing the blinking lights for the duration of one blinking period and consequently are not properly detecting the lights.	5 min
7.	Place the robot facing towards the 50 cm wide cardboard wall and drive towards it. Stop between 20 and 30 cm in front of the wall using the ultrasonic distance sensor. Turn the robot until the ultrasonic sensor no longer detects it and drive past the cardboard wall.	The threshold is chosen lower than the minimum distance the sensor can detect and therefore the robot keeps driving into the wall.	5 min
8.	Drive around with random speed (between full speed backwards and full speed forwards) on each wheel by using the random function. Change wheel speeds every second	The input order of the parameters to the random() function is inverted resulting in the function returning 0 instead of random values between -100 and 100.	3 min
9.	Equivalent to the previous task but stop the robot after 5 seconds using timers	Timer 1 is started but timer 3 is used for the timing. After this is corrected the time until the program stops is 5 milliseconds instead of 5 seconds.	3 min

OPEN-ENDED TASK

The aim of this task is to attain a performance measure that allows for a quantitative comparison of the two conditions. The task is open-ended, meaning that there is not one single clear solution and many different ways of finding them. While the tutorial tasks each contain a single new code component, the open-ended task requires a creative combination of several components to be solved. The participants need to transfer knowledge from the tutorials and adapt it to the new context of the open-ended task.

The task is to remove three cube-like objects from a white area of 1.8 square meters in the shortest time possible after starting the robot. The starting position of the robot is unknown to the participant and the solution is supposed to work from any possible starting position. Three blinking lights are provided and can be placed anywhere including inside the cube-like objects. The robot cannot be manually influenced after it is started.

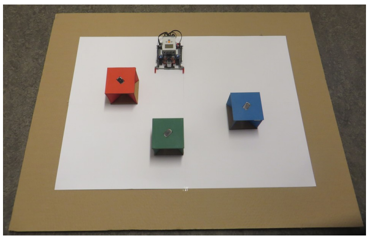


Fig. 2. Setup of the three coloured cubes on the 1.8 square meter large white area that the autonomous robot needs to push onto the surrounding.

INTERACTION CONTROL

Qualitative research offers insights into why differences occur while quantitative research setups, like the one presented here, allow researchers to quantify effect size with usually fewer insights about why a difference between conditions exists. This means that it is important to have one exclusive difference between conditions such that differences in the results can be linked to this single stimulus. Therefore, everything else that may have an influence on the results needs to be kept equal for every participant.

Common biases in human subject experiments are dependent on the behaviour of the experimenter and the perceptions of the participant. Apart from carefully designing and testing instructions, we cannot control for how a participant perceives and interprets them. However, we can control how the instructions are presented and can reduce biases introduced by direct human-human interaction with the experimenter. This experiment setup is designed without any direct oral and visual experimenter to participant interaction by providing instructions through pre-recorded voice and video instructions that use computer generated voices to avoid emotional inflictions through voice tonality. Other instructions are given as text on a screen or paper, and the timing when information is presented follows a predefined script. This process ensures that each participant receives neutral and similar instructions.

PRELIMINARY RESULTS

We tested this setup with mechanical engineering students with basic programming skills from a mechatronics course. However, it was a preliminary test as the instructions were slightly adapted in the course of the experiment, i.e. that there were small setup changes between the participants that may influence the result differently.

The results did not show a significant difference between participants that completed the tutorial without flaws versus the participants that completed the flawed tutorial. We cannot conclude that there is no effect, but the effect is not observable due to the small number of participants and the inter-participant variation of the performances regardless of the condition. All participants were able to complete the tutorials within the given time limits while not all participants were able to successfully solve the open-ended task.

LIMITATIONS AND STRENGTHS

Having an experimental setup that is equivalent for every participant limits the flexibility of the study. The desirable difficulties presented in the tutorial need to be difficult, yet solvable for the participants. Finding this challenge point requires already extensive testing prior to the actual experiment with a group of participants that have a similar skill level as the participants used later in the actual study. Since participants cannot partake in the study twice, those used in the preliminary study will not be able to participate again.

Another downside to a rigid setup is that it makes qualitative research more difficult. The setup allows for standardized qualitative data gathering such as analysing the codes the participants write, their keystrokes, video and audio recordings and questionnaires. However, such a setup does not allow for situational inquisitions like interviewing the participant during the experiment, or asking for feedback on specific design decisions or on potential problems. Direct human interaction between the experimenter and the participants is possible after gathering the data and retrospectively asking question may still lead to insightful observations.

Probably the main limitation is that different participants have varied skills and different approaches to open-ended problem solving. For example, some participants read the provided information about the robot more carefully and approach the problem in a very structured way, while others rely on frequently testing prototypes. The ability to gain insights from testing varies between participants. These differences between participants may lead to the inconclusive observation in our preliminary results and shows the importance of choosing participants with an as equal as possible skill level. This restricts the validity of the study to this type of participant but it also allows statistically significant findings with fewer participants. Another limitation and simultaneous strength is that each participant works alone. This makes the experiment less meaningful as programming nowadays is often done in pairs, but it also eliminates the uncontrollable influence of group interactions on the experiment.

The strength of quantitative research is that it can verify or falsify hypotheses objectively by statistically evaluating if a theory applies in the context of the experiment or not. A hypothesis therefore needs to be falsifiable (Popper 1935) and usually originates from a prediction made with an existing theory. If the hypothesis is falsified the theory is incomplete in the tested context. Knowing the context of the experiment is therefore essential. This includes for example the age, gender, education, culture, etc. of the participants, the field of study the theory is applied to (here robotics education) and the experiment setup (physical setup, instructions, time of the day, etc.). To validly compare results between participants it is vitally important to keep these experimental settings equal for all participants that shall be included in the comparison. When the only difference between participants is the intentionally introduced stimulus that differentiates participants from two experimental conditions then any observed difference in the results can be attributed to the stimulus. Furthermore, quantitative research can determine effect size because it allows for statistically evaluating the significance of the effect.

The aforementioned experimental setup is optimized towards repeatability of these experimental settings with a special focus on how to provide equal experiences to every participant by standardizing instructions and presenting them as neutral a form as possible.

While qualitative research helps to generate theories and find hypotheses, those hypotheses need to be quantitatively tested to verify the theory for similar contexts so that one can use the theory for predictions.

With the experimental setup described in this paper, we aim to contribute with an experiment idea for testing the hypothesis that desirable difficulties can improve open-ended problem solving and we hope to encourage the development of quantitative studies in the field of computer science education with highly controlled interactions between the participants and the experimenter.

ACKNOLEDGEMENTS

This research is supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30

REFERENCES

Bjork, R.A., 1975, Retrieval as a memory modifier: An interpretation of negative recency and related phenomena. Information processing and cognition, Loyola Symposium, 123 – 144

Bjork, R.A., 1994, Memory and metamemory considerations in the training of human beings, Metacognition: Knowing about knowing, 185 – 205.

Bjork, R.A., 2011, Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning, Psychology and the real world: Essays illustrating fundamental contributions to society, Worth Publishers, New York, NY, US

Gerstenberg, A. & Steinert, M., Open ended problems – a robot programming experiment design to compare and test different development and design approaches. Proceedings of NordDesign 2018

Muller, D.A., 2007, Saying the wrong thing: improving learning with multimedia by including misconceptions, Journal of Computer Assisted Learning, 24(2), 144 – 155

Roediger, H.L. & Karpicke, J.D., 2006, Test-enhanced learning: Taking memory tests improves long-term retention, Psychological Science, 17, 249–255

Popper, K., 1935, Logic of Scientif Discovery, Springer

C.7 Contribution 7: Development and Verification of a Simulation for Leveraging Results of a Human Subjects Programming Experiment

DEVELOPMENT AND VERIFICATION OF A SIMULATION FOR LEVERAGING RESULTS OF A HUMAN SUBJECTS PROGRAMMING EXPERIMENT

Achim Gerstenberg
Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway
achim.gerstenberg@ntnu.no

Martin Steinert
Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway
martin.steinert@ntnu.no

March 24, 2019

ABSTRACT

Quantitatively evaluating and comparing the performance of robotic solutions that are designed to work under a variety of conditions is inherently challenging because they need to be evaluated under numerous precisely repeatable conditions Manually acquiring this data is time consuming and imprecise. A deterministic simulation can reproduce the conditions and can evaluate the solutions autonomously, faster and statistically significantly. We developed such a simulation designated to leverage data from a human-subject experiment post-experimentally. We present the development of the simulation and the verification that it actually reproduces the results obtained with the physical robot. The aim of this publication is to provide insight into the development details such that other researchers can replicate the setup and to show the degree of validity of the simulation.

1 Introduction

Gerstenberg and Steinert [1] presented an experimental setup for engineering design methodology research where participants are expected to program a mobile robot to autonomously detect and move objects in the shortest time possible and from any starting position within a given area. The time the robot needs to complete the task is the performance measure that can be used for testing the influence of different design methodologies on performance outcome in such an open-ended task. During the experiment the participants generate and test their solutions. The test results during the experiment are not sufficient to obtain statistically valid performance results but the codes that the participants generate can be evaluated post-experimentally. This can be done by loading the participant codes onto the robot and then manually executing the codes, observing the robot and documenting its performance from many different starting positions. While gathering this data the conditions must be kept as equal as possible for comparability. This means that the starting conditions must be accurately repeated, the battery equally charged, frictions in the motors remain equal, etc.. To reduce the error introduced by not perfectly replicating the conditions, each starting position can be repeatedly executed and the results for this starting position are then averaged. Acquiring data manually from many starting position is very time consuming and annoying. Therefore, this paper presents a digital and deterministic simulation that speeds up the post-experimental performance evaluation and, in addition, makes it more comparable by guaranteeing equal conditions when comparing solutions. We describe the task and the robot used in the experiment that this simulation is made for, how we developed the simulation from observing the physical robot, how we validate that the simulation gives qualitatively and quantitatively valid results and use the simulation to illustrate that small perturbations in the conditions can have a significant effect on performance when evaluating only a single or few repetitions.

2 The physical robot and the task

The robot is built from LEGO technic and is controlled by a LEGO Mindstorms NXT 2.0 system. Two electric motors can move the robot via two belts; one on the left and one on the right side. The robot has in total 4 sensors. A ultrasound distance sensor is mounted in the center front and two identical light/colour sensors are mounted on the sides. The light/colour sensor can detect light intensity and measure the colour of nearby objects. These three sensors point forward. Near the front and pointing downwards, the robot has a reflection sensor. The sensor emits red light and measures how much of this emitted light is reflected back into the sensor. The reflection sensor can detect differences in reflectivity of the surface underneath the robot.

The robot is programmed by the participant in the NXC language. NXC stands for "not exactly C" and is a programming language very similar to C. The participant is provided with a library that includes functions specifically written for this robot. It simplifies the interpretation of raw data such as converting the time of flight of the ultrasound pulse from the ultrasound distance sensor to a distance in centimeters or the raw light sensor values into a meaningful value for detecting blinking lights. The library allows the participant to quickly use the robot to solve the task instead of spending time on programming the basic robot functionality.

A rectangular cardboard playground platform of approximately 1,50 m by 1,20 m has a white area rectangle in the centre surrounded by a 17 cm cardboard fringe. On top of the white area three coloured cubes (red, green and blue) are placed. The participant programs the robot so that the robot autonomously removes the cubes entirely from the white area in the shortest possible time. In the top of the cube a cut-out allows inserting one blinking light per cube that can be detected by the robot from any direction. The participants can optionally place up to three blinking lights anywhere on the playground including inside the top of the cube. The blinking lights are detectable with the light/colour sensor and using the blink variable from the library. The robot can detect the difference of reflectivity between the cardboard surface and the white area with the downwards pointing reflection sensor. If the robot drives off the cardboard playground the task is failed. The robot shall be capable to solve the task from any starting position and orientation inside the white area.

3 Development approach

In contrast to most simulations that are used to test different solutions before building the physical robot this simulation was developed after the physical robot and environment already existed in order to use the simulation to generate a performance measure in a controlled, repeatable and faster way. The approach for developing the simulation is a repeating cycle of measuring the physical robot behaviour and then digitally replicating it and comparing the real robot behaviour to the simulated robot behaviour. The movement of the physical robot and each sensor are measured individually and several times. These measurements are averaged. The averaged measurements describing the robot can be found in the datasheet [2] and are in part shown in the next subsection. The physical environment and the robot behaviour were modelled in a 2D digital, scaled and deterministic simulation. For comparing the simulation with reality the simulation and the physical model are set up under similar conditions and each sensor and movement of the robot are compared individually. If the simulation is not accurate within the error of the measurements of the physical robot behaviour then a more detailed digital representation of the physical robot is developed and the comparison is repeated until the simulated robot behaviour is well within the error range of the measurements of the physical robot behaviour. If this cannot be achieved by making more and more detailed digital representations of the physical robot then we implement elements that can no longer be explained from the setup of the physical robot. We justify this approach because the aim is not to make a comprehensive explanatory model of how the physical robot behaves but the aim is to digitally replicate the physical robot's behaviour as precisely as possible to predict task performance outcome.

After calibrating the physical shapes and sizes, robot propulsion and sensors of the simulation individually, the simulation was compared to the behaviour of the physical robot in the context of solving the task. This comparison was done with three different codes that use as different task solving approaches as possible and cover all sensors and functions available to the participants. All three are tested from several different starting positions. The qualitative comparison is done by looking if the real and simulated trajectories are similar and the interaction of the robot and the cubes and edges of the white are phenomenologically similar. The quantitative comparisons are discussed in the verification section.

4 Technical solution

For displaying, animating, programming the robot behaviour and simulating the physical interactions between the objects we use the Unity game development software version 5.6.1 [3]. The first step was to digitally replicate a top-down representation of the cardboard and the white area, the coloured cubes and the robot dimensions. This is done by drawing a scaled image of each of the objects and inserting them into the unity scene. All of these images have a box collider tracing the outline of the objects. Unity detects when two box colliders enter or exit into overlap and gives an event trigger that can be used for capturing the state of the robot in the environment. The moving objects (robot and cubes) are modelled as a rigid body. This means that the physics engine of unity calculates the interacting forces between the objects and translates them accordingly. For simulating the deceleration of the cubes, the rigid body of the cubes is assigned a drag value. Onto the robot image/object the sensors are placed as images as well. Only the downwards reflection sensor uses a box collider. It is used to detect when this sensor is entering or exiting the white area or the cardboard area.

4.1 code organization

The unity scene animation is controlled by code written in C sharp. This includes the movement and the simulation of the sensors, coordinating the execution of the participant codes in the correct order from predefined starting conditions, documentation of the robot behaviour for later analysis, the simulation representation of the library that the participants used to control the physical robot and a reoccurring simulation loop that animates the scene and calculates the physics interactions between the objects. This calculation by the physics engine and the animation are executed during a code block called “fixedUpdate()” which is provided by the Unity game engine. FixedUpdate is executed 50 times a second. Additionally to the physics engine we use the fixedUpdate class to include our code for translating the robot in the scene, update the array that saves the sensor value for the blink sensor and check if the task is completed to reset it for the next simulation. All other code evaluations are calculated in between two successive FixedUpdates. We use coroutines to coordinate the schedule for executing the participant codes. Coroutines are like threads that run between each FixedUpdate and can be yielded for specific times like until the next fixedUpdate or until another coroutine has finished. While FixedUpdate is like the relentless clock that runs until the simulation stops, coroutines can be used to execute code at specific times in a sequence. This, apart for scheduling the participant codes, is used to interrupt the execution of the participant code when those include the "wait" or "turn" command until the waiting time has elapsed or the robot has finished the turn.

4.2 simulating motors

The movement of the digital robot is modelled with two speed state variables that determine the amount of translation of the robot object during the execution of FixedUpdate. They represent each of the driving belts of the physical robot. These two speed state variables are set by the motor and the turn function within the experiment library. The library is accessible to the participants while coding their solution. The translation along the forward axis of the robot is proportional to the average of those two speed variables and scaled by a calibration factor that is determined from the slope of a linear fit between -75% and 75% motor speed in figure 1. For motor speeds over 80% the motors of the physical robot are not sufficiently strong to move the robot accordingly. Even with full batteries the speed levels off at motor speeds higher than 75% or below -75%. This is modelled in the simulation by limiting the maximum speed to 75% or -75% respectively if a higher amount is written in the code. It is verified by comparing the times that the physical and simulated robot need at different coded motor speeds to cross the long side of the white area. While pushing a cube the physical robot drives slower because the motors are not strong enough to fulfill the power demand required from the PID control. This occurs when pushing a cube with a desired motor speed above 20%. For motor speeds above 20% the robot speed is reduced by 5% of the coded motor speed.

When the robot is driving in a turn, i.e. when the speed state variables for each side are not equal, then the simulated robot orientation is rotated proportional to the difference between the two speed state variables. In case the absolute value of the speed state variables are equal but they have opposite signs then the robot is turning on the spot. The proportionality factor is determined from the slope between 0% and 60 % motor speed of the angular velocity plot of the physical robot shown in figure 2 and limited above motor speeds of 60%.

The motor speed in the simulation is programmed to change by a maximum of 9 percent points at the next FixedUpdate until it reaches the desired motor speed. In order to simulate the inertia of the physical robot and the acceleration limit was determined by qualitatively observing and comparing the behaviour of the physical and the simulated robot when changing from a rotation to a straight forward movement because it is difficult to accurately measure the acceleration of the physical robot otherwise.

Besides turning the robot on the spot by setting the two speed state variables, the library provided to the participant gives the possibility to turn the robot by a given amount of degrees and with a given rotation speed using the turn function from the library. In the simulation this is implemented using a coroutine that halts the participant code until the turn is completed. This means, similar to the physical robot, that the participant code cannot simultaneously execute another

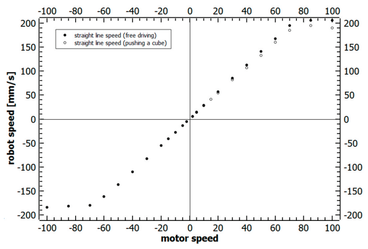


Figure 1: Motor speed as used in the participant code vs. physical distance driven per time for fully charged batteries, with and without pushing a cube.

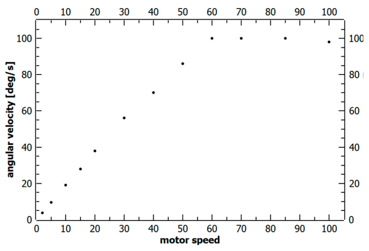


Figure 2: Angular velocity of the physical robot when having opposing speed state variables (rotating on the spot).

operation such as reacting to sensor inputs. In the simulation the turn is implemented by calculating the orientation where the turns need to stop and then calculating the target orientation for each FixedUpdate depending on the rotation speed. Then the turn coroutine sets the speed state variables to zero and the robot orientation is directly controlled by the coroutine and no longer by the code that is executed during FixedUpdate. However, the coroutine rotates the robot and is yielded after each rotation step until FixedUpdate has been executed to keep the turn synchronized with the physics simulation that occurs in FixedUpdate.

4.3 simulating sensors

The units used during measuring the sensor outputs of the physical robot are centimeters for distances and degrees for angles. The angles in the physical world and the simulation are equivalent and need to translation. However, the centimeters measured in the real world do not correspond to the distance units used in the Unity editor. One distance unit in the Unity editor corresponds to 9.49223 centimeter in the physical world. The measurement and fit functions shown below are in centimeters and the conversion number is used to translate from the real world to the simulated values.

4.3.1 Downwards reflection sensor

The simulated downwards reflection sensor is a child object of the robot in the Unity scene. This means it uses the robot object as its reference frame and therefore moves synchronous with the robot. It has a box collider surrounding it. It is used to detect entering or exiting the box colliders of the white area or the cardboard area. Whenever the box collider of the reflection sensor enters or exits one of those area box colliders, a state variable is set to the according

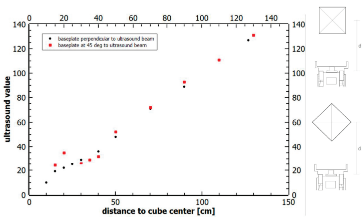


Figure 3: The plot shows the ultrasound value measured by the physical robot depending on the actual distance from the cube center. The sketches on the right side of the graph illustrate the cube orientation perpendicular to the ultrasound sensor (top) and at 45 degrees orientation (bottom).

reflective value that was measured with the physical sensor. The reflectivity value for the white area is measured and simulated at 47, cardboard at 36 and outside the cardboard at 16.

4.3.2 Ultrasound distance sensor

The simulated ultrasound sensor is, similar to the downwards reflection sensor, also a child object of the robot in the Unity scene that moves with the robot object. The simulated sensor value is estimated from the distance between the center of the ultrasound sensor object and the center of the cube. The conversion between the simulation units and the centimeter used in the physical world is made with the slope of the measurement shown in figure 3 for when the cube base plate is orientated perpendicular to the ultrasound sensor. For angles where the cube's base plate is oriented at 45 degrees to the forward direction of the sensor, the distance measurement is similar but the cutoff angles - this is the largest angle where the sensor can still detect the cube - is smaller. For a perpendicular orientation the cutoff angle is 35 ± 3 degrees whereas at 45 degrees it is 22 ± 2. We model a linear decline of the cutoff angle according to these numbers: $cutoff = -\frac{13}{45} \cdot \gamma + 35$ where γ is the angle between the forward direction of the ultrasound sensor and a vector perpendicular to the cube's base plate.

4.3.3 Blink light sensor

The blink value is determined from the forward pointing light / colour sensors on the left and right side of the robot. In the simulation those sensors are child objects of the robot object and move synchronous with the robot. The sensors on the physical robot detect the blink light by the light intensity difference between the times when the blink light is turned on and turned off. This requires time for the blink light to turn on and off again and causes a delay in the blink light measurement. In the simulation this is modelled by assessing delayed values from an array that stores the past blink values. In this simulation we use a delay of 500 milliseconds. The simulation, as well as the physical robot, evaluate the sensor measurement from both the left and right light sensor. The larger value is used as the blink value that is stored in the array and can be used in the participant code. The blink value for each light sensor is influenced by four factors:

1. Δ, the distance factor related to the distance d between the sensor and the blink light
2. β, the viewing angle factor related to the angle α between the forward sensor direction and the direction from the sensor to the blink light
3. ϕ, the relative cube orientation correction dependent on the angle φ between the direction from the sensor to the blink light and the direction perpendicular to the edge of a cube's base plate
4. δ, the relative cube orientation distance correction, the effect mentioned above is dependent on the distance d between the sensor and blink light

The current blink value in the simulation is calculated by:

$$blinkvalue = \Delta(d)\beta(\alpha) - \Delta(d)\beta(\alpha) \cdot \phi(\varphi)\delta(d)$$

The blink value neglecting the cube orientation towards the sensor is $\Delta(d)\beta(\alpha)$ but the actual blink value can be decreased due to the orientation of the cube. The cube orientation correction $\phi(\varphi)$ is furthermore dependent on the

MARCH 24, 2019

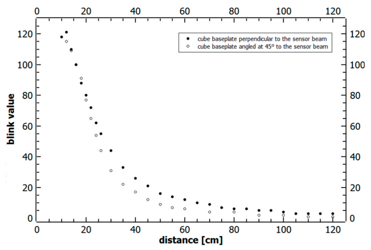


Figure 4: Blink values dependent on the distance from the sensor to the blink light in a cube for the blink light positioned straight ahead of the sensor. The black symbols represent the measurement for a cube orientation where the base plate of the cube is perpendicular to the forward sensor direction. The white symbols represent a cube's base plate oriented at 45 degrees to the sensor.

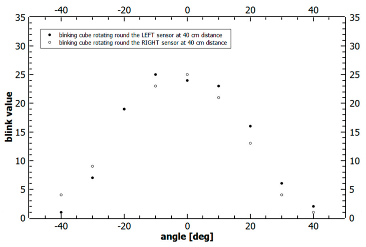


Figure 5: Blink values for the left sensor (black symbol) and the right sensor (white symbol) at a distance of 40 centimeters depending on the angle between the forward direction of the sensor and the direction to the blink light. The angle is negative if the blink light is positioned to the left of straight forward direction from the sensor. At all angles the cube's base plate is held perpendicular to the direction from the sensor to the blink light.

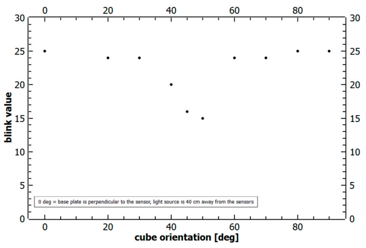


Figure 6: Blink values for a blink light inside a cube positioned 40 centimeters straight in front of the sensor for different orientations of the cube relative to the sensor. Angles of zero and ninty degrees represent a cube with a base plate perpendicular to the forward direction of the sensor. At 45 degrees the corner of the cube is pointing towards the sensor.

distance and corrected by multiplying $\phi(\varphi)$ by a factor $\delta(d)$. The product of $\phi(\varphi) \cdot \delta(d)$ is a relative correction and therefore must be multiplied with $\Delta(d)\beta(\alpha)$ to get the absolute correction.

The distance factor is determined from the measurements shown in figure 4. We use the measurement for the perpendicular orientation of cube's base plate relative to the forward sensor direction. The distance factor Δ is fitted with

$$\Delta = 3 \cdot 10^{-6}d^4 - 0.0011d^3 + 0.144d^2 - 8.55d + 200$$

This blink value based on the distance gets influenced when the blink light is not positioned straight in front of the sensor. The blink value decreases if the blink light is positioned towards the sides of the sensor. This effect is also measured at a sensor to blink light distance of 40 centimeters and normalized to the blink value of 24. The measurements for the different sensors on each side deviate slightly from one another and are shown in figure 5 and the normalized fits are:

$$\beta_{left} = \frac{1}{24}(8 \cdot 10^{-6}\alpha^4 + 7 \cdot 10^{-5}\alpha^3 - 0.0277\alpha^2 - 0.1567\alpha + 25.023)$$
$$\beta_{right} = \frac{1}{24}(7 \cdot 10^{-6}\alpha^4 + 7 \cdot 10^{-5}\alpha^3 - 0.027\alpha^2 - 0.0891\alpha + 26.005)$$

The uncorrected blink value can then already be calculated to $\Delta(d) \cdot \beta(\alpha)$ for each of the sensors. However, if the cube base plate is orientated at angles above 30 and below 60 degrees to the forward direction of the sensor the blink value decreases. This dependency is shown in figure 6 where an angle of zero or ninety degrees is equivalent to the base plate being perpendicular to the sensor direction. The relative correction is fitted from this data to:

$$\phi = \frac{1}{24}(24 - (0.0338 \cdot \varphi^2 - 3.0929 \cdot \varphi + 87.133)$$

The fit is accurate for angles between 30 and 60 degrees. Outside of this interval the simulation assumes no influence caused by the cube orientation. The influence of the cube orientation is distance dependent. This can be seen from the difference of the two curves in figure 4. We use the blink value difference of these two curves and the blink value difference between zero and forty five degrees cube base plate orientation from figure 6 to fit a correction $\delta(d)$ of the cube orientation correction.

$$\delta = 3 \cdot 10^{-9}d^5 - 10^{-6}d^4 + 0.0002d^3 - 0.0144d^2 - 0.482d - 4.6$$

We are aware that these polynomial fits are over-fitted outside of the domain that is relevant for the simulation. This is justified because we want to make a precise interpolation in the domain the sensor values are changing and we can program a constant for where the sensor values are no longer influenced.

4.3.4 Colour light reflection sensor

The sensor of the physical robot sends out coloured (red, blue and green) light and measures how much light is reflected back into the sensor. In the experiment and the simulation only the red reflection is used. The reflection effect cannot be estimated by using a single distance between the sensor and an object but is ideally simulated by reflection of infinitely many light rays. We tried and determined that it is possible to simulate the red reflection sensor by using 21 raycasts for each of the sensors equally spaced within the view wangle of the sensors to get a simulated value that is consistent and within the error margin of the physical sensors. The rays originate from the sensor location and measure the distance until they hit an object. The rays are sensitive to a polygon collider in the shape of the crossed walls of the physical cubes and a circular collider. The circular collider is added to decrease the distance from the sensor to the reflecting object because when the simulated sensor is directed at the cube center from a close distance the simulated value was smaller than the actually measured value. We cannot logically explain why this manipulation is necessary but we justify this by wanting to accurately imitate the physical sensor even if we do not understand why the one to one replication of the physical environment does not give the desired results.

The simulation is calibrated to the reflectivity measured with the physical robot and cubes. Figure 7 shows the red reflection values of each sensor for different distances between the sensor and the cube. The sensor is aligned towards the center of the cube. These curves are fitted and the fit formulas are used to estimate the reflectivity for each raycast for each sensor. For each of the sensors the values for all 21 raycasts are then averaged to get the overall value for this sensor. The fit function for the reflection of red light from a red cube for the left sensor is:

$$red_{left} = -0.0114d^3 + 0.8174d^2 - 19.52d + 158.94$$

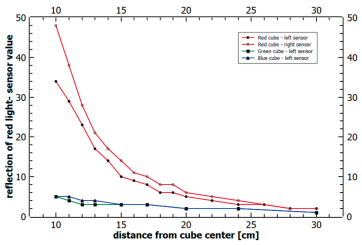


Figure 7: distance dependency of the reflection values of red light for a red, green and blue cube straight in front of the sensor. The distance is measured between the sensor and the cube center.

and the fit for the reflection of red light from a red cube for the right sensor is:

$$red_{left} = -0.0177d^3 + 1.244d^2 - 28.979d + 229.03$$

.

The measurements for the reflection of red light from green and blue cubes for both sensors are significantly smaller and we consider them similar within the measurement precision of the sensors. Therefore, we estimate them with one similar fit which is:

$$bluegreen = -0.1572(d - 5) + 5.5442$$

.

The calibration was tested by placing the physical robot in front of the cubes and validating this setup with the simulated values. Special focus was on when the cube fills the view angle of the sensor only partly, when the sensor points onto an edge of the cube walls and when the cube is locked in the pushing location central in front of the robot.

5 Verification of the simulation

5.1 Why and how we verify

The goal of the simulation is to have a more repeatable, accurate and time-saving tool for evaluating the performance of the programmed solution with a high number of runs rather than using the physical robot. The simulation can only be used as a substitute for running the physical robot if the simulated robot behaviour is an accurate enough representation of the physical robot's behaviour. Thus, the physical robot has to be well characterized in order to be the reference for the development and evaluation of the simulation. We will describe the three solutions that we use to validate the simulation. We describe how the physical and the simulated robot behave and compare them qualitatively and quantitatively. For the qualitative comparison the behaviour of the physical robot is judged from videos showing the trajectories and interactions with the cubes and the boundaries. For the quantitative comparison we answer the following questions:

1. How accurately can the physical robot repeat its performance under the same starting condition?
2. Does the simulation give the same results as the physical robot?

5.2 The test codes

Three solutions that reliably solve the task are chosen such that they include all sensors and functions necessary for detecting and removing the cubes.

5.2.1 ultrasound solution 1

This ultrasound based solution drives straight ahead whenever the ultrasound sensor detects an object within a distance of 120 centimeters or otherwise turns clockwise. The robot does not fall off the cardboard because the cubes, that are detected by the ultrasound sensor, are removed as soon as they are fully outside the white area. The physical cubes are removed by the experimenter and the digital cubes are programmed to disappear when the cube is outside the white area.

5.2.2 ultrasound solution 2

The second solution is also ultrasound based. It also turns clockwise until the ultrasound sensor detects object within a distance of 120 centimeters. Then it turns further to fully face the cube. Then it drives straight forward until it reaches the edge of the white area and continues driving for half a second longer to ensure that the cube is fully pushed outside of the white area. It then turns clockwise back towards the white area and continues searching cubes with the ultrasound sensor. This code does not react when it misses or looses a cube after detecting it.

5.2.3 blink solution

This solution uses blink lights in each cube to detect cubes. It turns clockwise in 20 degree steps until the robot measures a blink value that is larger than 13. After each turn the robot stops in this position for 1 second to allow the blink value to update to the correct value for this robot orientation. The robot aborts the search after 18 turning steps (one full rotation) without finding a blink light and then drives straight for two seconds in order to start a new search in a different area. If the robot detects a blinking light it drives straight ahead until it reaches the edge of the white area or it no longer detects a blink light in front of the robot for more than one second.

5.3 Qualitative results

When qualitatively comparing the behaviour of the physical robot with the simulated robot we start both scenarios with the same starting configuration and with the same code. We observe the trajectories of the robots and how the physical and simulated trajectory diverge, how the robots interact with the cubes and if they are removed in the same order and generally if there are any repeating patterns such as the robot getting stuck in corners or missing cubes after initially detecting them.

5.3.1 ultrasound solution 1

The physical robot turns until it detects a cube and then drives straight forward. If the robot is not directly heading towards the center of the cube it will eventually loose the cube out of the view angle of the ultrasound sensor and start turning again until the cube falls into the view angle again. Depending on the distance between the physical robot and the cube it performs one to three correction turn until it touches the cube. Theoretically we would expect many more corrections but we believe that the angular momentum of the robot when it turns leads to an overshoot so that the cube lies firmly within the view angle of the ultrasound sensor when the robot begins to drive straight forward. The overshoot occurs either because the robot's angular momentum makes it impossible for the robot to change directions abruptly or because a delay in the ultrasound measurement causes a delayed command to the motors to drive straight. We can replicate the overshoot of the physical robot in the simulation by both introducing a delay in the ultrasound measurement and by simulating a limited acceleration. Since the limited acceleration is also observable in the physical robot and we cannot identify a delay of the ultrasound measurement we implemented only the limited acceleration in the simulation. Both the physical and simulated robot remove the cubes in the same order and from the same quadrant of the white area in the three tested starting configurations. When cubes are located near the boundary and the robot is located on the opposite side of the playing field the detection range can be insufficient and the robot keeps turning infinitely. This occurs with the physical robot as well as the simulation.

5.3.2 ultrasound solution 2

The physical robot rotates until the ultrasound sensor detects a cube and then drives straight forward. For cubes more than eighty centimeters away it occurs that the robot does not fully turn towards the cube and misses it and keeps driving straight without corrections until it reaches the white area. In case no correction is needed, i.e. in general with short distances to the cubes this code performs more efficiently than ultrasound solution 1 because it approaches the cubes in a straight line and avoids driving an additional distance. However, we expect that in the average over many trials this solution is inferior because it looses a lot more time when it misses cubes. The simulated robot generally shows a similar behaviour. It also turns towards the cubes, detects them approximately at equal angles and occasionally misses cubes. However, it misses cubes far more seldom. So far we cannot explain this difference as it does not seem cube distance dependent and predictable. The different trajectories of the physical robot when restarting the robot under the same starting conditions can differ largely. This is because if the robot misses a cube and drives past it the situation is vastly different and the robot's reaction to the different situation is also very different. Instead of one repeating scenario we observe a range of different scenarios depending on if a cube was removed or not. The simulated trajectory qualitatively matches one of those scenarios. This ultrasound solution has similar issues with the ultrasound detection range both physically and in the simulation.

5.3.3 blink solution

The detection range using the blink lights is shorter than the detection range for the ultrasound sensor. The robot therefore often cannot find a cube when turning around and searching. The physical and the simulated robot most of the times detect and remove the same cube first but less reliably as for the previous codes. This means that the trajectories of the repetitions differ significantly depending on if a certain cube was detected or not. The simulation reproduces the scenario that was most common during the tests with the physical robot and is then usually accurate for the first cube and in many cases also the second cube. The simulated trajectory for removing the last cube was never similar to any trajectory obtained using the physical robot. The robot then continues to drive to a different location and starts searching again. When the cube is near the center it is far more likely to fall into the detection range than when the cube is near the boundary. The blink solution often moves cubes away from the center but does not fully move them outside the boundary. The cubes near the boundary are far less likely to be detected and in these cases the task completion times are significantly longer. This phenomenon occurs equally in the simulation and in reality.

5.4 Quantitative results

Ensuring that the simulation matches the behaviour of the physical robot qualitatively is a good starting point. If we want to use the simulation for quantitatively evaluating the performance of solutions we need to validate that simulation estimates the quantitiave result of the physical robot accurately. For this we start with measuring the repeatability of the physical robot by repeatedly executing the same scenario. We then execute the physical robot and the simulation from many different conditions and compare the results statistically.

5.4.1 How repeatable is the performance of the physical robot?

The three described solutions are evaluated with the physical robot from three different starting positions with up to 13 repetitions from each starting position. We show the mean time the robot needed to complete the task and the standard deviation over the N repetitions. Failed attempts are excluded from the statistics as they would imply an infinite task completion time. The failed attempts are caused by the robot getting stuck outside the cube detection range.

Table 1: Results of repeatedly executing three solutions with the physical robot from three different starting conditions

	start pos.	M	SD	N	failed
ultrasound 1	1	19.0	6.7	11	1
	2	22.6	4.3	12	0
	3	26.4	9.2	13	0
ultrasound 2	1	27.9	9.8	9	1
	2	26.6	5.7	10	0
	3	30.2	0.6	10	0
blink solution	1	89.9	45.8	7	0
	2	88.4	53.3	8	0
	3	110.8	98.5	8	0

5.4.2 Statistical comparison of the physical vs. the simulated robot

We measure the task completion time for the three solutions from twenty different starting configurations using the physical robot and the simulated robot. The starting configurations used in the physical evaluation and the simulation are equal and each starting position is run once. We present the mean task completion time and its standard deviation for each solution over those twenty runs. N describes the number of included measurements. If N is less than twenty this means that the solution failed $20 - N$ times. This is the case for the ultrasound based solutions and this happened when the nearest cube object was outside the ultrasound detection range when the robot was searching for cubes. These runs were aborted because they would lead to an infinite task completion time. Failed attempts are excluded from calculating the mean and standard deviation.

Table 2: Comparison of the completion times, physical vs. simulated robot for 20 starting positions

	solution	N	M	SD	% of phys. M
physical	ultrasound 1	18	20.5	2.92	100
	ultrasound 2	19	28.6	11.06	100
	blink	20	106.6	43.07	100
simulated	ultrasound 1	20	15.25	1.38	74.4
	ultrasound 2	19	20.28	3.54	70.4
	blink	20	84.21	72.58	78.4

For the initial placement of the robot for the twenty starting positions the robot's front edge is aligned along rectangular lines on the white area. This means that the initial rotation is along one of the four major directions and the locations are incremental. This setup is chosen such that the initial placement of the robot is easily repeatable. To check if we introduce a bias by selecting the starting configurations in this way we run the simulation from 99 randomly chosen starting positions and orientations. The random starting configurations are generated with a random number generator in the Unity software. Starting configurations where the robot is initially touching a cube are removed. The results are shown in table 3:

Table 3: Simulated task completion times for 99 randomly assigned, non-rectangular starting positions and orientations

solution	N	mean	std.dev.
ultrasound 1	99	15.16	1.06
ultrasound 2	99	20.86	7.37
blink	99	75.53	57.66

In the discussion we will argue that the measured task completion time is sensitive to perturbations. To investigate this we run the three codes from the previously used twenty starting positions and initially orient the robot one degree further counter-clockwise. We calculate the pairwise differences in task completion time between the non-perturbed and the perturbed starting orientation for each of the twenty starting configuration and then take the mean and standard deviation of the differences. The results are shown in table 4.

Table 4: Influence of one degree counter clockwise perturbed starting orientation on task completion time

solution	N	pert. mean	pert. std.dev.	diff. mean	diff. std.dev.
ultrasound 1	20	14.97	1.85	0.35	0.67
ultrasound 2	19	19.05	3.28	1.26	1.07
blink	20	64.00	32.20	42.57	59.32

Another imperfection of the physical robot is that the motors do not perfectly keep the rotation speeds that they are assigned in the code. To estimate the influence of inaccurate motor speed we simulate the three solutions from the previously used twenty starting positions with a one percent motor speed increase on the right motor. This means that the robot drives a visually unnoticeable left turn when it is originally programmed to drive straight.

Table 5: Influence on 1% faster motor speed of the right motor on task completion time

solution	N	pert. mean	pert. std.dev.	diff. mean	diff. std.dev.
ultrasound 1	20	14.89	2.1	1.07	1.03
ultrasound 2	19	20.00	4.34	2.23	2.51
blink	20	62.10	28.20	39.97	65.06

5.5 Discussion of the results

We want to use the simulation to compare the performances of code solutions written for the physical robot in a more time saving and more controlled way. The simulation needs to reproduce the results obtained with the physical robot at least relatively to compare different solutions. Most intuitively one tries to program a simulation that matches the reality as accurately as possible by comparing and improving the simulation to match reference cases taken with the physical robot. This entails that the physical and the simulated robot are deterministic and thus reliably repeat their respective behaviours when started under the same conditions. The simulation is deterministic but the physical robot shows significant deviations both in its trajectory as well as in the task completion time which we use as the main performance metric. Table 1 gives the task completion times of the physical robot for three different solutions from three different starting positions of the robot. We see that the deviations from the mean depend primarily on the solution and also on the starting position. For the ultrasound solution 1 the standard deviation of the task completion time is between 19% and 35% of the mean, for ultrasound solution 2 it is between 1,4% to 35% and for the blink solution it is between 50% and 89%. Apart from the one starting position with the low standard deviation for ultrasound solution 2 the deviations are too large to use the trajectories and task completion times as references for developing a simulation. As a consequence, despite the simulation being in itself accurately repeatable, it cannot evaluate a solution more accurately than how accurately the physical robot can repeat its performance, because we have no clear reference to optimize the simulation to. This means that a the simulation from one starting configuration, despite being numerically precise, has the same uncertainty as a single evaluation of this starting configuration with the physical robot. We develop the simulation by qualitatively observing the physical robot and the simulation and adjust the simulation until it qualitative shows the same characteristic behaviour, such as interactions with cubes and the boundary, in as many cases as possible. This approach aims to make the simulated behaviour qualitatively indistinguishable from the physical robot behaviour. However, this qualitative validation of the simulation is subjective and depends on the judgement of the experimenter looking at the video captures of the physical robot.

The large deviations of the physical robot originate from small perturbations that can neither be controlled for in physical measurements nor can the simulation replicate reality with more precision than the perturbations that cause the deviations. Therefore, a single measurement of the physical or the simulated robot is not a good indication for the overall performance of the solution. However, we assume that these perturbations are normally distributed and their effects can be mitigated by repeatedly executing and evaluating the robot from many starting conditions and performing statistics. To illustrate this approach we measure and simulate the task completion times for the three solutions from twenty different starting positions. The measured and the simulated task completion times seldom match pairwise for a single starting position but the question is if the results match relatively when comparing the means of the different solutions. The results are shown in table 2. Firstly we see that the means of the simulated task completion time are between 70% and 78% of the measured means task completion times, i.e. the simulation does not absolutely give the same results as the measurement with the physical robot. For comparing the performance of different solutions it is sufficient that the simulation gives the same result when relatively comparing the task completion times. We compare the three solutions by normalizing the means and standard deviations by the task completion time of the blink solution, which was the slowest of the three solutions. The means and standard deviations measured with the physical robot are normalized to the physically measured blink solution mean and the simulated results are normalized to the simulated blink solution result. The normalized task completion time mean for the physically measured ultrasound solution 1 is 19,3% of the blink solution task completion time with a standard deviation of 2,7% whereas this comparison for the simulated case gives a normalized mean of 18,2% with a normalized standard deviation of 1,6%. These normalized means are well within each others standard deviations and are considered equal within the measurement accuracy. The same is the case for ultrasound solution 2 with a normalized mean of 26,9% and a normalized standard deviation of 10,4% for the physical evaluation and a normalized mean of 24,1% and a normalized standard deviation of 4,2% for the simulated evaluation. The same conclusion is true when normalizing to other values. This means that we can compare different solutions relatively without needing a simulation that is absolutely accurate. The twenty starting positions for the robot that were used for this evaluation were chosen such that they can be reliably reproduced in the real world and replicated as similar as possible in the simulation. This means that the starting positions were reasonably chosen to cover a wide range of different starting positions and orientations. It can be that the selection caused a bias in the setup. To ensure that this is not the case we simulated the three solutions from 99 randomly chosen starting positions. The results are in table 3. The task completion time means for the simulation with 20 chosen and 99 random starting positions differ by less than 1% for ultrasound solution 1, by less than 3% for ultrasound solution 2 and by about 10% for the blink solution. The increase in relative difference can be explained with the accompanied increase of the standard deviation of these solutions. This shows the importance of a large sample size for solutions with a large standard deviation. The differences between the simulations are significantly less than their standard deviations (and standard errors) and we conclude that the chosen twenty starting positions do not show a significant bias.

During the observation of the physical robot we saw that despite starting the same code and placing the cubes and the robot as similarly as possible at the same positions for each repetition we see that the robot initially drives very similar trajectories that diverge from each other over time and some events cause a drastically different trajectory afterwards. These drastic differences between trajectories occur when one trajectory removes a cube while the other only moves or rotates the cube without removing it or even misses the cube entirely. Another scenario is when the robot is programmed to react when it reaches and detects the boundary of the white area and the trajectories are almost aligned with the boundary. An initially small perturbation in the location or orientation of the robot then has a large influence on where it detects and reacts to the boundary causing a larger deviation of the trajectories. Both scenarios make the system sensitive to small differences in robot location and orientation. To illustrate this sensitivity and investigate the effect of a single perturbation under controlled conditions we use the previous twenty starting conditions to simulate the task completion times when the initial robot starting orientation is changed by 1 degree counter clockwise. The mean task completion time from the unperturbed starting positions in table 2 and the mean task completion time from the counter clockwise rotated starting positions in table 4 are similar within their uncertainty limits. We cannot distinguish a significant effect on the mean task completion times caused by perturbing the starting positions. We then calculate the absolute pairwise difference between the task completion times of the unperturbed and perturbed simulation for each of the twenty starting positions. The mean of these absolute differences and the standard deviation thereof is also shown in table 4. We see that the misalignment of the robot by one degree already explains $0.35/1.38 = 25\%$ of the standard deviation observed over the twenty different unperturbed starting positions for ultrasound solution 1. For ultrasound solution 2 the misalignment accounts for 36% of the original standard deviation and for the blink solution the perturbation causes an absolute difference mean that is 59% of the standard deviation. We repeated the same idea of comparing perturbed and unperturbed robot behaviour with increasing the motor speed of the right motor by 1% point. The results are shown in table 5 and the mean task completion time is similar to the unperturbed result within the measurement accuracy. From the pairwise differences in task completion time for each starting position and its mean we see that it explains 78% of the unperturbed standard deviation of ultrasound solution 1, 63% for ultrasound solution 2 and 55% for the blink solution. Both perturbations have a significant influence on the task completion time for a single evaluation from a single starting position. These perturbation are very likely to occur when executing the physical robot and are both undetectable and therefore impossible to control for. This explains why we observe significantly varying results when repeating the physical measurement under conditions that appear similar but are in fact perturbed. This makes it impossible to replicate the physical conditions in the simulation with a precision that has a negligible effect on the trajectory. Evaluating the performance of different solutions from a single measurement is not possible. Evaluating the performance requires repeated measurements and statistics. Manually executing and measuring the physical robot often enough to get a reliable statistic result can in principle be done but is very time consuming and therefore not feasible. Additionally, can the simulation keep the conditions absolute repeatable whereas an evaluation with the physical robot is subject to drainage of the batteries, wear and tear of the mechanical parts and changes of the surrounding conditions such as light and temperature that cause perturbations that have significant influences on the repeatably as illustrated above. These influences make a comparison of different solutions under similar conditions impossible and the inaccuracy introduced by not perfectly matching the behaviour of the physical robot within the simulation is far less significant than the inaccuracy introduced by changing conditions. Another advantage of the simulation is that it is very time saving, allows to generate more results from more starting positions and therefore allows a statistical analysis with a lower uncertainty.

5.6 Conclusion

We develop a simulation to a self-built LEGO Mindstorms robot. We aim to use the simulation to compare the performance of different code solutions programmed to solve a task that involves finding and moving cubic objects. We replicate the physical environment digitally and measure the behaviour of the physical robot to make a mathematical model that we use to implement a simulation. The simulation is primarily based on these measurements and then adjusted from qualitative comparisons between the physical and the simulated robot behaviour. After adjusting and verifying the simulation for the individual sensors and actuators we verify the simulation by comparing the simulated robot behaviour with the behaviour of the physical robot in a systems approach with three solutions that cover the entire scope of available functions combined in solutions that solve the task. We conclude that the trajectories of the physical robot deviate significantly despite being repeatedly executed under as similar as possible conditions as possible. The physical robot behaviour is not deterministic. This implies that the simulation cannot accurately replicate the trajectory of the physical robot when the physical robot does not have a consistent trajectory. We verify that the simulation qualitatively shows a similar behavioural characteristic as the physical robot. For quantitatively comparing code performances a repeated evaluation from different starting conditions and a statistical analysis is required. We compare the task completion times of the physical and the simulated robot for twenty different starting conditions. We show that the simulation does not absolute give the same result as measured physically but it is relatively accurately when comparing different solutions. We ensured that the starting conditions used for the quantitative verification are unbiased. We argue that the sometimes drastically different trajectories and task completion times of the physical robot are caused by a sensitivity of the system to small perturbations of the robots position and orientation. We use the simulation to demonstrate that likely inaccuracies such as a change of the robot's initial orientation by one degree or a one percent speed difference in the motor speed can already explain the standard deviation calculated from the task completion times of the twenty starting conditions. Overall we conclude from these results that this simulation is needed and a useful tool for relatively comparing solution performances under controlled conditions with sufficient data for statistical methods.

References

[1] Achim Gerstenberg and Martin Steinert. Open ended problems - a robot programming experiment design to compare and test different development and design approaches. In *DS 91: Proceedings of NordDesign*. Design Society, 2018.

[2] Achim Gerstenberg. robotdatasheet.pdf in robotexpsetup on github, May 2018. `https://github.com/AchimGerstenberg/RoboExpSetup/blob/master/printouts/robotdatasheet.pdf` (accessed 24-March-2019).

[3] Unity Technologies ApS. Unity version 5.6.1f1, May 2017. `https://unity.com/` (accessed 24-March-2019).

C.8 Contribution 8: Evaluating and Optimizing Chaotically Behaving Mobile Robots with a Deterministic Simulation

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

29th CIRP Design 2019

Evaluating and Optimizing Chaotically Behaving Mobile Robots with a Deterministic Simulation

Achim Gerstenberg*[a], Martin Steinert[a]

[a]Norwegian University of Science and Technology, Department of Mechanical and Industrial Engineering, Richard Birkelands vei 2B, 7491 Trondheim, Norway

* Corresponding author. Achim Gerstenberg. *E-mail address:* achim.gerstenberg@ntnu.no

Abstract

Testing and comparing prototypes is essential during the development of mobile autonomous robots. These robots often behave chaotically, i.e. their behavior is sensitive to small perturbations and their performance is unpredictable. An evaluation and comparison of solutions cannot be based on a few test runs. We present an approach that uses a digital simulation to acquire the data necessary for statistical comparison. The simulation can be sped up and automated and thus allows the developers to compare solutions more quickly than physical tests can.
While it is not possible to replicate a test setup in the physical world with sufficient accuracy, the simulation is deterministic and can be used to investigate the dependency of a specific perturbation.
We illustrate this approach using three solutions to an example task where the robot needs to remove objects from an area in the shortest time possible. When executing each solution from 99 randomly chosen starting conditions an average performance difference of as little as 2% was distinguished with statistical significance. Differences in performance under an environmental perturbation such as a slight misaligned of the robot's starting orientation or a hardware perturbation of imprecisely controlled motor speeds are also found to be significant for some solutions showing how the approach can be used to test robustness of solutions to perturbations.
Robot performance is dependent on the interplay between the algorithm that controls the robot, the robot hardware and its environment. By testing different algorithms and subjecting them to perturbations of the environment and the robot's hardware, it is possible to optimize the algorithm-hardware combination for the least restrictive hardware requirements necessary while still ensuring the desired performance. In this paper we offer a quantitative approach for iterative development of the algorithm-hardware combination of mobile autonomous robot solutions.

Keywords: mobile robot; chaos; performance evaluation, requirement; simulation

1. The challenge of evaluating autonomous mobile robot solutions

Mobile autonomous robots play an increasingly important role in today's life. Self-guided lawn mowers cut the grass in the garden, vacuum cleaners clean the floors and drones inspect power lines. The actions of these mobile autonomous robots are based on algorithms and can be chaotic [1,2]. This means that small perturbations of the robot lead to large deviations in the robot's behavior over time. External perturbations, such as air movement acting on a drone, and perturbations that are inherent to the robot, like measurement errors of the robot's sensors or mechanical wear and tear of the robot's actuators can often not be avoided or minimized to a degree that the system becomes stable within the desired time frame. Lyapunov exponents and the error growth factor are tools used to characterize how chaotic a system is and how long it remains predictable [3,4]. Although predictability often is desired, it is not necessarily a reasonable measure of performance regarding evaluation of the performance of a solution in terms of solving the desired task. Sometimes a chaotically behaving robot may be superior to a stable and predictable robot behavior. In the development process, testing and comparing prototypes is essential for making design choices [5-8] and the Nation Institute for Standard and Technology has developed physical testing grounds for mobile autonomous robots in urban search and rescue settings [9]. However, quantitative evaluations of such robot behaviors are difficult because the behavior fluctuates unpredictably. Evaluating solutions from single or few test runs is inevitably statistically insignificant when unavoidable perturbations have major influences on the test results. To quantitatively compare solutions, a statistical evaluation with sufficiently many repeated tests and precise data capturing is necessary. The number of tests needed to find a significant performance difference between two solutions depends on the actual difference of performance, how precisely we can measure the performance, how consistently the robot repeats this performance and how many uncontrolled sources of perturbations influence the robot. More uncontrolled perturbations require more tests, as the performance fluctuations average out. Acquiring this amount of data is tedious and time consuming.

2. Solving this challenge with a deterministic simulation

We want to find a way to measure the robot's performance precisely, offer control over perturbations and automate the measurements for convenience.

This is achieved with a digital, deterministic simulation. It simulates the robot's behavior in the given modelled environment, precisely documents the resulting performance and can be programmed to automatically repeat the measurements often enough to obtain a statistically useful result.

The simulation is deterministic, meaning that if it is started twice under similar conditions it gives equal results. These results may however differ from a physical measurement that tries to replicate the same conditions. This does not imply that the simulation is faulty, but that the system is sensitive to perturbations that are so small that they become difficult to accurately replicate in the simulation and vice versa. In other words, the significant perturbations are smaller than what can be measured and thus replicated in the simulation. The simulation can therefore not be quantitatively validated by comparing a single simulated robot behavior with a single physical measurement. It needs to be validated qualitatively by observing behavioral patterns and quantitatively by repeatedly executing the physical robot and statistically comparing the physical results with the simulated results. The simulation needs to be validated such that it gives correct relative results when comparing different solutions. This means that if a solution performs in average over many simulations twice as good then it also needs to perform twice as good in average over many physical measurements. Although this requires repeated physical measurements this process is still significantly time saving if afterwards the simulation is used to compare many more solutions. In cases of popular and often well documented robot platforms a proven simulation software already exists (for example for mobile educational, animal-like, humanoid and planetary exploration robots [10]).

An additional advantage that exists only with the simulation is that perturbations can be controlled. This means that the system can be tested while avoiding all perturbations but one allowing the developers to test the robustness of a solutions to a single type of perturbation. For example, does the performance change significantly if a sensor is less accurate and feeds the control algorithm with more fluctuating measurement values or if the speed of an actuator changes. This allows the developers to make better design choices when choosing hardware requirements and thereby save costs.

3. The example setup

We illustrate the presented approach with examples. We first briefly explain the robot and the task that it is supposed to solve and then use this example to illustrate the accuracy of the approach for comparing different solutions and how to use it for determining the robustness of a solution to a single type of perturbation.

3.1. Task and robot

The task is to program the robot to autonomously remove three objects from a 1.8 square meter rectangular area within the quickest time possible after starting the robot. The robot is supposed to complete the task from any starting position within the rectangular area. This white rectangular area is surrounded by a darker 17 centimeter wide frame. The robot has a downwards pointing reflection sensor that can detect the decreased reflectivity of the darker frame. The entire construction is elevated by 3 centimeters, which means that if the robot falls off it cannot get back and the task cannot be completed. The removable objects are cube-shaped with reflectors for ultrasound waves. The robot has a forward aiming ultrasonic distance sensor to detect those cubes. Furthermore, radially outwards shining blinking lights can be placed in the middle of those cubes. The robot has on each side a forward pointing light sensor for detecting the intensity of those blinking lights. The robot drives by controlling the speed of a motor that drives a belt on each side making it possible to drive forward, backwards and turn. A detailed description of the task and the robot can be found in [11] and [12].

3.2. The simulation

The simulation is based on a 2D top view digital replication of the physical setup in Unity. It was developed by replicating the environment to scale and individually measuring the physical robot behavior for each sensor and actuator, interpolating the measurements and implementing the corresponding fit into the simulation. The simulation executes the different algorithms that control the robot from 99 different starting positions of the robot and documents the task completion time as the measure of performance.

The simulation we use in this publication is verified by qualitative and quantitative comparison with measurements of the robot completing the task in the real world.

The development details and verification of the simulation can be found in [13].

3.3. Example algorithms

To illustrate how to compare several solutions we use three different algorithms, two of which use the ultrasonic sensor to detect the cubes and the third uses the blinking lights as beacons.

Ultrasound solution 1: This ultrasound-based algorithm makes the robot drive straight ahead whenever the ultrasound sensor detects a cube within a distance of 120 centimeters or otherwise turns the robot clockwise. This distance evaluation is done continuously and when the robot aims to drive past the cube the robot corrects instantly and turns towards the cube. The robot does not fall off the cardboard because the cubes that are detected by the ultrasound sensor are removed as soon as they are fully outside the white area.

Ultrasound solution 2: The second solution is also ultrasound based. It also turns clockwise until the ultrasonic sensor detects a cube within a distance of 120 centimeters. When the cube enters the acceptance angle of the ultrasonic sensor the robot begins to drive straight ahead until it reaches the edge of the white area regardless if it has collected the cube or not. After reaching the edge, the robot continues driving for half a second longer to ensure that the cube is fully pushed outside of the white area. It then turns clockwise back towards the white area and continues searching cubes with the ultrasonic sensor.

Blink solution: This solution uses blink lights in each cube to detect cubes. It turns clockwise in 20 degree steps until the robot measures a blink value that is larger than a threshold. Measuring the blinking light intensity requires that the light sensor recognizes a full blinking period (light on, off and on again). After each turn, the robot stops in this position for 1 second to allow for an accurate intensity measurement. The robot aborts the search after 18 turning steps (one full rotation) without finding a blinking light and then drives straight for two seconds in order to start a new search in a different area. If the robot detects a blinking light it drives straight ahead until it reaches the edge of the white area or it no longer detects a blink light in front of the robot for more than one second.

4. How the simulation is applied

In the following sections, we apply the general concept explained abstractly in the beginning of this paper to the three example algorithms. We execute the three algorithms in the simulation and document the task completion times. Each algorithm is simulated 99 times from different starting positions. Those starting positions are such that the robot is fully within the white rectangular area and not already touching any cube. The starting x, y positions and the orientation angle of the robot are generated with a randomizer to avoid an involuntary bias that may be introduced by manually selecting the starting positions. Every algorithm is executed from the same 99 starting configurations to guarantee comparison under equal conditions. The simulation is verified to give similar results for simulation speeds of up to 60 times the original simulation speed. This means that, depending on the performance of the algorithm, these 99 simulation runs can be executed in between half a minute to 10 minutes. If those runs were executed manually with the physical setup this would take between several hours to a full day and the manual reset of the starting position is a considerable time loss.

5. Comparing the performance of the three example algorithms

Table 1 shows the sample size N - that is the number of times the algorithm completed the task and a task completion time could be documented -, the mean task completion time and the standard deviation of the task completion time.

Table 1. Descriptive statistics of the task completion times for the three example algorithms

Evaluated algorithm	N	M	SD
Ultrasound solution 1	99	15.16	1.06
Ultrasound solution 2	99	22.8	8.9
Blink light solution	97	68.9	34.5

We observe that ultrasound solution 1 in the mean outperforms ultrasound solution 2 which again outperforms the blink solution. Another performance indicator is that the blink solution only completed the task in 97 out 99 simulation runs. The question now is if these results are statistically significant. We therefore perform an independent sample t-test that compares each of the algorithm's performances. The t-value comparing ultrasound solution 1 with ultrasound solution 2 is t(196) = 8.48. This results in a highly significant p value of less than 0.0001. The task completion time of ultrasound solution 1 is statistically significantly lower than the task completion time of ultrasound solution 2. The t-value comparing ultrasound solution 1 with the blink solution is t(194) = 15.5. This results in a highly significant p value of less than 0.0001. The task completion time of ultrasound solution 1 is statistically significantly lower than the task completion time of the blink solution. The t-value comparing ultrasound solution 2 with the blink solution is t(194) = 12.6. This results in a highly significant p value of less than 0.0001. The task completion time of ultrasound solution 2 is statistically significantly lower than the task completion time of the blink solution.

This shows how different solutions can be compared statistically and that the method can give quantitative answers to the expected and qualitatively observed performance differences of the task solutions.

In the three examples we expected the following performance differences from the qualitative robot behavior observations.

Ultrasound solution 1 approaches the cubes in a right turn because it always aims slightly to the left of the cubes and does a corrective right turn when it misses it. This means that the solution drives a slightly longer distance than the "perfect solution" that aims precisely at the cubes and then minimized the distance driven by driving straight. However, ultrasound solution 1 works very reliably and never misses or loses a cube. The standard deviation of the task completion time is therefore small and the variation only depends on the distance driven which is determined by the starting position of the robot.

Ultrasound solution 2 has the potential to complete the task faster because it orientates the robot towards the cube and then drives straight. It thus drives a shorter distance than ultrasound solution 1 if it does not miss or lose a cube. However, we observe than this solution often does so and then needs to approach the cube again. This leads a larger deviation of task completion times and also causes a longer mean task completion time.

The sensitivity for measuring the blinking lights is not sufficient to achieve a detection range that is as long as the ultrasonic detection range. Therefore, the robot often searches for the blinking light in the cube but cannot detect it and needs to change its location hoping to detect a light from the new position. Furthermore, a precise measurement of the blinking light intensity takes about one second. This means that the robot can only turn slowly while searching for a blink light. These two factors make this blink solution very slow.

6. Resolution of the approach

The three examples we compared perform very differently and we can predict those differences already from observing the qualitative behavior and this extensive quantitative evaluation was not necessary to determine which solution solves the task the fastest. Now we want to find out how small a differences in task performance can be and is still detectable. For this we execute ultrasound solution 1 with a 2% decreased maximum speed of the robot while driving straight ahead. Qualitatively the robot behaves the same and executes a similar trajectory but does so slightly slower. Is the resulting increase in task completion time measurable?

The slower robot solved the task in all 99 simulation runs with a mean task completion time of 15.54 seconds and a standard deviation of 1.13 seconds. The comparison of the task completion time means between ultrasound solution 1 with full motors speed and decreased speed leads to a t-value of t(196) = 2.24 and this corresponds to a p-value of 0.016. This means that a 2% decrease in motor top-speed leads to a statistically significant decrease in mean task performance time which is detectable with this approach.

7. Quantifying the influence of minimally rotated starting positions

We now evaluate how robust the example solutions are to perturbations of the starting positions. We use the starting positions from before but rotate the starting orientation of the robot by one degree counterclockwise. By comparing the mean task completion times with the unperturbed mean task completion time we can estimate if the perturbation causes a significant difference. In addition, we calculate the absolute pairwise difference between the task completion time from each unperturbed simulation run with the corresponding perturbed simulation run. The average of this absolute pairwise difference gives a measure for the sensitivity to perturbations and how chaotic the system reacts (table 2).

Table 2. Descriptive statistics of the task completion times for the three example algorithms after rotating the robot's starting orientation one degree counterclockwise.

Evaluated algorithm	N	M	SD
Ultrasound solution 1	99	15.22	1.16
Ultrasound solution 2	99	22.6	9.7
Blink light solution	97	70.4	42.1

The results of the t-tests are shown in table 3. None of the examples shows a significant sensitivity of the average task completion time to the perturbation.

Table 3. Comparative statistics comparing the mean task completion times.

Compared algorithms	df	t-value	p-value
Ultra1 vs. ultra1 rotated	196	0.38	0.70
Ultra2 vs. ultra2 rotated	196	0.15	0.88
Blink vs. blink rotated	192	0.27	0.79

This is what we expect for the mean task completion time because, even if the system reacts chaotically to this perturbation, in average the consequences of the perturbation average out over 99 simulation runs. In other words, if 99 starting positions are sufficient to accurately determine the performance of a solution then 99 altered starting positions must give the same result for the same solution.

From the t-tests we know that the mean task completion times do not differ significantly. However, to illustrate that perturbations can change the outcome of a single simulation run we calculate the absolute value of the task completion time difference between the simulation from the unperturbed robot starting positions and the time from the same starting position with a one degree counterclockwise perturbation. The mean of these absolute pairwise task completion time differences in relation to the mean task completion time is a measure for how sensitive the solution is to this type of perturbation. We observe that the mean absolute difference of the task completion times is about 4 times the amount of the difference in mean task completion times for ultrasound solution 1. It is 5 times as large for ultrasound solution 2 and 42 times as large for the blink solution. Another way to determine how chaotically a solution reacts to the perturbation is by calculating the Pearson correlation between the unperturbed and perturbed simulations.

We see that the blink solution that already shows the large relative absolute pairwise difference also shows a low correlation of the task completion times between the unperturbed and perturbed simulation runs while the two ultrasound solutions show a strong correlation. Those results are shown in detail in table 4.

Table 4. Pairwise comparison of the simulation results of the unperturbed and by 1 degree perturbed starting positions of the robot.

Compared algorithms	N of compared pairs	Difference of means	Mean pairwise difference
Ultra1 vs. ultra1 rotated	99	0.06	0.23
Ultra2 vs. ultra2 rotated	99	-0.23	1.22
Blink vs. blink rotated	95	0.55	22.8

Since ultrasound solutions 1 and 2 turn clockwise until the robot detects a cube, one degree counterclockwise rotation of the robot's starting orientation makes little difference to the robot's behavior. This explains the very high correlation for the pairwise comparison between unperturbed and perturbed task completion times for both ultrasound solutions 1 and 2.

In contrast, the blink solution searches by turning the robot in 20 degree increments. The perturbation is not negated soon after starting the robot and propagates leading to diverging trajectories and missing a cube happens frequently. While for example the unperturbed simulation run removes the cube, the perturbed simulation may only touch and spins the cube which leads to a qualitatively very different trajectory afterwards. Furthermore, the blink solution reacts when it reaches the edge of the white area. When driving almost parallel to the edge, a small angular difference has a large impact on where the sensor detects and reacts to the edge of the white area resulting a largely varying trajectories. Although these differences average out in the mean task completion time over 99 simulation runs, they cause a large difference when pairwise comparing the unperturbed and perturbed task completion times and the two task completion times no longer correlate.

When evaluating the performances with the physical robot it is not possible to manually place the robot with such accuracy that a measurement can be repeated with a sufficiently similar robot placement when starting the robot. By determining the influence a one-degree misalignment of the robot can have on the outcome of the measurement we demonstrate the infeasibility of manually comparing individual measurements obtained with the physical robot.

We replicate the setup physically and execute the same algorithm repeatedly with the physical robot. The robot qualitatively behaves similarly in each repetition but the task completion time varies between repeated measurement despite trying our best to replicate the test conditions as precisely as possible. This confirms that the solutions are indeed sensitive to uncontrollable perturbations and studying the influence of such perturbations requires a deterministic simulation.

8. Quantifying the robustness under imprecise motor speeds

The approach of introducing one well-defined perturbation into the simulation cannot only be used to determine which influence an environmental perturbation has on an algorithm but also how changes in hardware influence the robot's behavior.

For example, we can introduce motors that do not precisely keep their speeds and quantitatively evaluate the consequences for robot performance for each of the example solutions. In this scenario the maximum speed of the left motor is decreased by 2% and the maximum speed of the right motor is increased by 2%. This means that the robot drives a slight left turn when it is programmed to drive straight at full speed. The simulation results for this are shown in table 5.

Table 5. Descriptive statistics of the task completion times for the three example algorithms with imprecise motor speeds.

Evaluated algorithm	N	M	SD
Ultrasound solution 1	99	15.01	1.48
Ultrasound solution 2	98	26.42	8.20
Blink light solution	99	82.50	46.4

We expect that ultrasound solution 1 remains mostly unaffected by this motor speed perturbation because if the slight left turn causes the robot to drive past one of the cubes the algorithm immediately recognizes this and adjusts the robot's orientation back towards the cube. Ultrasound solution 2 does not correct the robot's orientation after initially detecting and aiming at a cube. At a larger robot-to-cube distance the slight left turn caused by the perturbed motor speeds makes the robot drive past and miss the cube needing to approach it again and thus taking longer to complete the task. The blink solution has a similar issue. Therefore, we expect ultrasound solution 2 and the blink solution to perform worse after introducing imprecise motor speeds. Table 6 shows the independent sample tests that compare the task completion times under unperturbed conditions with the results with imprecise motor speeds.

Table 6. Comparative statistics comparing the mean task completion times of the unperturbed versus the results with imprecise motor speeds.

Compared algorithms	df	t-value	p-value
Ultra1 vs. ultra1 imprecise	196	0.82	0.41
Ultra2 vs. ultra2 imprecise	195	2.97	0.0034
Blink vs. blink imprecise	194	2.32	0.0211

We can confirm our assumption that introducing imprecise motor speeds does not cause a significant difference in task completion time for ultrasound solution 1 but significantly increases task completion time for ultrasound solution 2 and the blink solution.

Simulations like these can help finding appropriate requirements and can be cost saving by finding the lowest fidelity requirements necessary to satisfy the needs. In this case, having unmatched motor speeds does not significantly change the performance of ultrasound solution 1 while it worsens the performance of ultrasound solution 2 and the blink solution. This means that the requirements on motor speed reliability are stricter for the latter two solutions or - vice versa - the better solution allows motors with lower specification.

9. Conclusion and Outlook

Autonomous mobile robots behave chaotically. Small perturbations to the robot's state have large implications on its trajectory and thus also on how it performs a task. If one wants to compare the performance of different solutions to the task it is necessary to accurately replicate the test conditions. In the physical world this is usually not possible as even small deviances like misplacing the robot by one degree causes different test results. In order to achieve a valid comparison of solutions it is necessary to collect data from repeated measurements until the fluctuation caused by uncontrolled perturbations average out. Collecting this data can be inconvenient and time consuming. If those measurements can be made deterministic by perfectly replicating the test conditions then a desired perturbation can be introduced on purpose to study its effect on the robot's performance. A digital simulation is deterministic and offers the possibility to control and introduce perturbations and allows the robot developers to automate and speed up the measurements considerably. Since the robot behavior is chaotic, this simulation can never accurately replicate the behavior of the physical robot in a single run but it can give valid results when comparing the mean performance of solutions. We used this approach to compare three different solutions and quantify their performance in an example task. We determined a mean of a performance metric and used t-tests to estimate statistical significance. We were able to observe a statistically significant performance difference between two solutions where we know that the performance difference is 2%. The resolution is however dependent on the standard deviations and thus on how consistently the solutions performs. Although this method can give fairly accurate performance comparisons and gives hints where to look, it does not, by itself, give qualitative insights and observing the robot's behavior is still essential to make improvements. However, it allows the developers to quantitatively verify their assumptions.

In addition, the simulation cannot only plainly compare different solutions under a set of given conditions but can also alter these conditions and evaluate how robust a solution is to those perturbations. We demonstrated this by introducing an initial rotation to the starting orientation of the robot and by introducing a motor imprecision making the robot drive in a left turn when the algorithm assumes the robot to drive straight ahead.

Not surprisingly, a solution that uses more frequently updated measurements and adapts constantly showed more robustness to these perturbations than solutions that rely on less frequent measurements.

This means that the simulation is useful to optimize the interplay between the algorithm that controls the robot and the robot hardware and helps finding the least restrictive, and therefore often cheapest, hardware requirements necessary to still ensure the desired performance.

For example, the solution with frequently measurements can have less restrictive hardware requirements for the speed regulation of the motors.

In the future we envision that the development of the algorithm - hardware combination can be automated by using genetic algorithms that use the performance metric as feedback for further iterations.

Acknowledgements

This research is supported by the Research Council of Norway through its user-driven research (BIA) funding scheme, project number 236739/O30S. Thanks to Letizia Jaccheri for providing useful feedback while writing this paper.

References

[1] Nehmzow U, Walker K. The behaviour of a mobile robot is chaotic. AISB journal 2003;1:373-388.

[2] Nakamura Y, Sekiguchi A. The chaotic mobile robot. IEEE Transactions on Robotics and Automation 2001;17:898-904.

[3] Wolf A, Swift JB, Swinney HL, Vastano JA. Determining Lyapunov exponents from a time series. Physica D: Nonlinear Phenomena 1985;16:285-317.

[4] Nehmzow U. Quantitative analysis of arobot-environment interaction – on the difference between simulation and the real thing. Proceedings of Eurobot 2001.

[5] Schrage M. The culture(s) of prototyping. Design Management Journal 1993;4:55-65.

[6] Smith RP, Tjandra P.Experimental observation of iteration in engineering design. Research in Engineering Design 1998;10:107-117.

[7] Dow S, Heddleston K, Klemmer SR.The efficacy of prototyping under time constraints. Proceedings of the Seventh ACM Conference on Creativity and Cognition 2009; 165-174.

[8] Pahl G, Beitz W, Feldhusen J, Grote KH. Engineering design: A systematic approach. 3rd ed. London: Springer 2007.

[9] Jacoff A, Messina E, Evans J. Performance evaluation of autonomous mobile robots. Industrial Robot: An International Journal 2002;29:259-267.

[10] Robot models included in the Webot simulation software. https://cyberbotics.com/doc/guide/robots. Accessed: 2019-02-09

[11] Gerstenberg A, Steinert M. Open ended problems – a robot programming experiment design to compare and test different development and design approaches. Proceedings of NordDesign 2018

[12]Gerstenberg A. RobotExpSetup on GitHub. 2018. https://github.com/AchimGerstenberg/RoboExpSetup (retrieved March 29th, 2019.

[13] Gerstenberg A, Steinert M. Development and verification of a simulation for leveraging results of a human subjects programming experiment. arXiv eprint: 1903.10420. March 2019

C.9 Contribution 9: The Relevance of Testing in Engineering Product Development Investigations on a Robot Programming Task

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

29th CIRP Design 2019 (CIRP Design 2019)

The Relevance of Testing in Engineering Product Development - Investigations on a Robot Programming Task

Achim Gerstenberg*[a], Martin Steinert[a]

[a]Norwegian University of Science and Technology, Department of Mechanical and Industrial Engineering, Richard Birkelands vei 2B, 7491 Trondheim, Norway

* Corresponding author. Achim Gerstenberg. *E-mail address:* achim.gerstenberg@ntnu.no

Abstract

Prototyping iterations have been linked to design performance in complex problems solving. Does prototyping also enhance design performance in engineering product development where the information needed to solve the problem is already available? In this paper we experimentally compare if early testing of prototypes leads to different development outcomes compared to an approach that entirely relies on planning in the earlier phases of the project. In the experiment we ask participants to program a mobile robot to autonomously perform a task with a quantifiable performance outcome. The task is open-ended and allows for several solutions to the problem. To enhance the statistical validity of the results we develop and use a digital simulation. Both in the manually recorded and in the simulated results we observe a tendency that the participants that are allowed and encouraged to test their prototypes frequently perform better than participants that are not allowed to test prototypes. The performance of the initially non-testing participants equalizes after they had time to test and improve their designs and may even become better than the performance of the testing participants. However, both these performance differences are not statistically significant and a study with a larger number of participants is required. From qualitative observations, we conclude that the testing behavior of the individual developer may have a stronger influence on the outcome than if or how often a developers test their designs and further studies are needed.

Keywords: prototyping iteration; mobile robot; quantitative experiment;

1. Introduction

Iterations seem to be an inevitable and potentially desired part of product development [1]. Some literature describes iterations as opportunities for front-loading problem solving [2], speeding up the development process [3,4] and discovering unknown unknowns [5,6] while others see iterations as potentially wasteful repetitions that shall be minimized [7]. In this paper we experimentally compare if early testing of prototypes leads to different development outcomes compared to a development approach that entirely relies on planning in the earlier phases of the project. To investigate the influence of early iterations on the development performance of an autonomous mobile robot, we compare two experimental conditions where one is allowed and encouraged to test their designs frequently while the other is initially not allowed to test. Later we allow both conditions to test in order to see the influence of delayed testing on the performance.

The task used in the experiment is a complicated open-ended task according to the definition of Boone and Snowden [8] and has a quantifiable performance outcome that we use as a dependent variable. Open-ended means that multiple solutions exist, it is not easy to predict which solution works best. The information provided to the participants is sufficient to solve the task without acquiring additional information.

2. Experimental concept and conditions

In this experiment we want to quantitatively investigate if early testing of prototypes help with developing better robotic solutions. The study participants are assigned to two wither the non-testing or testing experimental condition. The non-testing

condition can compile their codes to discover syntax mistakes but they cannot load their codes onto the robot and thus cannot execute them during the first 80 minutes of the programming phase. In the testing condition the participants are allowed and encouraged to load and execute their codes frequently.
Participants in the testing condition are reminded that they can test if they have not tested during the last 5 minutes. However, the participants are free to decide if and when they want to test. 80 minutes after the programming phase started, both groups are requested to run their solutions in order to measure if the non-testing to testing stimulus in the first 80 minutes leads to a difference of performance. After this first evaluation after 80 minutes, four more evaluations follow in 10 minutes intervals to see if the performances progresses differently over time depending on the testing behavior during the initial 80 minutes. The initially non-testing group is also allowed to test after the first evaluation, i.e. from minutes 80 to 120 both experiment conditions are equal.

The participants of the experiment are recruited from 3rd year cybernetics students at the Norwegian University of Science and Technology. The participants volunteer and are not paid individually but receive a collective financial contribution for a class trip. 31 participants took part in the study. Three participants are excluded because of a malfunction of the experimental setup. One participant quit the experiment due to insufficient programming knowledge. Information about the participants included in this study are shown in table 1.

Table 1. Random assignment of the participants to the conditions

	male	female	Sum
Non-testing condition	11	2	13
Testing condition	11	3	14
Sum	22	5	27

The participants are assigned randomly to the two conditions and the time slots are equally distributed between the two conditions to avoid biases from the time of the day or the weekday. The participants are invited by a standardized email which includes information about the experiment content, the time and place with a way description to the experiment and it informs that the experiment is conducted without direct personal interaction with the experimenter.

2.1. The task

For studying the influence of testing prototypes on robot development performance, we use a task that offers a quantitative performance metric. Additionally, we want this task to be open-ended, meaning that there are several reasonable paths to solving the task and it is not obvious which one to prefer in order to study creative problem solving performance. The task also needs to be solvable within the given programming time of 120 minutes and repeatable under similar condition for every participant.

The task is to program the robot such that it can autonomously remove three cube objects out of an area with white surface in the shortest time possible. The robot shall be capable of solving the task from every starting position inside the white area while the starting positions of the cube objects remains the same. This means that the participants cannot find a solution that is specialized to one position but need to find a solution that adapts to the different conditions.

The participant can choose if and where to place up to three blinking lights. These lights can be detected by the robot and they fit into cut outs in the cube objects such that the light can shine out radially. Figure 1 shows the white area with the robot and the cube objects that need to be pushed outside of the white area.

A more detailed description of the experimental setup can be found in [9].

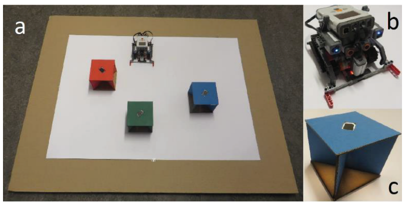


Fig. 1. (a) the physical setup with the playing field, the cube objects and the robot; (b) detailed view of the robot; (c) a cube object with a blinking light

2.2. Testable hypothesis

The research question is if early testing of prototypes influences the performance when developing robotic solutions. We therefore generate two falsifiable hypothesis, which reflect this research question.

Hypothesis 1:
After 80 minutes of programming, the solutions of participants in the testing condition in average remove more cubes and solve the task faster than the solutions from the non-testing condition.

Hypothesis 2:
After 120 minutes of programming, the solutions of participants in the testing condition in average remove more cubes and solve the task faster than the solutions from the non-testing condition.

2.3. Detailed timeline and interaction with the participant

Generally throughout the experiment, the interaction between the experimenter and the participant is avoided to not introduce biases caused by this interaction. The participants are informed that the experiment is conducted without direct personal interaction already in the invitational email.

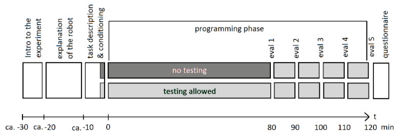


Fig. 2. Timeline of the experiment.

An overview of the experiment sequence is shown in Fig. 2.

After the welcoming by the computer generated voice the participant is asked to sign a consent form that informs about the purpose of the experiment, that the data is captured anonymously and the possibility to abort the experiment at any time. In a questionnaire the participants are asked to self-report their coding experience in C, Python, Arduino and using Lego Mindstorms. C is chosen because it is close to the language for programming the robot used in the experiment (NXC) and Python because it is the most taught language in the study program the participants were recruited from. Experiences with Lego Mindstorms or Arduino are relevant because they are related to the task given during the experiment. The participants are then introduced to NXC by a short code example. It also serves as a vague test of programming knowledge.

Next the participants get access to the robot and are given an introduction to the robot by a standardized and automated presentation on a screen. The participants receive a datasheet about the accuracy of the sensors and the driving capabilities as well as description of the library developed for the experiment. The library includes various functions for easy access to the sensor measurements and simplified commands for moving the robot. The participants are given 15 minutes to study this information and can keep the information sheets until the end of the experiment.

This is followed by in introduction into the development environment for programming the robot (BricX) that includes how to write, load and execute a code on the robot. After the participants have successfully loaded an example code onto the robot they receive the task description. The robot description and the task description are given by an automated presentation with images and computer generated voice. In addition the participants get the task description on paper and keep it until the end.

Before starting the programming phase the participants are informed about the experimental condition. For the non-testing conditions the key needed for loading code onto the robot is disabled during the first 80 minutes of the programming phase.

After the 80 minutes of building time with or without testing, all participants are asked to execute their best code at defined points in time. These evaluations occur at 80, 90, 100, 110 and 120 minutes of the building phase in order to gather data that is comparable between participants. The participants execute the code on the robot from given starting positions. The starting positions are different for each evaluation to avoid a single solution optimized for one position. To keep the results and the experience comparable every participant is given the same starting position and orientation of the robot.

The experimenter can observe and document the number of removed cubes and the time the robot needs to complete the task from a video stream. This stream and the executed codes are documented.

3. Manually recorded results

These result are obtained by observing how the robot fulfills the task during the five predefined evaluations. The positioning of the cubes is the same in all evaluations but each evaluation has a different starting position and orientation of the robot. Each participant experiences the same starting configuration. This means that we get one result for the number of removed cubes and a task completion time if the task is completed for each participant and evaluation. The dataset about the number of removed cubes therefore consists of 65 (13 participants times 5 evaluations) data points for the non-testing condition and 70 (14 participants times 5 evaluations) data points for the testing condition. The amount of data points for task completion time varies, depending on the evaluation, between 7 and 24 out of 27. This means that a meaningful comparison of task completion times between evaluations is not possible. In the earlier evaluations with less task completion, the comparison between the experimental conditions is also pointless. Although the intended performance metric was the task completion time we therefore concentrate on how many cubes the solutions removed, i.e. how close it came to completing the task.

Fig. 3 and 4 show the results for the non-testing and testing condition. They show histograms of how often a certain number of cubes are removed for each evaluation. Each evaluation consists of 27 data points and the height of the bars indicates how often no, one, two or three cubes are pushed off the white area. The evaluation is over when either the robot completes the task, the robot falls off the cardboard area, the participant aborts the execution of the program or none of the afore mentioned happens within 400 seconds after starting the robot.

We observe a bimodal distribution, meaning that the robot tends to either remove very few, often no, cubes or completes the task. Instances where the robot removes one or two cubes are rare. This phenomenon seems more pronounced for the non-testing condition and in the first evaluation of the testing group.

In the depth direction, we can compare the trend over time from evaluation 1 to evaluation 5. The participants keep developing their solutions between evaluations and we can observe for both conditions that over time an increasing number of solutions complete the task successfully and the amount of solutions that remove no cube decreases.

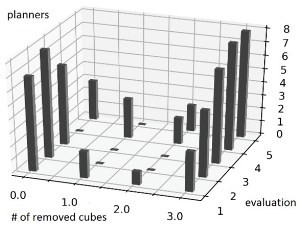


Fig. 3. Results of the non-testing participants. The histograms show how often a certain number of cubes are removed and in which evaluation.

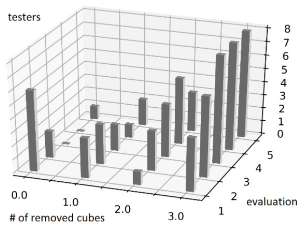


Fig. 4. Results of the testing participants.

The improvement between the first evaluation and the fifth evaluation is in both conditions statistically significant.

However, we observe no statistically significant difference between the non-testing and the testing conditions when comparing the two groups within one evaluation. From these results we cannot confirm any of the proposed hypothesis.

3.1. Limitations of the data and how to leverage it

The participants execute their solutions once at every evaluation. This means that we gather one data point per solution. The task demands that the solution works from any starting position of the robot. Since we only gather data from one position we do not know how this position influences the result and a single result may therefore not be a representative evaluation of the performance of the solution. To solve the issue the robot can be rerun after the experiment has ended from many additional starting positions to determine the average performance.

Furthermore, we observed that the result is very sensitive to the exact initial placement of the cubes and the robot and reproducing sufficiently similar conditions for every participant is not possible.

Both problems can be solved using a deterministic simulation. It can rerun the solutions from many starting positions and under perfectly repeatable conditions and automatically document the results.

The simulation we use in this publication is verified by qualitative and quantitative comparison with measurements of the robot completing the task in the real world. A precise description of the simulation and how it is verified can be found in [10].

4. Simulated results

The simulation reruns each solution from 99 randomly generated starting positions of the robot. For evaluating every solution, we use the same 99 positions each.

In each of those 99 simulation runs between zero and three cubes will be removed. This allows us to calculate the mean number of removed cubes for each solution as a performance measure and the standard deviation to estimate how consistently the solution performs. Similarly to the manually recorded results section, we can count how many solutions have a mean value of removed cubes within a certain interval. This data is plotted as histograms in Fig. 5 and Fig. 6 with a bin size of 0.75.

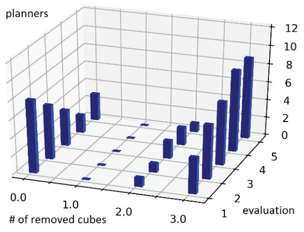


Fig. 5. Simulated results of the non-testing participants.

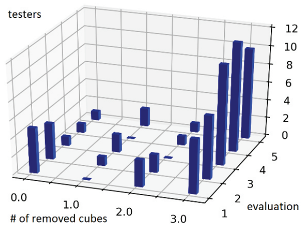


Fig. 6. Simulated results of the testing participants.

Similarly, to the manually recorded results, also the simulated results show a bimodal distribution especially for the non-testing group but also for the testing group and a performance improvement over time for both groups. The simulation results therefore confirm the manually recorded results. To compare the two experimental conditions we use the median and the mean of the standard deviations of the number of removed cubes calculated for each solution from the 99 simulation results. This is because the data for the two conditions are bimodal and the mean would not give a representative result for the number of removed cubes. The standard deviation of the number of removed cubes is not bimodal and we can therefore use the mean.

Table 2. Descriptive statistics (N – sample size, Mdn – median, MSD – mean of standard deviations) of the number of removed.

	Non-testing			Testing		
Evaluation	N	Mdn	MSD	N	Mdn	MSD
1	13	0.21	0.37	14	1.57	0.39
2	13	1.59	0.24	14	1.69	0.45
3	13	2.78	0.38	14	2.33	0.44
4	13	2.64	0.44	14	2.43	0.52
5	13	2.75	0.38	14	2.27	0.58

From the medians, we can see that the number of removed cubes improves during the first three evaluations for both groups and then remains constant with a higher number for the non-testing group. The medians in the first evaluation seem to differ largely. However, this is caused by the bimodal nature of the results. The tendency would have been reversed if one more solution for the non-testing group had performed successfully. Although the median of removed cubes is lower for the non-testing group during the early evaluations it is higher in the last three evaluations. The mean of the standard deviations of the solutions coded by the non-testing participants is lower throughout all evaluations and indicates that testing may lead to more consistently performing solutions.

Table 3. Comparative statistics comparing the non-testing and testing condition with a Mann-Whitney U test for the number of removed cubes and an independent sample t-test for comparing the standard deviations of the number of removed cubes.

	Mann-Whitney U test		Independent sample t-test	
Evaluation	U statistic	p-value	t value	p - value
1	87	0.432	0.17	0.865
2	84	0.376	1.77	0.090
3	87	0.433	0.51	0.612
4	84	0.376	0.67	0.510
5	70	0.166	1.75	0.093

To determine if the differences between the two experimental conditions are statistically significant we perform a Mann-Whitney U test on the number of removed cubes and a two-tailed independent sample t-test on the standard deviations. None of the differences in the number of removed cubes is statistically significant and there is a tendency that the solutions by the non-testing participants perform more consistently in evaluations 2 and 5. However, there is no experimental argument why this should be the case in these two evaluations and the significances for the other evaluations do not show any indication that there is such an effect.

In cases where the robot removed all three cubes and hence completed the task we can use the task completion time as a performance measure. The simulation is aborted if the task is not completed after 400 seconds and then the task is considered incomplete. We can count how often a solution completed the task out of the 99 simulation runs and if it completed the task at least once we calculate a mean task completion time and a standard deviation. To compare the two groups we use the mean of the mean task completion times and the mean of the standard deviations. The results are shown in table 4.

Table 4. Descriptive statistics (N – number of solutions with task completion times, MM – mean of the mean task completion times in seconds, MSD – mean of the standard deviations of the task completion times in seconds)

	Non-testing			Testing		
Evaluation	N	MM	MSD	N	MM	MSD
1	6	111.0	64.6	9	109.2	61.7
2	7	86.2	47.4	9	117.7	56.6
3	10	97.6	45.8	13	96.0	62.6
4	12	118.3	52.0	13	111.8	67.1
5	11	78.3	42.1	13	100.6	59.5

We do not see that the solutions of one group are consistently faster. However, there is a tendency that the testing group produces more solutions that complete the task. This difference is largest in the first evaluation (6 out of 13 (46%) of non-testing and 9 out of 14 (64%) of testing participants have a solution that completes the task) and decreases in later evaluations.

Generally, we expect that the task completion times improve over time as the already working solutions are further improved. In some cases the performance decrease in later evaluations can be explained by that more solutions complete the task and newly contribute to the statistic.

Except for the first evaluation, the mean of the standard deviations of the task completion times is lower for solutions of the non-testing group. Surprisingly, this standard deviation does not decrease over time for the testing group.

We assign the task completion times into 5-second intervals and count how often out of the 99 simulation runs a solution completed the task in each time interval. The results can be seen in the histogram plots in Fig. 7 for the non-testing and Fig. 8 for the testing participants. In the unlabeled axis, every line corresponds to one solution by one participant. The lines are ordered by how many entries the histogram has, that is how often the solution completed the task in this evaluation. The histograms of the solutions that solved the task the least often are located towards the left while the histograms of the more successful solutions are on the right side of the plot.

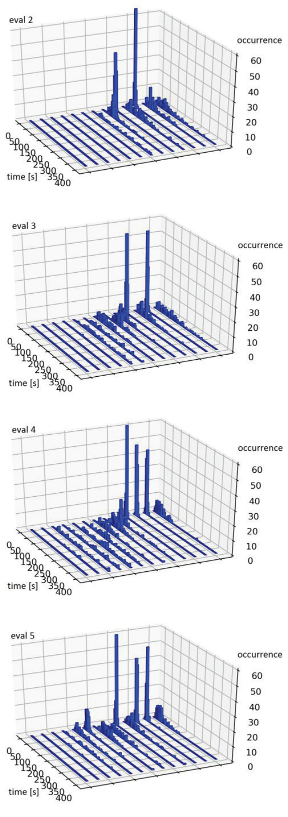


Fig. 7. Histogram of task completion times over 99 simulation runs for the non-testing condition

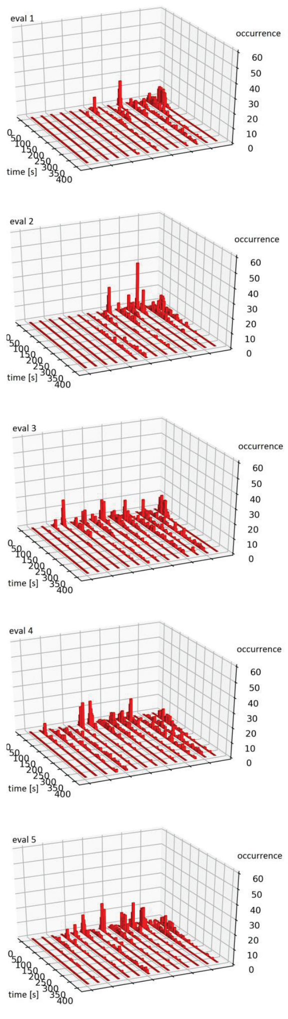


Fig. 8. Histogram of task completion times over 99 simulation runs for the testing condition.

In the histogram, we cannot only observe how many solutions complete the task at least once but also the time distribution for every solution. We see that some solutions, although they complete the task more often, do not necessarily have a more consistent task completion time. The three most consistently performing solutions are developed by the non-testing group.

5. Conclusion

In this experiment, we deal with the question if the solutions developed by participants who tested their designs while developing robot control software perform differently from the solutions of participants that could not test their designs within the first 80 minutes of the development time. We measure the performance of the solutions during the experiment when the participant executes the robot during pre-defined evaluations. The amount of this manually collected data is limited and difficult to obtain under precisely similar conditions. Therefore, we develop a digital simulation that can leverage this data by post-experimentally executing the codes that the participants produced during the experiment. Although not the primary aim of the experiment, this method proved very helpful to obtain more accurate results.

Hypothesis 1 was if testing leads to more removed cubes and faster task completion times in average after 80 minutes.

We observed a tendency that solutions developed by the testing participants in average removed more cubes and more frequently completed the task and thus perform better. However, this difference could not be confirmed to be statistically significant. This is mostly because the differences between participants within each group were very large. The development skill of the individual participant was more influential than, if the participants were able to test their designs. We did not observe a difference in task completion time.

The second hypothesis was that the group that could test during the initial 80 minutes of development still perform better after further 40 minutes of development time where both groups could test.

Again, we see no statistically significant difference. If at all, we see a tendency that the initially non-testing participants performed better by a small margin. They removed in average more cubes and did so, although only equally fast, with a more consistent task completion time.

A qualitative observation we made is that the non-testing participants planned the structure of their codes more carefully while the testing participants tended to discover flaws in their designs during testing and then patched those in a hustled way resulting in confusing code architectures that made improving the codes unnecessarily complicated.

As we saw from the results in evaluation 1, a sizable amount of participants who could not test their designs developed solutions that completed the task. Especially one participant proved that the provided information is sufficient to solve the task reliably in over 95% of the simulation runs. For this participant the task was simple enough to flawlessly construct and execute a mental model of the solution. We are aware that many product development problems are more complex and cannot be solved within two hours and, more importantly, are extremely difficult to solve by constructing and executing flawless mental models from already available information. This is because they either the necessary information is not available or the problem is too complex to be flawlessly developed without testing. This experiment is limited in this sense as finding a task is a compromise between making it as complex and representable for product development as possible while still make it possible to solve the task within the given time, have a quantifiable performance metric and ensuring identical conditions for all participants.

To make conclusive statements if there is an effect of testing prototypes early during the development process we need more participants to obtain statistically significant results. From this study we can say that the individual difference between developers seem to be larger than the effect of testing despite recruiting participants from the same study program and the same year.

Although we cannot find a quantitatively significant difference when introducing the testing versus non-testing stimulus we observed qualitatively differences. The way how participants plan their tests and how they react to test results has a major influence on their individual result. To better understand this we need to combine quantitative research with qualitative research to uncover connections and develop new hypothesis for further quantitative research.

Acknowledgements

This research is supported by the Research Council of Norway through its user-driven research (BIA) funding scheme, project number 236739/O30S.

References

[1] Smith RP, Tjandra P.Experimental observation of iteration in engineering design. Research in Engineering Design 1998;10:107-117.
[2] Thomke S, Fujimoto T. The Effect of "Front-Loading" Problem-Solving on Product Development Performance. Journal of Product Innovation Management 2003;17: 128-142.
[3] Schrage M. The culture(s) of prototyping. Design Management Journal 1993;4:55-65.
[4] Leon HCM, Farris JA, Letens G,Hernandez A. An analytical management framework for new product development processes featuring uncertain iterations. Journal of Engineering and Technology Management;30:45-71.
[5] Steinert RM, Leiffer LJ. Finding one's way: Rediscovering a hunter-gatherer model based on wayfaring. International journal of engineering education 2012;28:251-252.
[6] Jensen MB, Elverum C, Steinert M. Eliciting unknown unknowns with prototypes: Introducing prototrials and prototrial-driven cultures. Design Studies 2017;49:1-31.
[8] Snowden ME, Boone DJ. A Leader's Framework for Decision Making. Harvard Business Review 2007;85:1-9.
[9] Gerstenberg A, Steinert M. Open ended problems – a robot programming experiment design to compare and test different development and design approaches. Proceedings of NordDesign 2018
[10] Gerstenberg A, Steinert M. Development and verification of a simulation for leveraging results of a human subjects programming experiment. arXiv eprint: 1903.10420. March 2019

C.10 Contribution 10: Fixation on Premature Concept Choices - a Pitfall of Early Prototyping?

Manuscript in PDF

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2019) 000–000

www.elsevier.com/locate/procedia

29th CIRP Design 2019 (CIRP Design 2019)

Fixation on premature concept choices - a pitfall of early prototyping?

Achim Gerstenberg*[a], Heikki Sjöman[a], Martin Steinert[a]

[a]*Norwegian University of Science and Technology, Department of Mechanical and Industrial Engineering, Richard Birkelands vei 2B, 7491 Trondheim, Norway*

* Corresponding author. Achim Gerstenberg. *E-mail address:* achim.gerstenberg@ntnu.no

Abstract

Building prototypes is an essential element in conceptual design. We argue that using resources to build prototypes may induce adherence to the chosen concept and prevents further exploration into other concepts. This phenomenon has previously been attributed to sunk cost caused by building prototypes. In a controlled human subject experimental study in a robot development context we investigate the influence of building and testing prototypes by allowing one group of participants to test their prototypes frequently while the other group is not allowed to test and can only rely on the provided information about the hardware. We report about participants prematurely committing to concept choices and adhering to those after building and testing prototypes while non-testing participants make superior concept choices based on the provided information. While planning may be feasible in some projects with low uncertainty, problems that are more complex require prototyping for knowledge acquisition. We give suggestions on how to reduce the costs of prototyping and the associated effect it has on design fixation. These suggestions are very similar to the test driven development approach known from software development. They include the definition of critical functions and the respective tests before building the prototypes. When designing the prototypes the focus lies on making a conscious choice of how to prototype with the lowest fidelity necessary to comply with the previously defined test and attempting risky development stages early with the intention of maximizing the work not done.

Keywords: design fixation; sunk cost; conceptual design

1. Introduction

While several studies show the benefits of building and testing prototypes during the conceptual design phase [1-6] other studies argue that prototypes can be a waste of resources [7] and potentially lead to design fixation [8,9]. In this publication we use Jansson's and Smith's definition of design fixation [10].

Viswanathan and Linsey did not find evidence that building prototypes in itself is a cause for design fixation [2] but that the sunk cost associated with building prototypes can lead to increased design fixation [11] following the theory of the sunk cost effect [12,13].

According to this literature we expect that spending resources on prototyping the initial concept can lead to a design fixation on this concept due to the sunk cost effect. This adherence to the initial selection then prohibits the development and comparison of alternative concepts leading to premature concept choices. Developers may be particularly susceptible to this effect if the prototypes show somewhat promising results even when a better concept choice exists.

We investigate design fixation in an experimental study with a robotic task. The experiment aims to answer the question if testing prototypes allows for comparison and therefore better selection of concepts or if it leads to fixation onto the prototype that is initially built.

All participants are given sufficient information to make intelligible concept choices without the need of testing prototypes and are primed by a functioning but inferior solution just before developing the solutions to induce a bias towards the inferior concept. The participants who can test and evaluate their prototypes are compared to participants who solve the task after planning their solutions using mental models without testing and evaluating those physically.

We first describe the experimental setup and explain and compare the different concepts used by the participants. We proceed with the statistics about the concept choices of the participants from both experimental conditions, a discussion about the implications on early stage product development, how we think prototypes shall be used and what future work can be done to further understand and quantify the effect.

2. Experimental design

We perform a controlled human subjects study in a laboratory setting to compare if testing of prototypes leads to different concept selection than deciding and planning based on mental models of solutions. The participants of the study are asked to individually program a provided physical mobile robot in a way that it autonomously finds and moves objects. In this context testing a prototype means loading the developed code onto the robot and executing it to observe the robot behavior. The first group can build, test and compare prototypes of different concepts and can base their final concept decision on the discoveries from testing while the other group needs to compare their mental models of different concepts and then execute the concept that appears most favorable based on information provided prior to the design phase of the experiment. The same information is also provided to the first group and in principle those participants can also decide purely based on this information but participants of this group are encouraged to test prototypes at regular time intervals and are reminded after every 5 minutes without testing that they can test. The participants that are allowed to test are reminded but can still decide themselves if and when they want to test. The design phase lasts 80 minutes for both groups.

As the last instruction before the design phase both groups are primed with an inferior solution by mentioning a hint towards this inferior solution. Thereby the need of evaluating the concepts, either by testing or by carefully studying the information, is amplified.

2.1. Participants of the study

The participants are voluntarily recruited from a third year cybernetics class at a technical university (more precise information disclosed for blind review). The participants do not get paid for participation individually but are indirectly monetarily rewarded towards a class trip. A total of 27 participants (22 male, 5 female) take part in the study and are randomly assigned to either condition. The non-testing group includes 13 participants and the testing group includes 14 participants.

2.2. The task

Participants are given a Lego Mindstorms robot that they program so that it can autonomously solve an open-ended task. The task is to remove three cube objects from a white area of 1.8 m^2 in the shortest time possible after starting the robot. The starting position of the robot is unknown to the participant and the solution shall work from any possible starting position of the robot. The robot has two motors to drive forwards, backwards and turn and sensors to detect the reflectivity of the surface underneath the robot, an ultrasonic distance sensors and light sensors that can detect the color of an object and recognizes a blinking light. Three blinking lights are provided and they can be placed anywhere including inside the cube objects.

The possibility of fitting the lights into the cube objects in a way the robot can detect them from any direction relative to the cube object is given as the last part of the task description. This serves as priming all participants initially towards using the blinking lights for finding the cube objects.

All participants see a programming environment with several similar simultaneously opened coding windows. This makes it possible to develop several codes in parallel.

More detailed information about the physical setup, a more precise description of the robot and the programming library, the standardized interaction with the participants and other experiment examples this setup can be used for are described in [14]. Figure 1 shows the playing field with the robot and the cube objects that the robot needs to move outside of the white area while not falling off the cardboard area itself.

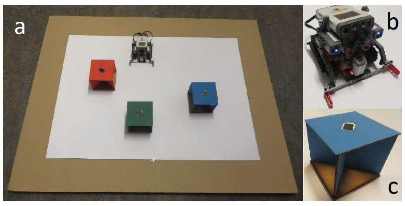


Fig. 1. (a) the physical setup with the playing field, the cube objects and the robot; (b) detailed view of the robot; (c) a cube object with a blinking light

2.3. Possible concepts

We retrospectively identified four concepts that were used by the participants to detect and remove the cube objects. In the following we will explain and compare those four concepts.

1. Object detection based on the ultrasonic distance sensor
2. Object detection based on detecting the blinking lights inside the cube objects with the light sensors
3. Combination of the ultrasonic and the light sensor
4. No sensor used – random trajectory to remove cube objects by chance

Concept 1: The ultrasonic distance sensor sends out an ultrasound pulse and waits until a reflection of this pulse returns to the sensor and calculates a distance reading from the time of flight of the pulse. Receiving a reading from this sensor can be regarded as almost instant and this sensor can be used for very frequent and accurate measurements while the robot is moving. The range of the sensor is larger than the operating area of the task. This means that a cube object at any position can be detected with the ultrasonic sensor from any position of the robot. A disadvantage of the ultrasonic distance sensor is its wide acceptance angle. This means that the sensor detects an object when the sensor and thus the robot is not necessarily directly pointing at the object.

A solution using this sensor to detect cube objects has the largest potential to solve the task quickly and with the shortest code. The best ultrasound based solution written by a participant takes 19 seconds to solve the task averaged from 99 random starting positions of the robot. The shortest functioning ultrasound based code takes four lines of code. Being the fastest and simplest to realize concept makes the ultrasound based solution superior over all other concepts.

Concept 2: The robot has forward pointing light sensors on either side that can detect the blinking light sources provided to the participants. If these blinking lights are placed in the cube objects, they can be used as beacons for detecting the cube objects. The intensity of the blinking light correlates with the distance between the sensor and the light. The lights blink with a frequency of 0.93 Hz and the algorithm needs a full blink cycle to differentiate the blinking light source from a non-blinking light source. This means that the minimum time for measuring the blinking light intensity accurately is one blink cycle. The detection range is less than the operating area of the robot and this means that a robot that is turning to search for an object may be out of range for detecting one and needs to move closer first.

This means that even if all cube objects are within the detection range the measurement delay makes the detection slow because the robot needs to wait for an accurate measurement and in some cases the robot needs to change location after an unsuccessful search to get into detection range. This requires extensive code and the search speed is limited by the measurement frequency. The fastest blinking light based solution averaged over 99 starting positions is 52 seconds and therefore significantly slower than the ultrasound based solution. The code needed for a reliably working solutions is longer and more difficult to implement that an ultrasound based solution.

Concept 3: Some participants combined both search methods. This redundancy makes this solution potentially more resilient although the ultrasound methods already works very reliably. The non-testing group could not find this out without testing. The solution is slowed down by the measurement time needed for accurately determining the intensity of the blink lights and is therefore usually slower and more complex to code than the purely ultrasound based solutions.

Concept 4: The last concept employed by one participant is to use no search method at all. The robot drives straight until it detects the edge of the operating area and then turns until it is facing back towards the center and drives straight again. The benefit of this solution is that the robot can always drive at full speed because it does not need to take accurate measurements. The disadvantage is that it does not recognize any cube objects and only removes those by chance. The first cube is often removed very quickly but finding the last remaining cube by chance is less likely. When the robot is started from different starting positions the time for completing the task varies significantly between repetitions.

All the mentioned information about the sensors are available to the participants in a printed datasheet.

This means that ultrasound based solutions are clearly the most favorable and easiest to implement solutions. From the provided information it is possible to conclude that the blinking light based solution is slow because of the measurement delay and its range can be insufficient in some cases. The solution that did not use any search method is difficult to develop without testing because it is nearly impossible to mentally compute the trajectory even if the provided information is accurate.

3. Experimental results

We examined the final codes that the participants developed during the programming time of 80 minutes. We identify the search method used for detecting and approaching the cube objects and categorize them into the above mentioned concepts.

We observe that most participants use the ultrasound based concept: 8 out of 13 (62%) in the non-testing group and 8 out 13 (62%) in the testing group. 5 out of 14 (35%) of the testing participants chose the blinking light concept as their final concept while none of the non-testers chose it as their final concept. The solutions that combines both concepts were used 4 times by the non-testing and 2 times by the testing participants. The one participant who used no search method was in the non-testing condition.

Table 1. Frequency of concept choices for the non-testing and testing experimental condition. The non-testing condition includes 13 participants and the testing condition includes 14 participants.

Concept	Non-testing	testing
Concept 1, ultrasound based	8	7
Concept 2, blink light based	0	5
Concept 3, combination of concept 1 and 2	4	2
Concept 4, no object detection used	1	0

4. Interpretation of the data

The majority of the participants from both groups use the superior ultrasound solution. This means that it is reasonable to come to this conclusion both with and without testing of prototypes. However, we observe that exclusively the non-testing group chooses the inferior blinking light based concept. Since the only experimental difference between the groups is that one of them can test their prototypes while the other cannot we conclude that this difference leads to a poorer concept choice.

The non-testing group is reliant only on the provided information. This forces them to make and compare mental models to decide which concept is most favorable. In contrast, the testing participants are encouraged to test their solution early. This means that they may decide more prematurely and without carefully studying the information. From the provided information it is possible to deduct, but not immediately obvious, that the ultrasound based solution will outperform the blinking light based one. We therefore assume that some of the testing participants were not aware of this and initially chose the inferior concept. The testing participants could have built and tested several different concepts and then decide which one works better. None of the testing participants tried out a different concept but instead began and continued with the initially chosen concept. We assume that this is due to avoiding the sunk cost of giving up development work already done or using additional time for developing a concept that then later proves inferior. Indeed spending time on developing something new in case the superior concept is already started is ineffective but it can help discover inferior premature concept choices.

The described effect may have been enhanced by the partially positive feedback gained from testing the inferior blink light solution. Although this solution is inferior, it is still a viable option and a promising test result may impede the participants from questioning their concept choice.

Although we biased participants towards making an inferior initial concept choice and incentivized them to test their designs early, which may have increased the chance for premature decisions, we belief that too early testing and focusing on an initial idea is generally a reasonable scenario during conceptual design. This is also demonstrated by Purcell & Gero [15].

5. Implications on product development

In this experiment the information provided was sufficient to plan a successful solution. However, many product development projects are more complex and require prototyping to gain insights necessary to solve the problem and prototyping becomes an integral part of the development process. In those projects without sufficient information early concept choices often cannot be rationally made and are likely premature. If the sunk cost of building prototypes leads to an adherence to this prematurely chosen and possibly inferior concept then it is important to minimize the sunk cost while still gaining the required insights from the prototypes and comparing several concept design choices. To minimize sunk cost we suggest to limit the prototyping to critical functions and reduce the fidelity of the prototype to the minimum necessary to gain the desired insights. This means that before starting to build a prototype one needs to define the most critical, absolutely needed, requirements of the entire project (e.g. a plane needs to fly) and how to test the prototypes according to those critical functions as well as the criteria for a passed test.

Secondly, the developers need to choose a concept where they find that the critical functions can be prototyped easily. The focus shall be on what is the minimum building effort needed to gain sufficient insight when testing the prototypes for later comparison of concepts. This will lead to the prototypes with the lowest necessary resolution and the focus lies on maximizing the work not done. Here we aim to firstly develop the solutions to the critical function that is most likely to fail. If the solutions fail and we find out that we cannot solve the critical function the concept has failed early, can be given up sooner and the sunk cost is minimized. The intention of testing is not primarily to find out how concepts different compare in absolute numbers as these are influenced by how well the prototypes are built and we cannot precisely compare prototypes that are built to different standards.

The intention is to find unsuspected flaws in the designers mental model, get an impression of the order of magnitudes of performances and performance potentials of the solution, a feeling for how difficult a further development of the technology will become and where the benefits and disadvantages lie.

When minimizing sunk cost and knowing about the danger of design fixation it is possible to use the remaining available resources and testing of other concepts and allow for an informed comparison when deciding which concepts to proceed with.

The general concept resembles many ideas from test-driven development (TDD) [16] of software. In TDD the developers first define the desired behavior and write code that tests if the software solution fulfills this behavior. These tests are called unit tests and are written before any code that solves the tests is written. They focus on one specific behavior that shall be added to the solution and concisely define criteria for when the test is considered as passed.

In our proposed method for physical projects those unit tests correspond to the definition of critical functions and the associated test criteria that must be fulfilled to pass the test.

Then the software developers program the minimal code necessary to pass the unit test while avoiding development of unnecessary features that are not fundamental for passing the test. Since the unit tests as well as the solutions to these unit tests focus on the essential requirements, the development and testing of unneeded features is avoided and potentially sunk costs are minimized. In physical projects, this means that the lowest prototype fidelity that is necessary to pass the test criteria and provide learning insights from testing is sufficient. The resources saved can then be used for building, testing and comparing other concepts and design fixation can be decreased. The final step in TDD is refactoring where code duplications are removed and performance gets optimized. In the physical world this is equivalent to optimizing the chosen concept with the insights gained earlier by prototyping using conventional product development models.

In software projects, the unit tests can easily be repeated to check if future software versions still fulfill the unit tests. In the case of hardware development testing a prototype is usually not as automated as rerunning a code and is therefore more time consuming. Furthermore, the prototype may be destroyed during the test and using it again for another test is not possible.

We now illustrate this approach with an example from the experiment used in this study. The most critical function when developing the robot is to ensure that the robot does not fall off the cardboard as this makes completing the task impossible. A reasonable unit test therefore is that the robot needs to approach the edge of the cardboard at full speed from all possible angles and manage to stop before falling off the cardboard. The minimum solution that passes this test is to use the downwards reflection sensor and stop the motors when the reflectivity is below a threshold. Obviously, this does not solve the entire task. The complete solution requires the robot to turn around and then continue searching but this does not need to be developed before we know that we can detect the edge and prevent the robot from falling off.

6. Conclusion

Building and testing prototypes is a known method in product development. It allows to compare competing concepts and make design decisions based on test results. One would therefore assume that testing prototypes helps with selecting favorable concepts.

In contrast to this, some literature suggests that the sunk cost of creating and then having to discard prototypes leads to design fixation. After building an initial prototype designers then adhere to the chosen concept to avoid spending resources on building additional prototypes. This saves resources but if the initial concept choice is inferior to other concepts, the design fixation consolidates the poor concept choice.

In an experiment, based on developing a robotic solution, we nudge participants into prematurely making an inferior concept choice by presenting them with information about an inferior solution. We then observe if participants that have the chance to test prototypes are more or less likely to ignore the inferior concept and choose a superior concept than participants that make their concept choices only based on provided information and comparing mental models of their ideas. The results show that participants, if provided with sufficient information that rely on mental models and thus have less sunk cost are able to make superior concept choices than participants that in principle can test and compare different concepts but refuse to do so.

The results are indicative but not significant enough to conclude that the sunk cost of prototyping leads to design fixation. However, we see this possibility and suggest a process that is adapted from test-driven software development and aims to minimize sunk cost while still using comparison of prototypes to make concept choices.

Acknowledgements

This research is supported by the Research Council of Norway through its user-driven research (BIA) funding scheme, project number 236739/O30S.

References

[1] Schrage M. Cultures of prototyping. Design Management Journal 1993;4:55-65.
[2] Viswanathan VK, Linsey JS. Physical Models in Idea Generation: Hindrance or Help?. ASME proceedings, 22nd International Conference on Design Theory and Methodology 2010;5:329-339.
[3] Steinert RM, Leiffer LJ. Finding one's way: Rediscovering a hunter-gatherer model based on wayfaring. International journal of engineering education 2012;28:251-252.
[4] McKim RH. Experiences in Visual Thinking. Boston: PWS Publishing Company; 1972.
[5] Gerstenberg A, Sjöman H, Reime T, Abrahamsson P, Steinert M. A Simultaneous, Multidisciplinary Development and Design Journey – Reflections on Prototyping. Proceedings of Entertainment Computing – ICEC 2015:409-416.
[6] Dow S, Heddleston K, Klemmer SR.The efficacy of prototyping under time constraints. Proceedings of the Seventh ACM Conference on Creativity and Cognition 2009; 165-174.
[7] Baxter M. Product design: Practical methods for the systematic development of new products. London: Chapman & Hall; 1996
[8] Christensen B, Schunn C. The relationship of analogical distance to analogical function and pre-inventive structure: The case of engineering design. Creative cognition: Analogy and Incubation 2005;35:29-38.
[9] Kiriyama T, Yamamoto T. Strategic Knowledge Acquisition: A case study of learning through prototyping. Knowledge-based systems 1998;11:399-404
[10] Jansson DG, Smith SM. Design Fixation. Design Studies 1991;12:3-11
[11] Viswanathan VK, Linsey JS. Design Fixation in Physical Modeling: An Investigation on the Role of Sunk Cost. ASME proceedings at the International Design Engineering Technical Conferences & Computers and Information in Engineering 2011;119-130.
[12] Arkes H, Blumer C. The psychology of sunk cost. Organizational behavior and human decision process 1985;35:124-140
[13] Kahnemann D, Tversky A. Prospect theory: An analysis of decision under risk. Econometrica 1979;47:263-291
[14] Gerstenberg A, Steinert M. Open ended problems – a robot programming experiment design to compare and test different development and design approaches. Proceedings of NordDesign 2018
[15] Purcell AT, Gero JS. Design and other types of fixation. Design Studies 1996;17: 363-383
[16] Beck K. Test-driven development: by example. Addison-Wesley Professional; 2003

C.11 Contribution 11: A low-cost predictive display for teleoperation: investigating effects on human performance and workload

A low-cost predictive display for teleoperation: investigating effects on human performance and workload

Henrikke Dybvik[1], Martin Løland[1], Achim Gerstenberg[1], Kristoffer Bjørnerud Slåttsveen[1], Martin Steinert[1]

[1]Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology (NTNU), Richard Birkelands vei 2B, 7491 Trondheim, Norway
Henrikke.dybvik@ntnu.no, +47 95417245

Abstract

Teleoperation in an environment with latency is difficult and highly stressful for human operators, resulting in high cognitive workload and decreased human performance. This work investigates if a simple predictive display can increase performance and lower subjective workload for the human operator when teleoperating a remotely operated vehicle (ROV). A predictive display based on image transformation was developed by applying positional and scale transformations to the video feed and tested. An experiment was designed, consisting of a simple navigational task (peg-in-hole game) with a ground ROV, in three distinct conditions: C1. Latency, C2. Latency with predictive display (PD) and C3. Baseline (no added latency). Findings from N = 57 participants show a statistically significant increase of 20% in human performance with the aid of the predictive display. Although differences in subjective workload was not statistically significant, both subjective performance and actual game performance did increase significantly by using the predictive display. In fact, the latter almost doubled for participants defining themselves as regular gamers. Lastly, A principle component analysis (PCA) was conducted investigating confounding factors with confirmatory results.

Keywords: predictive display, human operator, performance, subjective workload

Acknowledgements

This research is supported by strategic funds from the department for Mechanical and Industrial Engineering (MTP) at NTNU.

1 Introduction – Predictive technology can combat the detrimental effects of latency in teleoperation

Teleoperation, also called remote operation, is electronic remote control of machines or vehicles and it includes applications of remotely operated vehicles (ROVs) on ground, under water, subsea, aerial and in space (Draper et al., 1998). Teleoperation is a subclass of telepresence; "the perception of presence within a physically remote or simulated site" (Draper et al., 1998). Telepresence is generally viewed as being beneficial to mission performance and is furthermore hypothesized to improve efficiency and/or reduce operator workload (Draper et al., 1998). There are multiple challenges related to teleoperation, one of which is latency. In this work, we are interested in latency, also called time delay, which refers to the delay between operator input action (steering commands) and visible output response of the video feed (Chen et al., 2007). Teleoperation in an environment with latency, especially basic driving, is difficult and highly stressful for the human operator, resulting in high cognitive workload (Matheson et al., 2013) and decreased human performance (Chen et al., 2007), e.g. observed as an increase in task completion time or reduced accuracy (Lane et al., 2002). Approaches to overcome the detrimental effects of latency in teleoperation include increasing the level of automation (which excludes the human from the loop), provide information to increase the situational awareness of the human operator and predictive technology. Predictive technology spans several approaches, either categorized as dynamic system models or free model approaches. Model free approaches include superimposed information models, 3D graphic models, and video manipulation. Superimposed information and 3D graphics models show promising results by greatly reducing task completion times, but require advanced algorithms, potentially expensive equipment and extensive information regarding the environment and the ROV. Video manipulation can increase performance of human operators' and it is simpler in comparison, as it alters the delayed video feed to mimic movements and environment in real time. Simple video manipulation can provide time efficient and inexpensive means to enhance performance of human operators' in settings where extensive information regarding the ROV and its environment is unavailable, or the opportunity to utilize expensive equipment or highly advanced algorithms is not a possibility.
With basis in existing video manipulation methods based on image transformation, we developed a simpler predictive display by applying image positional and scale transformations to the video feed. This predictive display requires a few lines of code and can be applied to several ROV configurations. In this work, we are interested in human operators' performance and their subjectively experienced workload while using predictive technology. The aim of this article is to investigate if a simple predictive display can increase performance and lower subjective workload of human operators' during teleoperation. To do so an experiment was set up to investigate changes in human operator performance and workload when operating an ROV under three distinct conditions, each condition with a distinct display and latency. The participants were presented with a single, simple navigational task, framed as a peg-in-hole game using a ground ROV with a first-person camera view. The conditions were C1. Latency, 2. Latency with Predictive Display (w/PD) and 3. Baseline. Data collected included objective performance (task score), and subjective workload (RTLX), demographics and other variables. N=57 participants were recruited and the hypotheses (task performance and subjective workload) tested using ANOVA. A post hoc Exploratory Data Analysis (EDA), specifically a principal component analysis (PCA) explore influencing factors.
Following the introduction, the article is structured as follows; the background second section covers challenges in teleoperation, focusing on latency and its detrimental effects on human performance and workload. Means to compensate for latency are discussed, emphasizing various predictive technologies. The third section describes development and implementation of a predictive display, and the experiment design, including stimuli, data collection, procedure, and data analysis. Section four presents the results of the statistical tests before providing the result from the EDA. A discussion of the presented results follows in section five, before the conclusion.

2 Background – Latency in teleoperation, human performance, and workload

This section describes challenges in telepresence, detailing latency and its detrimental effects, with a focus on human operator performance and subjectively experienced workload. Human operator performance decrease and workload increase as latency is introduced in teleoperation. Means to compensate for latency are discussed, predictive technologies in particular. Lastly, the section provides means to measure human operator performance objectively and workload subjectively.

2.1 Latency in teleoperation and its related challenges

2.1.1 Telepresence and related challenges

Draper et al. (1998) defines telepresence as "the perception of presence within a physically remote or simulated site". Teleoperation is one subclass of telepresence (Sheridan, 1995). Telepresence is beneficial to mission performance and is furthermore hypothesized to improve efficiency and/or reduce operator workload. Chen et al. (2007) reviewed 150 articles investigating factors in telepresence, and how they influence operator performance and challenges related. They found eight main factors; field-of-view (FOV), orientation, camera viewpoint, depth perception, video quality and frame rate, time delay (or latency) and motion.

2.1.2 Latency challenges in telepresence

In this work, we are interested in time delay, or latency, which will be used throughout this article, which refers to the delay between operator input action (steering commands) and visible output response of the video feed (Chen et al., 2007). Latency is usually a result of information having to be conveyed over a communication network (Chen et al., 2007). The total latency of the teleoperation system can further result from a combination of a number of reasons, such as software design, hardware design, physical limitations such as distance and obstacles, signal processing, etc. Thus, total latency can be both fixed and variable (Lane et al., 2002). There are important distinctions between the two, e.g. they influence performance differently (Davis et al., 2010; Neumeier et al., 2019; Oboe & Fiorini, 1998).
The causes of latency are not within the scope of this work, and we consider fixed delay only. We are interested in the total perceived latency; i.e. the time from when the human operator issues a command until they visually perceive a reaction in the vehicle in the video feedback.

2.1.3 Latency in teleoperation and its detrimental effects

Latency produces a mismatch between given input commands and visual feedback of vehicle reactions. This creates a conflict for human perception. To correct for this during operation the human operator must remember the inputs command given until they see the desired output produced by the vehicle in the video feed (Matheson et al., 2013). In addition, as new information is prompted on the video feedback this must be mentally connected with the commands issued previously (i.e. the vehicles previous state), and thereafter combined that with issuing new commands based on this conjunction of information (Ricks et al., 2004). Latencies as low as 10 - 20 ms can be detected by humans' visual perception (Chen et al., 2007). Taken together, this can degrade human performance (Chen et al., 2007) and can increase subjectively experienced workload (Ricks et al., 2004).

2.1.4 Latency in teleoperation degrades human performance

The detrimental effect of latency on human performance can be seen in Table 2.1.1, which includes relevant research investigating the effect of video feed latency on human performance in a given task. Human performance includes course completion time, task completion time, task score, accuracy, etc. This table describe the task and the related increase factor, where a 40% increase in task completion time corresponds to an increase factor of 1.40. For example a needle-driving task at 100 ms latency had an increase factor of 1.5 (Xu et al., 2014). The relationship between latency and task completion time is task dependent, notably it is similar for similar tasks. For example; a linear relationship between latency and task completion time was found in a mobile robot operating task (Ando et al., 1999) and a vehicle peg-in-hole task (Lane et al., 2002), whereas an exponential relationship was found in a telerobotic surgical technique task (Xu et al., 2014).

2.1.5 Latency in teleoperation increase workload

The notion of workload or cognitive load is argued to be predictive of both performance in human-machine interactions as well as the mental state of the operator. Workload is described as a relation between the mental resources a task demands and the resources available from the human operator (Parasuraman et al., 2008). It is a multidimensional construct emerging from the interaction between task, context, operator capabilities, behavior, perceptions and (mental and physical) state (Hart & Staveland, 1988a; Parasuraman et al., 2008). This mental load posed on a human operator by latency in teleoperation negatively affects their ability to control a vehicle in an efficient manner (Ricks et al., 2004). The human operator's *subjectively* experienced workload is important (Hart & Staveland, 1988a), since this might alter behavior. Should an operator experience a situation as high workload, the operator might adopt strategies to mitigate workload. In the specific case of teleoperation human operators tend to perform steering commands correcting for the mismatch in given input and visually perceived output, causing the vehicle to oscillate and limiting top speed (Appelqvist et al., 2007). Teleoperation in an environment with time delay, in particular basic driving, is difficult and highly stressful for the human operator, resulting in high cognitive workload (Matheson et al., 2013). Extended exposure to such an environment can create cognitive overload leading to mental fatigue (Lim et al., 2010; Matheson et al., 2013).

Table 2.1.1. Task completion time for a variety of tasks and latencies.

Author	Task	Participants	Latency [ms] and increase factor		
			100 – 300 ms	400 – 700 ms	800 – 1500 ms
(Fabrizio et al., 2000)	Pin transfer	N = 6	1.04 - 1.21*	1.17 - 1.41*	1.11 - 1.58*
(Xu et al., 2014)	Energy dissection	N = 16	1.4 - 1.8	2.7 - 4.3	
(Xu et al., 2014)	Needle-driving	N = 16	1.5 - 2.1	2.5 - 6.2	
(Perez et al., 2016)	Surgical simulator	N = 37	0.75	1.5	
(Lum et al., 2009)	Block transfer	N = 14	1.45	2.04	
(MacKenzie & Ware, 1993)	Target acquisition	N = 8	1.64		

* Estimated from graph

2.1.6 Latency compensation

There are multiple approaches to reduce the detrimental effects of latency. First, increasing the level of automation (LOA) reduces the operator workload and improve safety (Dorais et al., 1999; ENDSLEY, 1999; Goodrich et al., 2001; Luck et al., 2006; Schutte, 2017). A second option is providing the human operator with information and/or previously given input commands, increasing situational awareness and leading to higher performance and/or decreasing subjective workload (Chen et al., 2007; Miller & Machulis, 2005; Nielsen et al., 2007). A third option is predictive technology, which can be displays, control algorithms and graphical models attempting to predict the state of the ROV based on the vehicles current state and commands issued by the operator. Chen et al. (2007) conclude it is the most promising solution if eliminating latency from the system is impossible, and highlight that predictive displays has been shown to reduce task performance time by up to 150%.

2.1.7 Predictive technology

A range of experiments where predictive technology has been used are shown in Table 2.1.2, illustrating a wide variety of experimental tasks, robot configurations and predictive method. Exact robot configuration can be known, including examples such as robot-arm manipulators fixed to a user defined reference frame, or not known, such as vehicles subjective to external forces or floating freely. The unknown robot configuration challenges the predictive technology as it must account for unknown and changing external factors. Common for the experiments in Table 2.1.2 is that they involve lateral movement in an alignment or aiming task, which are particularly exposed to detrimental effects of latency in video feedback. Correctional behavior commonly occurs, causing operators to overshoot a target or employ a wait-and-move strategy. This behavioral strategy increases task completion time and occurs around one second latency (Lane et al., 2002).
In general, predictive technology calculates a future predicted state of the robot based on different variables and methods. Methods can rely on dynamic system equations, such as Zhang and Li. (2016) who used a spacecraft's state equations and its dynamic properties to calculate the predicted state. An image of the predicted state is provided to the operator which can issue commands accordingly. In contrast, a model free approach, which excludes dynamics, is often used in contexts where accurate modeling of external forces isn't a possibility, such as in space applications. Predictive technology within model free approaches includes superimposed predictive information, 3D graphic models and video manipulation.
The first category superimposes (or overlays) information on a delayed video feed, providing the operator with an estimate of the vehicles future state. Superimposed predictive information is often visualized as vector graphics where lines of dots follow a path. For example, Mathan et al. (1996) superimposed directional velocity information of a lunar rover on a video display. Further, airplane and helicopter displays have a *tunnel in the sky* showing where the aircraft should be going and a cross indicating the predicted trajectory (Grunwald et al., 1981). In cases with large amounts of lateral movement this approach might not be applicable as the predicted heading can come off screen.
3D graphics model (or virtual reality (VR) based predictive display) use sensor technology input such as Monocular Simultaneous Location and Mapping (SLAM), stereo imagery, vision-based structure from motion (SFM), light detection and ranging (LiDAR), or radio detection and ranging (radar), etc., to construct a three-dimensional world, wherein images from ordinary cameras are rendered on the surface of the virtual world.
Then, a virtual camera is placed inside the virtual world in the predicted position of the real camera and operators' are presented with the virtual video feed as virtual reality (VR) or augmented reality (AR). This method is

particularly popular in combination with robot arm manipulators. The 3D environment can be constructed a priori, and exact location of the robot arm is known (Ricks et al., 2004). A limitation arises when tasks are performed in unknown and unstructured areas, and since environment geometry is unknown real time mapping and rendering can be difficult. Additional hardware may be required and calculations can become computationally intensive. Moreover, additional challenges, such as oscillopsia occur when latency is introduced in VR head-mounted displays (Allison et al., 2001).

Video manipulation does not require 3D information about the environment. It alters the delayed video feed to mimic movements and environment in real time. A simple example would be to zoom into the image if the robot is moving forward. Matheson et al. (2013) halved task completion time at a latency of three seconds in an ROV experiment using this method, by cropping and projecting the image. A similar result is obtained by capturing a wide FOV video, possibly 360 degrees, and then only displaying a section of that image to the operator. The section can be moved around in the video as a response to steering commands and thus provide fluid and seemingly real time feedback (Baldwin et al., 1999). Advantageous to video manipulation techniques are low cost, ease of implementation and not requiring a structured environment. Furthermore, prediction error propagation cannot occur since the presented video feed consists only of alterations to the latest image. However, it cannot recreate parallax movement (such as passing an object or corner) which 3D graphics models can achieve.

Table 2.1.2. Predictive technology with task completion time reduction.

Author	Robot system Task	Predictive technology Camera	Participants Latency	Reduction in task completion time
(Lu et al., 2018)	Car simulator Driving	Model-free framework Simulated human	N = 12 Not reported	8%
(Hu et al., 2016)	2-6 DOF manipulator Camera alignment	Simulated 3D Virtual	N = 15 300 ms, 500 ms, 1000 ms	33%, 58%, 65%*
(Zheng et al., 2016)	Car simulator Driving	Model-free framework Simulated human	N = 5 900 ms	35%
(Lovi et al., 2010)	Robot arm on Segway Object alignment	Vision-based monocular modelling At end effector	N = 5 300 ms	33%*
(Matheson et al., 2013)	Rover Driving	Projected field of view Fixed to car	N = 12 3000 ms	48% - 64%*
(Rachmielowski et al., 2010)	Virtual with Phantom OMNI Alignment	Reconstructed 3D environment At end effector	N = 12 300 ms	29% - 30%*
(Mathan et al., 1996)	Lunar vehicle Manoeuvring	Superimposed directional information Fixed to car	N = 8 5000 ms	24% - 30%
(Bejczy et al., 1990)	6DOF PUMA robot Tapping	Superimposed phantom robot Fixed	N = 2 1000 ms, 4000 ms	13% - 34%, 40% - 56%

* Estimated from graph

3 Method - Experiment investigating a predictive display under three conditions

An experiment was set up to investigate changes in human operator performance and workload when operating an ROV under three distinct conditions, each condition with a distinct display and latency. The participants were presented with a single, simple navigational task, framed as a peg-in-hole game which was the same for all three conditions. The conditions were C1. Latency, C2. Latency with Predictive Display (w/PD) and C3. Baseline, and they are described in detail in this section. First, this section describes development and implementation of a predictive display Then, the experiment design follow, which includes research objective, hypotheses, stimuli (description of task and conditions), data collection (objective performance and subjective workload), setup, experimental procedure, and data analysis.

3.1 Predictive display development

Predictive technology that reconstructs a 3D environment based on sensory data requires advanced algorithms, potentially expensive equipment, and extensive information regarding environment and ROV. In cases where this is not a possibility video manipulation provides simple and inexpensive means to increase human operators' performance.

The projected display by Matheson et al. (2013) is the simpler video manipulation method of the ones considered in Table 2.1.2, while retaining a great increase in human operator performance. However, information on the vehicles' ground trajectory is required to calculate changes in perspective. By disregarding the effects of change in perspective and applying positional and scale transformations to the video feed we obtain an even simpler approach. As such, by applying positional and scale transformations to the video feed we developed a predictive display based on image transformation. The predictive display can be applied to several robot configurations though it was developed for ROVs initially; It is appropriate only for screen-based systems and other alternatives are needed for predictive head-mounted display systems.

3.1.1 Predictive display implementation in detail

The developed predictive display repositions the delayed video feed on the monitor so objects in the video feed appear in correct size and position on the screen as if there was no latency (see Figure 1 and Figure 2). It uses user input (i.e. steering commands) and predefined ROV speed to predict how the FOV would move in the scene, repositioning and scaling the video feed accordingly.

The positional transformation can be explained by considering an ROV with an onboard camera rotating about its center of mass, turning with an angular velocity of ω°/s. The camera FOV is φ°, with horizontal resolution R_h pixels. A counterclockwise rotation for Δt s moves the ROV $\Delta\theta$°. Objects in the video feed moves $(R_h\cdot\omega/\varphi)\cdot\Delta t=\eta\cdot\Delta t=\Delta P_h$ pixels to the right. η = *pixel turn rate*, which depends on screen resolution, angular velocity and camera FOV. The pixel turn rate, user input and total system delay t_d is used to create the predictive display. The video feeds' position on the monitor is calculated at a set interval *dt* (preferably at a minimum video frame rate (FPS)). If the ROV moves to the left, time since last update *dt* multiplies with pixel turn rate to find change in horizontal video position ΔP_h. The video feed then moves ΔP_h to the right on the monitor. When a time t_d has passed (system delay has caught up), the video feed is moved back.

For backward and forward translation, similarly as for *pixel turn rate*, a *pixel scale rate* can be found and used to scale the video feed. For backwards and forwards ROV translation, scaling of objects depends on how close they are to the camera. An average distance is used as an approximation. The video feed scale transformation works as the aforementioned positional transformation.

Finally, the predictive display uses a red arrow to visualize the future position as illustrated in Figure 2.

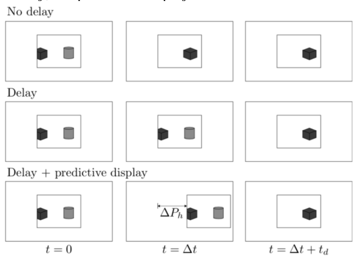


Figure 1. Monitor for the human operator. The outer box is total screen size, whereas the inner box is the video feed.

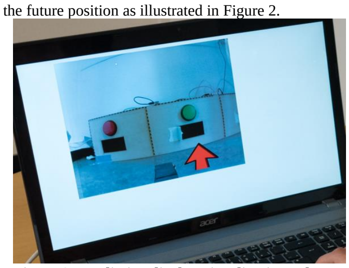


Figure 2. Predictive display visualization. The operator has recently turned the ROV to the right, and as a result the video has moved to the left. The red arrow has not moved and works as an indication of where the ROV will be heading when the video feed has caught up with the time delay.

3.2 Research objective and hypotheses

Research objective: Investigate if such a simple predictive display still increase human operators' performance and reduce workload.

Based on the research objective, we sought to test the following hypotheses:

- A simple predictive display significantly increases human operators' performance (objectively measured by task score performance, - i.e. the number of hits achieved in 90s by the participant).
- A simple predictive display significantly decreases human operators' subjective workload (subjectively measured by RTLX's six dimensions, mental demand, physical demand, temporal demand, performance, effort and frustration, evaluated on eleven-point scales (Hart, 2006)).

3.3 Stimuli – Peg-in-hole-game under three conditions

The experiment encompassed a single, navigational task, in which we measured operator performance by means of an achieved score over a fixed time period.

3.3.1 Rationale behind task selection

Chen et al. (2007) reports benefits of predictive technology to be very task dependent. A peg in hole task was selected due to its applicability in teleoperation (Lane et al., 2002). A task encompassing as much lateral navigation as possible was selected, as this is where the predictive display can provide the most help, in contrast to for example navigational tasks with longer stretches of forward motion (and the maximal velocity of the ROV would create a ceiling effect). A short timeframe of 90 seconds was chosen to reduce any learning effect that might accompany a longer maneuvering course. A fixed time period made total experiment length predictable, participants used 10 min and 56 s on average (SD 1 min and 12 s). This aided in recruiting new participants. Furthermore, time pressure in combination with score achievement made participants fully devoted to the task at hand, and we argue this led to participants performing close to the best of their ability. We further argue that a single, simple task will minimize the effect of other factors on performance, e.g. trouble understanding the task, or being highly experienced in related tasks such as gaming, driving, or other navigational tasks.

3.3.2 Task

Participants were given a modified 'peg-in-hole' task. The peg was mounted on a remotely controlled ground vehicle, and there were three rectangular holes in three rectangular boxes with accompanying LEDs. One LED would light up at a time, in random order, to which the participant was instructed to perform as many 'hits' as possible by inserting the peg in the hole within the given timeframe. Task and time given (90s) was the same for three distinct conditions. During the task, a red timer indicating remaining time was constantly visible in the screens' upper right corner.

3.3.3 Three conditions

All participants repeated the task three times, under three distinct conditions. The display provided to the participants would differ in each condition. The conditions, latency and displays were as follows:

Condition 1. Latency: 700 ms delay (250 ms inherent system delay + 450 added delay). No predictive display.
Condition 2. Latency with Predictive Display (PD): 700 ms delay (250 ms inherent system delay + 450 added delay). With predictive display.
Condition 3. Baseline: 250 ms inherent system delay[1]. No predictive display.

Throughout the paper we refer to the conditions as:

C1. Latency
C2. Latency w/PD
C3. Baseline

3.3.4 3x3 Latin Square Design

The sequence of the conditions was randomized according to a 3x3 Latin Square Design to avoid potential order and/or learning effects. All six combinations were used. Each participant was automatically assigned to one of the combinations, ensuring equal group sizes across conditions as far as possible. Due to the number of participants recruited; three of the combinations had 10 participants, and three combinations had 9 participants.

[1] The variability of inherent system delay was repeatedly quantified (10 times) to 1 – 5 ms difference each time. The average of those 10 measurements was used as inherent system delay.

3.4 Data collection – Performance measured objectively and workload measured subjectively

N = 58 participants were recruited to test the predictive display. We collected objective measures of human performance and subjective measures of workload. Demographic data were also collected.

3.4.1 Participants

Participants were voluntary selected from NTNU, Department of Mechanical and Industrial Engineering. Our aim was to recruit as many participants as possible within the time constraint we were working with. A total of 58 participants performed the experiment, one participant was excluded in the analysis due to incomplete information. The remaining N = 57 participants received the same information and were included in the analysis. Age ranged from 23 to 30 years (24.7 ± 1.5). There were 19 female and 38 men. We gathered level of education, how often they played video games, how often they use a computer and eye health information, which can be found in Table 3.4.1.

Table 3.4.1. Participant data.

Variable	Options	Frequency	Percent
Gaming	Daily	2	3.5
	Weekly	15	26.3
	Monthly	8	14.0
	Yearly	17	29.8
	Never	15	26.3
Education	Nursery school	1	1.8
	Some college credit, no degree	38	66.7
	Bachelor's degree	10	17.5
	Master's degree	8	14.0
Eye health	No visual aid	32	56.1
	Spectacles	4	7.0
	Contact lenses	10	17.5
	Both spectacle and contact lenses	11	19.3

3.4.2 Objective performance measurements

Two performance measurements are common among experiments on predictive technology: course completion time and task score (Lu et al., 2018; Mathan et al., 1996; Matheson et al., 2013; Zhang & Li, 2016; Zheng et al., 2016). In the former, the task is to navigate through a predefined pathway with the vehicle and measuring the time necessary to complete the course. In the latter, the task typically involves aligning or aiming at a given target, assigning a score to the number of times the target was met. Using a task score as a performance measure enables a fixed time for experiments, which was desirable for us to be able to recruit more participants. The number of hits made by participants in each of the 90 s test period was used as a performance measure.
Additional objective data collected included total number of hits made in all three test periods, and number of key presses in each of the test periods.

3.4.3 Subjective workload measurements

NASA Task Load Index (TLX) is common and highly accepted for remote operation and ROV applications (Hart, 2006; Hill et al., 1992; Hu et al., 2016; Ma & Kaber, 2006; Zhang & Li, 2016), and was initially developed for experimental tasks that include cognitive and manual control tasks, and supervisory control tasks (Hart & Staveland, 1988b). TLX is multidimensional, provide good diagnostic properties for assessing underlying mechanisms of subjective workload, and has been shown to have high sensitivity (Hart, 2006; Hendy et al., 1993; Hill et al., 1992; Vidulich & Tsang, 1987). A modified version of TLX, Raw TLX (RTLX) was chosen to assess workload. The six dimensions (mental demand, physical demand, temporal demand, performance, effort and frustration) were rated on eleven-point scales. The weighting process in TLX consists of pairwise comparison of all six dimensions. It was not conducted, since we are not interested in the subjective importance of each dimension in a specific task, rather we're interested in comparing the subjective workload of different tasks (the three conditions). Furthermore, this weighing process consumes time, and, in this context, it was deemed more important to have a short survey, leaving more time for recruiting participants and conducting experimental runs. This

modification is what is referred to as RTLX. One additional modification was made to the survey, as a pilot study of the experiment showed that a participant found it more intuitive to rate good performance with a high number. In the original survey a low value corresponds to good performance. Therefore, this metric and the corresponding description was reversed, such that a high value corresponded to good performance. After data collection, this value was reversed back for conventional analysis and reporting.
Furthermore, a question of perceived delay time was added to the survey, to investigate participants' subjective experienced latency in each individual condition and to compare the individual conditions, the latter in hopes of providing a measure of effectiveness for the predictive display in reducing the subjectively perceived latency of the system.

3.4.4 Data collection procedure

Both survey data and experiment data were recorded with the ROV computer using an SQLite database.

3.5 Setup

A 17" laptop running a 2.3GHz Intel Core i7-3610QM CPU and Windows 10 was used. The laptop screen served as monitor and the keyboard's arrow keys were used to steer the ROV. The keyboard and a remote mouse were used to answer the surveys. The ROV was running a Raspberry Pi 3 Model B+, and equipped with a forward facing Raspberry Pi Camera V2 and a wide angle lens with horizontal FOV of 76.5°. The robot was constructed using three wheels, two of them connected to a DC motor and the third a caster wheel for support (see Figure 4). A wooden box with three holes and LEDs were used to register task performance. The distance between the holes (center to center) was D = 30 cm while the holes itself has a width of W = 10cm. This translates to a Fitts's index of difficulty of Id = log2 (2D/W) = 2.58 bits (Fitts, 1954). The robot ran eduROV[2] software, which provided an interface to control the robot, handling control commands, adding desired latency to the communication, and logging data.

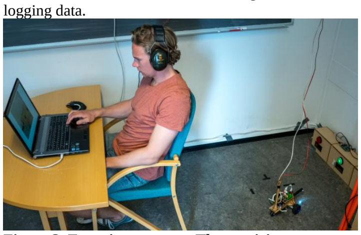


Figure 3. Experiment setup. The participant can only see the robot through the display provided on the laptop screen, which is a first-person camera view.

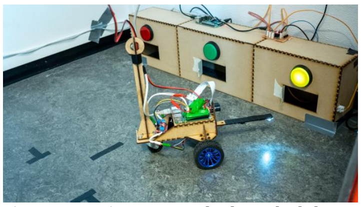


Figure 4. Experiment setup. The three wheeled ROV with the peg mounted and the wooden box.

3.6 Experimental procedure

After entering the experiment room, participants were shown the setup to ensure that they understood the situation and what they were tasked to do. The participant was placed in a chair at a desk with a laptop, with their back to the game (see Figure 3). The participant would have no visual perception of the physical setup during the experiment. To ensure there was no auditory perception of the ROV, participants wore an ear protection headset. Information was given in writing on the computer screen. After giving consent to participate in the experiment, participants filled out a demographic survey. Information describing the experiment was provided; How to steer the vehicle, the task and performance measure, and the following procedure of the experiment. Each participant was automatically assigned to one of the groups corresponding to the 3x3 Latin Square Design. The participant would then conduct a 30s practice period followed by a 90s test period. After each block of practice and test period the participant filled out a survey of mental workload and perceived delay time. The starting position (indicated by the black mark in Figure 3 and Figure 4) was identical for all periods. The third block concluded the experiment and the participants were escorted out.
The participant was not informed of the fact that one of the conditions would have a predictive display, nor how it worked. To be able to take advantage of the predictive display is therefore dependent on the individual participants

[2] https://github.com/trolllabs/eduROV/

ability to intuitively understand the display. It was assumed that the practice period before each test would suffice in giving the participant the needed training in the display for the game. However, the questionnaire included a question of time delay, which could have influenced participant's attention to delay in the next conditions.

3.7 Analysis – Classical statistics and exploratory multivariate analysis

3.7.1 Classical statistics – Analysis of variance (ANOVA)

Subjective measurements used for analysis were collected after each condition and performance measurements were collected continuously during each condition. An analysis of variance (ANOVA) was conducted to investigate the effects of the predictive display on both subjective and objective measurements, i.e. this statistical test investigated the predetermined hypotheses. The characteristics of the data was inspected and in the case of violations of assumptions, the non-parametric alternative to one-way repeated measures ANOVA, the Friedman test was conducted. Data distribution was visually inspected using Normal Q-Q Plots for all variables and conditions. ANOVA F-test is found to be insensitive or robust (Krishnaiah, 1980; Schmider et al., 2010) to general nonnormality, and can for equal group sizes be used with confidence in most practical situations. We consider the sample size of 57 to be high, and we have continued with the analysis and when possible conducted a Friedman test for comparison purposes. Mauchly's test evaluates sphericity, an assumption which is considered difficult not to violate in practice (Weinfurt, 2000), over-detecting deviations from sphericity in large samples (Kesselman et al., 1980). Maxwell & Delaney (2003), recommend using an adjusted test, interpreting the result of using a Greenhouse-Geisser correction and thus ignoring the result of Mauchly's test. This was done here, calculating epsilon according to Greenhouse & Geisser (1959), and using it to correct the one-way repeated measures ANOVA. The Bonferroni post hoc test (Maxwell, 1980, Maxwell &Delaney 2004) was used to test all possible pairwise combinations of conditions. Statistical tests were performed using SPSS Statistics (*IBM SPSS Statistics 25*, 2017).

3.7.2 Exploratory data analysis

Furthermore, we wanted to explore and hypothesize regarding other potential relationships between the variables. Therefore, an exploratory data analysis (EDA) (Tukey, 1977), specifically a principal component analysis (PCA) was conducted to explore whether there were any interesting patters or observations in the data collected. Here, no hypothesis was determined, and all effects described emerged post-hoc. The PCA was conducted using scikit-learn (Pedregosa et al., 2011) and Jupyter Lab Notebook (Kluyver et al., 2016).

4 Results

The following section presents the results of the statistical tests before providing the result from the EDA.

4.1 Performance (objective)

A one-way repeated measures ANOVA was conducted to determine whether differences in human performance (the number of hits, an objective measure) between the three conditions were statistically significant. Descriptive statistics of performance data are illustrated in Figure 5, and Table 4.1.2 shows all pairwise comparisons of the conditions. Table 4.1.1 contains the ANOVA F-test statistic, data characteristics and pretests.
Performance was statistically significant different in the three conditions, with a performance increase of 20% from C1. Latency to C2. Latency w/PD. Performance increased from *M = 6.2 hits* in C1. Latency, to *M = 7.5 hits* in C2. Latency w/PD, to *M = 16 hits* in C3. Baseline. There was a statistically significant increase in performance of *M* = 1.3 hits (*SD = 0.26*) from C1. Latency to C2. Latency w/PD. In summary, there was a statistically significant difference between means and, therefore, we accept the alternative hypothesis; The predictive display significantly increases performance of the human operator.

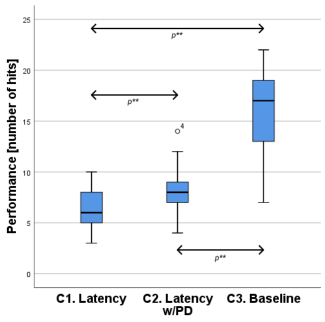


Figure 5. Descriptive statistics of performance (objective). Original data reported. Statistically significant differences at p<0.01 are indicated by *p*.**

Table 4.1.1. One-way repeated measured ANOVA F-test for Performance (objective)

Variable Performance	N	Outliers	Normality	Sphericity	Epsilon (ε)	F-statistic	Sig.	Sample effect size	Population effect size [c]
Performance [number of hits]	57	Yes (1)[a]	Approx. normal[b]	, χ2(2) = 25.7, p <0.0001	0.728	*F*(1.46, 81.54) = 316.34	p<.0001**	$\eta^2 = 0.850$	$\omega^2 = 0.787$
(outlier removed)	56	No	Approx. normal[b]	χ2(2) = 26.4, p <0.0001	0.721	*F*(1.44, 79.32) = 308.69	p<.0001**	$\eta^2 = 0.849$	$\omega^2 = 0.786$

*: p < 0.05, **: p < 0.01

a) There was one outlier in C2. Latency w/ PD, as assessed by visual inspection of a boxplot. SPSS Statistics defines outliers as values greater than 1.5 box-plots from the edge of the plot. This value (14 hits) is genuinely unusual, we know from the experiment and the data that this participant performed above average in all three conditions. We ran the analysis both with and without outliers, reporting both results.

b) Visual inspection of Normal Q-Q Plots and histograms for all three conditions

c) Calculated according to (Wickens & Keppel, 2004).

Table 4.1.2. Pairwise comparisons of performance (objective).

Variable	C1. Latency – C2. Latency w/PD			C1. Latency – C3. Baseline			C2. Latency w/PD – C3. Baseline		
	Mean Diff.	SD	Sig.[b]	Mean Diff.	SD	Sig.[b]	Mean Diff.	SD	Sig.[b]
Performance [number of hits]	-1.298	0.264	*p* < 0.0001**	-9.754	0.491	*p* < 0.0001**	-8.456	.471	*p* < 0.0001**

b: Adjustment for multiple comparisons: Bonferroni.

*: p < 0.05, **: p < 0.01

4.2 Subjective workload

This section presents the results from statistical analysis of subjective workload measures. Overall Subjective Workload is presented first, before also presenting the individual workload dimensions.
Since we conducted RTLX, the values of the individual workload dimensions (mental, physical, temporal, performance, effort and frustration) were averaged to obtain an estimate of the overall workload (Hart, 2006). This averaged score is addressed as Subjective Overall Workload in the following. Separate one-way repeated measures ANOVA was conducted for overall workload and the six individual workload dimensions to determine the effects of the predictive display on lowering subjective workload in the three conditions. The results from the ANOVA

F-test, including pretests for all variables can be found in Table A2, Appendix A, whereas descriptive statistics and pairwise comparisons can be found in Table A1, Appendix A, and Table 4.2.1, respectively. The following paragraphs describe individual results before providing an overall explanation.

4.2.1 Subjective Overall Workload

Subjective Overall Workload was statistically significant different under the three conditions. There was a decreased subjective workload from *M = 5.3 (SD = 0.2)* in C1. Latency, to *M = 4.9 (SD = 0.2)* in C2. Latency w/PD, to *M = 3.5 (SD = 0.2)* in C3. Baseline. Pairwise comparisons of the three conditions was carried out using the Bonferroni post hoc test, which revealed that the mean decrease in subjective workload from C1. Latency to C2. Latency w/PD was not statistically significant (*M = 0.35, SD = 0.16, p = 0.133*). There was a statistically significant mean decrease in subjective workload from C2. Latency w/PD to C3. Baseline (*M = 1.44, SD = 0.18, p < 0.001*), and from C1. Latency to C3. Baseline (*M = 1.775, SD = 0.14 p < 0.001*). A Friedman test produced corroborating results. Therefore, we cannot reject the null hypothesis and cannot accept the alternative hypothesis. The predictive display does not decrease human operators' subjective overall workload.

4.2.2 Mental demand (Individual workload dimension)

Mental demand was statistically significantly different in the three conditions, however, post hoc analysis with a Bonferroni adjustment revealed that mental demand did not significantly decrease from C1. Latency to C2. Latency w/PD. There was a statistically significant decrease in mental demand from C1. Latency to C3. Baseline and from C2. Latency w/PD to C3. Baseline. The predictive display did not reduce participants' mental demand.

4.2.3 Physical demand (Individual workload dimension)

Physical demand was statistically significantly different in the three conditions, however, post hoc analysis with a Bonferroni adjustment revealed that physical demand did not significantly decrease from C1. Latency to C2. Latency w/PD. A Friedman test with pairwise comparisons using a Bonferroni correction for multiple comparisons was carried out for comparison purposes, which gave the same result. There was a statistically significant decrease in physical demand from C1. Latency to C3. Baseline, and from C2. Latency w/PD to C3. Baseline. The predictive display did not reduce participants' physical demand.

4.2.4 Temporal demand (Individual workload dimension)

Temporal demand was not statistically significantly different in the three conditions, according to both ANOVA and Friedman test. The predictive display did not reduce participants' temporal demand.

4.2.5 Subjective Performance (Individual workload dimension)

Subjective Performance was statistically significantly different in the three conditions. Subjective Performance was evaluated at *M = 5.53* in C1. Latency, *M* = 4.74 in C2. Latency w/PD, and *M* = 2.70 in C3. Baseline, with a low value corresponding to a performance closer to perfect. There was a statistically significant decrease of *M = 0.79 (SD = 0.24, p = 0.006)* between C1. Latency and C2. Latency w/PD, a statistically significant decrease of *M = 2.83 (SD = 0.24, p < 0.001*) between C1. Latency to C3. Baseline, and a statistically significant decrease of *M = 2.04 (SD = 0.21, p < 0.001)* between C2. Latency w/PD to C3. Baseline. A Friedman test with a Bonferroni correction for multiple comparisons was carried out for comparison purposes, corroborating result at *p < 0.001*. The median of Subjective Performance was statistically significant different between C1. Latency (*Mdn* = 5) and C3. Baseline (*Mdn* = 2) (*p* < 0.001), statistically significant between C2. Latency w/PD (*Mdn* = 5) and C3. Baseline (*p* < 0.001), but not statistically significant different between C1. Latency condition and C2. Latency w/PD (*p* < 0.132). In addition to a statistically significant decrease from both latency conditions (C1. Latency and C2. Latency w/PD) to C3. Baseline, it is also noteworthy that the mean decrease towards C3. Baseline is greater from C1. Latency than the decrease from C2. Latency w/PD; Which means participants thought they performed better with the predictive display than without it, given different latencies, and given equal latency. In summary, the predictive display increased participants subjective performance, i.e. participants thought their performance was better with the predictive display.

4.2.6 Effort (Individual workload dimension)

Effort was statistically significant different in the three conditions, however, post hoc tests with a Bonferroni adjustment revealed that there was not a statistically significant difference between C1. Latency and C2. Latency w/PD. There was a statistically significant decrease from C3. Baseline to the two latency conditions (C1. Latency and C2. Latency w/PD). A Freidman test with a Bonferroni correction for multiple comparisons corroborated these results. The predictive display did not reduce participants' effort.

4.2.7 Frustration (Individual workload dimension)

Frustration was statistically significantly different in the three conditions and post hoc tests with a Bonferroni adjustment revealed that there was a statistically significant decrease in frustration from C1. Latency to C3. Baseline, as well as from C2. Latency w/PD to C3. Baseline, though not from C1. Latency to C2. Latency w/PD. The predictive display did not reduce participants' frustration.

4.2.8 Overall result for workload

The analysis of Subjective Overall Workload and the individual workload dimensions did not show a statistically significant difference between C1. Latency and C2. Latency w/PD, with the exception of the individual variable Subjective Performance, in which participants reported a statistically significant mean increase of 0.789.
The predictive display does not reduce participants' mental demand, physical demand, temporal demand, effort, nor frustration; However, the predictive display increased participants' subjective performance, i.e. participants' thought they performed better with the predictive display. In summary for subjective workload, we cannot reject the null hypothesis, i.e. the predictive display does not reduce participants subjective workload.

Table 4.2.1. Pairwise comparisons Subjective variables.

Variable	C1. Latency – C2. Latency w/PD			C1. Latency – C3. Baseline			C2. Latency w/PD –C3. Baseline		
	Mean Diff.	**SD**	**Sig.**[b]	**Mean Diff.**	**SD**	**Sig.**[b]	**Mean Diff.**	**SD**	**Sig.**[b]
Subjective Overall Workload	0.336	0.163	*p* = 0.133	1.775	0.141	*p* = 0.000**	1.439*	0.177	*p* = 0.000**
Mental Demand 0-10	0.158	0.235	*p* = 1.000	2.105	0.226	*p* = 0.000	1.947*	0.306	*p* = 0.000**
Physical Demand 0-10[f]	0.035	0.221	*p* = 1.000	0.702	0.227	*p* = 0.009	.667*	0.211	*p* = 0.008**
Temporal Demand 0-10	0.175	0.221	*p* = 1.000	0.456	0.236	*p* = 0.176	.281	0.288	*p* = 1.00
Subjective Performance 0-10	0.789	0.244	*p* = 0.006*	2.825	0.240	*p* = 0.000	2.035	0.212	*p* = 0.000**
Effort 0-10[f]	0.246	0.234	*p* = 0.894	1.351	0.213	*p* = 0.000	1.105	0.241	*p* = 0.000**
Frustration 0-10	0.679	0.283	*p* = 0.059	3.179	0.304	*p* = 0.000	2.500	0.306	*p* = 0.000**

b: Host hoc Pairwise comparisons were adjusted for Bonferroni

f: A Friedman test with pairwise comparisons using a Bonferroni correction for multiple comparisons was carried out for comparison purposes. Results were corroborated.

*: p < 0.05, **: p < 0.01

4.3 Subjective latency

A one-way repeated measures ANOVA was conducted on Subjective Latency to determine if there was a statistically significant reduction from C1. Latency – to C2. Latency w/PD condition. Subjective latency (evaluated in ms by the participants) was statistically significantly different in the three conditions. Post hoc tests with a Bonferroni adjustment revealed that there was a statistically significant decrease in Subjective Latency from C1. Latency to C3. Baseline condition as well as from C2. Latency w/PD to C3. Baseline condition, but not from C1. Latency to C2. Latency w/PD. A Friedman test corroborated these results.
There were multiple outliers for this measure, and we reran the analysis with the extreme outliers removed, which only slightly increased effect size, however it did not change the overall result. The predictive display did not reduce participants' estimation of latency when comparing C1. Latency and C2. Latency w/PD.

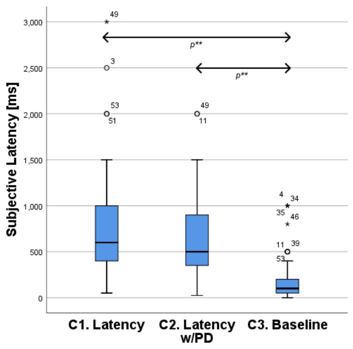


Figure 6. Descriptive Statistics Subjective Latency. Original data reported. Statistically significant differences at p<0.01 are indicated by *p*.**

Table 4.3.1. One-way repeated measured ANOVA F-test Subjective Latency.

Variable	N	Outliers	Normality	Sphericity	Epsilon (ε)	F-statistic	Sig.	Sample effect size	Population effect size[c]
Subjective latency	57	Yes (13)[a] (10 unique)	Approx.[d]	Yes χ2(2) = 5.575, p = 0.062	-	F(2,112) = 45.734	p < 0.001**	$\eta^2 = 0.450$	$\omega^2 = 0.343$
(5 extreme outliers removed)	52	4 unique	Approx.[d]	Yes χ2(2) = 3.617, p = 0.164	-	F(2,102) = 44.684	p < 0.001**	$\eta^2 = 0.467$	$\omega^2 = 0.359$

*: p < 0.05, **: p < 0.01

a) Number of outliers in parentheses. There was no reason to exclude any outliers and so they were kept in first analysis. In the second, 5 extreme outliers were excluded, which yielded a dataset with 4 unique outliers. Further reduction did not yield a dataset without outliers. Both results are reported here.

Table 4.3.2. Pairwise comparisons Subjective Latency.

Variable	C1. Latency – C2. Latency w/PD			C1. Latency – C3. Baseline			C2. Latency w/PD – C3. Baseline		
	Mean Diff.	SD	Sig.[b]	Mean Diff.	SD	Sig.[b]	Mean Diff.	SD	Sig.[b]
Subjective Latency [ms]	78.33	68.42	*p* = 0.771	582.21	73.14	*p* = 0.000**	503.88	55.37	*p* = 0.000 **

b: Host hoc Pairwise comparisons were adjusted for Bonferroni

*: p < 0.05, **: p < 0.01

4.4 Gamers

Those who play games weekly or more often were defined as gamers. The potential increased performance gain, measured objectively, by gamers is investigated here. A two-way mixed ANOVA was conducted, comparing the mean differences of performance (objective) between two independent groups, Gamers and Non-Gamers, under the three conditions. Descriptive statistics can be found in Table 4.4.1 and Figure 7.

Table 4.4.1. Descriptive statistics Gamer vs Non-Gamer. Original data reported.

Variable	Group	N	C1. Latency		C2. Latency w/PD		C3. Baseline	
			Mean	SD	Mean	SD	Mean	SD
Performance [number of hits]	Gamer	17	6.47	0.42	8.41	0.46	17.71	0.92
	Non-Gamer	40	6.08	0.27	7.10	0.30	15.20	0.60

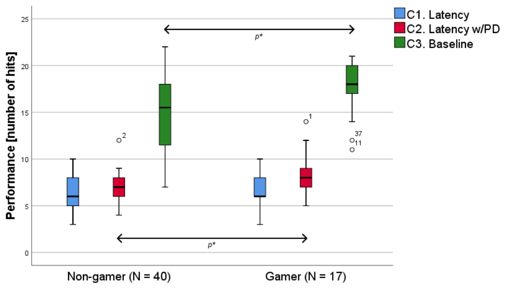


Figure 7. Performance of Gamers vs. Non-gamer. Original data reported. Statistically significant differences at p<0.01 are indicated by *p*.**

The interaction between gaming experience and conditions on performance had a level of significance of *p = 0.088.* Univariate post hoc tests indicated that there was not a statistically significant difference between gamers (*M = 6.5 hits*) and non-gamers (*M = 6.1 hits*) in C1. Latency (*F(1,55) = 0.622, p = 0.43*, sample effect size $\eta^2 = 0.01$*)*. However, there was a significant increase in performance for gamers in C2. Latency w/PD (*F(1,55) = 5.71, p = 0.02,* sample effect size $\eta^2 = 0.094$), in which gamers had *M = 8.4 hits*, whereas non-gamers had *M = 7.1 hits*. Furthermore, there was a significant increase in performance for gamers in C3. Baseline (*F(1,55) = 5.203, p = 0.026*, sample effect size $\eta^2 = 0.086$), in which gamers had *M = 17.7 hits*, whereas non-gamers had *M = 15.2 hits.* When considering the two independent groups (Gamer, Non-Gamer), there was a significant main effect of gaming (*F(1,55) = 6.311, p = 0.015,* sample size effect $\eta^2 = 0.103$), with gamers performing better than non-gamers. Gamers performed on average *M = 10.9 hits,* which is *M =1.4 hits (SD = 0.6)* above the performance of non-games with *M = 9.5 hits.*
The analysis was also conducted without outliers, which yielded more than a doubling of effect size (sample effect size $\eta^2 = 0.102$ and population effect size $\omega^2 = 0.057$), and a lower p value (p = 0.009), which means that the interaction between gaming experience and conditions on objective performance reached statistical significance. Univariate post hoc tests (on the pruned dataset) indicated a statistically insignificant difference between gamers (*M = 6.3 hits*) and non-gamers (*M = 6.0 hits*) in C1. Latency (*F(1,51) = 0.240, p = 0.63*, sample effect size $\eta^2 = 0.005$*)*, and a statistically insignificant difference increase in performance for gamers in C2. Latency w/PD (*F(1,51) = 3.52, p = 0.066*, sample effect size $\eta^2 = 0.065$), in which gamers had a *M = 7.9 hits*, whereas non-gamers had *M = 6.0 hits*. There was a significant increase in performance for gamers in C3. Baseline (*F(1,51) = 8.249, p = 0.006*, sample effect size $\eta^2 = 0.1.39$), in which gamers achieved *M = 18.4 hits*, whereas non-gamers had *M = 15.1 hits.* When considering the two independent groups (Gamer, Non-Gamer), there was a significant main effect of gaming (*F(1,51) = 6.929, p = 0.011,* sample size effect $\eta^2 = 0.12$), with gamers performing better than non-gamers. Gamers performed on average *M = 10.9* hits, which is *M =1.5 hits (SD = 0.6)* above the performance of non-games with *M = 9.4 hits.*
Gamers performed better than non-gamers on average.

Table 4.4.2. Two-way mixed ANOVA F-test on Performance (objective) for Gamers vs. Non-gamer.

Variable	N	Outliers	Normali ty	Homogen eity	Sphericit y	Epsilo n	F-statistic	Sig.	Sample effect size	Population effect size [g]
Performance[a] [number of hits]	57	Yes (4)[bc]	Yes[d]	Yes[e]	No $\chi^2(2)$ = 24.895, p = .000	ε = 0.728	F(1.46, 80.32) = 2.72[f]	*p* = 0.088	η^2 = 0.047	ω^2 = 0.02
(outliers removed)	53	No	Yes[d]	Yes[e]	No $\chi^2(2)$ = 21.406, p = .000	ε = 0.742	F(1.48, 75.65) = 5.769[f]	*p* = 0.009**	η^2 = 0.102	ω^2 = 0.057

*: p < 0.05, **: p < 0.01

a) Performance was separated for the two independent groups.

b) Assessed by visual inspection of a boxplot. SPSS Statistics defines outliers as values greater than 1.5 box-plots from the edge of the plot. The 4 outliers were kept in the first analysis as there was no reason to exclude them. In the second, they were excluded. Both results are reported here.

c) By examination of studentized residuals for values greater than ±3, one outlier was found with a studentized residual value of 3.04. The outlier was kept in the subsequent analysis since its value is close to the threshold and as there was no reason to exclude it.

d) Visual inspection of Normal Q-Q Plots of the distribution and the distribution of studentized residuals for all three conditions.

e) Levene's test assessed homogeneity of variance, and Box's test evaluated homogeneity of covariances.

f) A Greenhause Geisser correction was applied.

g) Calculated according to (Wickens & Keppel, 2004).

Table 4.4.3. Pairwise comparisons of differences in Performance (objective) for Gamer vs. Non-Gamer.

Variable Performance [number of hits]	C1. Latency			C2. Latency w/PD			C3. Baseline		
	Mean Diff.	SD	Sig.[b]	Mean Diff.	SD	Sig.[b]	Mean Diff.	SD	Sig.[b]
Gamer – Non-Gamer	.396	.501	*p = 0.434*	1.312	.549	*p = 0.020**	2.506	1.099	*p = 0.026**
(outliers removed)	0.260	0.53	*p = 0.626*	0.954	0.509	*p = 0.066*	3.300	1.149	*p = 0.006***

b: Host hoc Pairwise comparisons were adjusted for Bonferroni

*: p < 0.05, **: p < 0.01

4.5 Exploratory data analysis - PCA

A principal component analysis (PCA) was conducted to explore whether there were any interesting patters or observations in the data collected. We had no predetermined hypothesis, and all effects described in this section emerged post-hoc.

A total of 35 variables collected during the experiment were standardized (removing the mean and scaling to unit variance) and used in the PCA. Figure 8 shows a Scree plot of the Principal Components (PCs) eigenvalues. The first 10 eigenvalues are larger than 1, the first 5 have an eigenvalue above 2, the first two are greater than 4 and the first eigenvalue is greater than 7. Figure 9 shows the cumulative sum of explained variance, which did not have a clear 'elbow-shape', however the first 7-10 PCs retains 72.1% – 82.8%[3] of the variance of the original data.

[3] Accurate percentage of explained variance retrieved from data, and not estimated from graph.

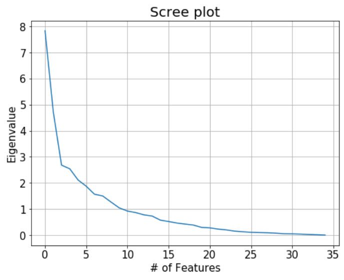


Figure 8. Scree plot.

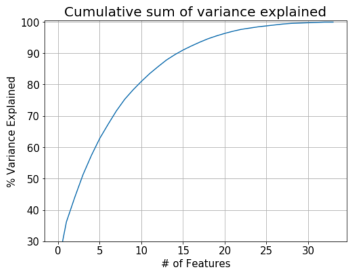


Figure 9. Cumulative sum of variance explained.

A Score plot, and a Loading plot of the two first Principal Components (PCs) can be found in Figure 10 and Figure 11, and Figure 12 respectively. The first two PCs explains 22.6% and 13.6% of the total variance. The following result emerged post-hoc, and thus interpretations made accordingly.

4.5.1 Interpreting the Score plot

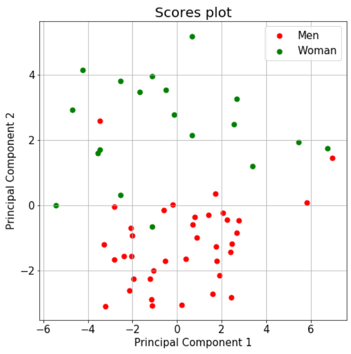


Figure 10. Score Plot of the first two principal components gender.

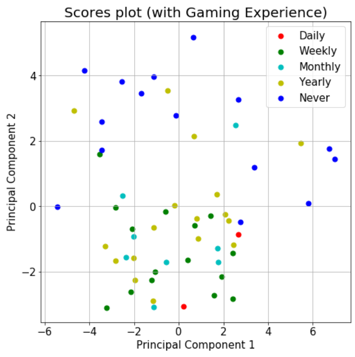


Figure 11. Score Plot of the first two principal components with gaming experience.

Figure 10. shows the Score Plot of PC1 and PC2 with legends indicating gender, from which we can see that PC2 tend to separate woman and men quite accurately. The woman cluster in the positive range of PC2 and the men in the negative range, with only a few datapoints crossing zero. Figure 11 show the Score plot with gaming experience, in which we observe subtle trends in the scatterplot based on gaming experience; Those who never gamed predominantly resides in the positive range of PC2; Furthermore those who games more often tended to cluster in the negative range of PC2. When viewing Figure 10 and Figure 11 simultaneously we observe that the women in this experiment typically gamed yearly or never, with two exceptions of woman gaming on a monthly basis. Men gamed most often, typically yearly, monthly, weekly and two participants daily.

4.5.2 Interpreting the Loading plot

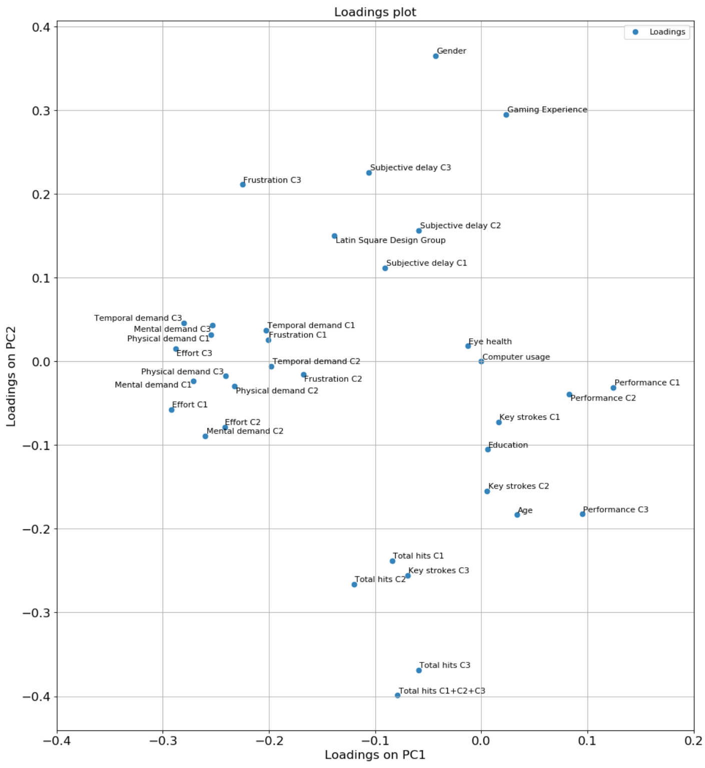


Figure 12. Loadings plot.

From the loading plot in Figure 11, we see that the total hits in each of the conditions seem to be correlated as they cluster together. Total hits in each condition (Total hits C1 – C3) and total hits for all conditions combined (Total hits C1 + C2 + C3) cluster together, as does subjective performance (Performance C1 – C3). We observe that eye health and computer usage have a loading close to zero, thus not contributing to the definition of the principal components, and unimportant for defining the direction of some underlying latent variable. All participants used a computer daily, thus this variable had the same value across the participant population. The eye health, level of education, key strokes in C1. Latency (Key strokes C1) and C2. Latency w/PD (Key strokes C2), and age also have a loading close to zero and are less important for the model.
The subjective performance is negatively correlated with the other TLX dimensions, especially noteworthy is the opposite positions of subjective performance (Performance C1 – C3) and frustration (Frustration C1 – C3). The hits in each condition, and the total hits for all three conditions are clustered together and are therefore correlated. Furthermore, we see that gender and gaming experience have a high loading on PC2, thus contributing greatly to defining PC2 (that gender contributed to PC2 we also knew from the scores plot), and that they are positioned quite close together in comparison to the other variables and therefore are correlated. Among these participants men tended to game more than woman, which is also reflected when looking at the raw data.

5 Discussion

5.1 Performance

The results show that there is a statistical difference in performance when controlling the ROV without and with the help of the predictive display. Subjects performed on average 20% better with a sample effect size $\eta^2 = 0.850$, and population effect size of $\omega^2 = 0.787$. This can be categorized as a medium to large effect (Kirk, 2013), especially when considering the simplicity and low cost of implementing the predictive display. Previous research describes a wide range (8% to 65%) of task time reduction from predictive technology. A direct comparison to any specific experiment is challenging, however a performance increase of 20% in this experiment is probably in the lower range of what was found in the other experiments in Table 2.1.1. The task time reduction measure is considered to be comparable to the performance gain measured in this experiment. However, the predictive method used here is the simplest solution to implement at the lowest cost. Moreover, the participants only had 30 seconds to intuitively learn and train in using the predictive display, since none of the participants were told that there would be a predictive display nor how it worked. Some immediately understood what the predictive display was trying to tell them, others did not understand that there had been a predictive display until the experiment was over. The ones who tried to use the predictive display the way it was intended typically performed better than those who did not use it. It may be that performance could have been improved more if participants were informed about the predictive display's functionality.
As expected, participants performed significantly better in the baseline condition, in which there was only 250 ms latency. This latency is well above what human perception is able to pick up on, which most participants did. As discussed under 5.4, most participant underestimated the latency in the third condition reporting barely above 0 ms.

5.2 Subjective workload

Subjects reported minimal differences between C3. Baseline, C1. Latency and C2. Latency w/PD. There was a statistically significant difference in subjective overall workload between the three conditions, however Bonferroni post hoc tests revealed that differences between C1. Latency and C2. Latency w/PD was not statistically significant. Therefore, we cannot say that the predictive display reduces subjective workload. The only significant difference was found in subjective performance, in which participants felt that they on average performed 14% better when using the predictive display. The actual performance increase was 20%. They also reported that they felt 11% less frustrated using the PD, though this is not statistically significant. Participants also stated that C3. Baseline was better in all metrics, with an exception of temporal demand where the difference was not significant. Participants reported no significant difference in mental, physical and temporal demand between C1. Latency and C2. Latency w/PD. We consider these three metrics to be a good description of the total subjective workload in this experiment setup. Some participants, especially those who did not understand what the predictive display was trying to tell them, even reported it as distracting. Due to the predictive display's functionality, the video feed is constantly moving around and scaling up and down. This can understandably be distracting. Some participants immediately understood how the predictive display worked, and they typically reported the predictive display as helpful. To the experimenter they also seemed to be more relaxed, however there are no recorded data illustrating this. During the task, a red timer indicating the remaining time was constantly visible for participants to see in the upper right corner. In addition, the ROV had rapid acceleration and was able move fast if the operator managed to do so. Overall, this made for a hectic and exiting experience for the subjects. This may explain why there is no significant change in the temporal demand, even compared to C3. Baseline. The fact that the participants reported a better value (smaller) in the other five metrics for the no delay condition, is as expected. The experimenter also observed a tendency of participants performing correcting steering commands, causing the ROV to oscillate greatly before hitting or missing the target, which corroborates prior research (Appelqvist et al., 2007). This was particularly prominent in the C1. Latency condition, again illustrating the detrimental effect of latency on both human performance and behavior. These findings support earlier research describing how video latency negatively affects the user experience in teleoperation.

5.3 Gaming

The gamers performed 30% better with the predictive display, while non-gamers performed 17% better. Interestingly the gamers increased their score almost twice as much as non-gamers when shifting from C1. Latency to C2. Latency w/PD, though the exact reason for this is unclear. The arrow in the predictive display acts as an aiming device, which could be a more familiar concept for gamers. This finding could also indicate that gamers are more used to having to adapt to unfamiliar setting and interfaces in a computer competing context. Furthermore,

when comparing the scores of gamers and non-gamers, it is interesting to note that gamers only performed better than non-gamers in C2. Latency w/PD and C3. Baseline, but not in C1. Latency. This could indicate that the amount of experience may not be crucial for obtaining a high score (equal to high performance) in a situation with considerable latency. Thought post hoc tests on the pruned dataset were not statistically significant at *$p<0.05$* in C2. Latency w/PD, the level of significance *$p = 0.066$* was close to that threshold. A level of statistical significance may have been achieved with additional participants conducting the experiment, and equal group sizes, as both may have a large effect on p-values (Krishnaiah, 1980). In both analysis, there was a significant main effect of gaming, meaning gamers performed better on average. More interesting is the population effect size, which increased from *$\omega^2 = 0.02$*, a small association to *$\omega^2 = 0.057$*, a medium association (Kirk, 2013), which means that the effect of gaming, and the ability gamers had to take advantage of the PD, reaches some practical significance. Taken together, we interpret this to mean gamers were better able to take advantage of the predictive display to increase objective performance.
We observe that the combination of predictive display and related training (in the form of playing similar games at least once a week) results in twice a performance gain compared to only predicative display. In this experiment participants were not informed of the predictive display's functionality, which leads us to consider what the performance gain might have been if participants' were aware of the functionality a priori and if they received training in using the predictive display. Simultaneously considering an increased effect size when removing outliers, i.e. a stronger result, leads us to believe that a greater performance gain might have been the result of specialized training prior to the experiment. Therefore, we hypothesize that the combination of predictive display and extensive training produces a greater increase in performance. Research corroborates this; A priori gaming experience have been found to relate to performance in desktop and immersive virtual environments (Richardson et al., 2011), and video gaming suggested a s a training regimen to increase processing speed, which contributes to increased cognitive performance (Dye et al., 2009). Moreover, studies investigating causality supports action video gaming as a training method (Dye et al., 2009; Green & Bavelier, 2003; Richardson et al., 2011). Generally, we hypothesize that assistive technology in combination with (potentially minimal) training produces high performance gain (output). When compared to the necessary implementation of technology and training (input), we consider this a good trade-off between input and output.

5.4 Subjective latency

About 75% of the participants underestimated the latency in the third condition. Many of them barely reported over 0 ms, but the actual latency was 250 ms. These findings support previous research, which states that smaller latencies closer to zero is difficult to differentiate from no latency. Questioning participants about latency could have influenced their attention to latency in the forthcoming conditions. However, the randomized Latin Square Design of conditions should account for any order effects caused by this question. Furthermore, this question was primarily included to investigate whether participants experienced lower latency with the aid of the predictive display when comparing conditions with equal latencies, which was not the case. The predictive display did not decrease the subjectively experienced latency for participants in this experiment.

5.5 Exploratory data analysis discussion

Effects discussed here emerged post hoc; Thus, is interesting to see effects of gender and gaming experience show up in the PCA, since there are known effects of both. From the scores plots (Figure 10 and Figure 11) we see that PC2 separates women and men quite accurately with a few exceptions. Furthermore, PC2 tends to separate participants by their gaming experience, and by combining the loadings plot (Figure 11) and scores plot (Figure 10 and Figure 11) we observe that the male participants, the exceptions in the upper regions of PC2, never gamed. When further investigating the loadings plot (Figure 11) and scores plots (Figure 10 and Figure 11) simultaneously we see that gender and gaming both had high loadings on PC2, thus contributing to PC2. In the loadings plot (Figure 11) we see participants objective performance (Total hits C1 – C3) having a high negative loading, which means it also contributes to the definition to PC2. Males are generally more experienced in gaming (Richardson et al., 2011), and in both studies investigated by Richardson et al. (2011) high gaming experience was related to higher task performance. Video gaming involves several spatial and cognitive abilities, and studies investigating causation show that gaming experience can improve mental rotation and visual attention (Moffat et al., 1998; Richardson et al., 2011). For instance, performance in visual search tasks, visual attention, visual memory, contrast sensitivity, and judging relative velocity have all been shown to improve with gaming experience (Dye et al., 2009; Moffat et al., 1998; Richardson et al., 2011). Performance in dynamic spatial tasks that required reasoning about moving stimuli (e.g. tracking objects) also improved (Richardson et al., 2011); And all those abilities are important for a high objective performance (Total hits C1 – C3) in this experiment. When specifically considering spatial abilities, there are known gender differences, including visuospatial abilities such as spatial orientation and spatial visualization (Moffat et al., 1998). Males outperform females in spatial performance tasks; In particular when it involves mental rotations, whether that task is paper-and-pencil (manipulations and transformations of geometric figures and forms) or in a virtual environment (Moffat et al., 1998; Richardson et al., 2011). Since males generally have more gaming experience than females and video game experience influence visuospatial processes, this might further contribute to gender differences in spatial tasks (Richardson et al., 2011), and moreover the objective performance (Total hits C1 – C3) in this experiment. In fact, females and males with similar levels of gaming experience did not differ in dynamic spatial ability, and gender differences were eliminated when gaming experience was included as a covariate (Richardson et al., 2011). Since the females in our experiment generally had less gaming experience, and those who did tended to cluster towards the male gamers, and since non-gaming males tended to cluster towards the females, we therefore identify an effect of gaming experience. We do recognize the high collinearity between gender and gaming experience, both had a high loading on PC2 (Figure 11); However, further analysis is needed to examine what exactly separates the data here. Still, PC2 consists mainly of objective measures, e.g. gender, gaming experience, and objective performance (Total hits C1 – C3). For PC1, we have high loadings on individual workload dimensions (which are subjective), in which all are correlated except for subjective performance, and so they contribute to the definition of PC1. In summary, PC1 consists mainly of subjective variables from surveys, whereas PC2 consists mainly of objective variables collected in the experiment.

6 Conclusion – An increase in human performance

This work investigated human operators' performance and their subjectively experienced workload in a teleoperation context when using a predictive display. Human operator performance decrease and workload increase as latency is introduced in teleoperation, but there exist several approaches to combat these detrimental effects; One of which is predictive technology. A predictive display based on image transformation was developed by applying positional and scale transformations to the video feed and tested experimentally. An experiment was set up to test the predictive display and investigate changes in human operator performance and workload when operating an ROV. N = 57 participants conducted a simple navigational task (peg-in-hole game), under three conditions: C1. Latency, C2. Latency with predictive display and C3. Baseline. ANOVAs showed a statistically significant increase of 20% in human performance with the aid of the predictive display. Differences in overall subjective workload was not statistically significant, except for with subjective performance where participants felt they performed better with the predictive display. Gaming experience was advantageous, in fact gamers increased their score with almost twice as much as non-gamers. An exploratory data analysis (EDA) investigated confounding factors with confirmatory results.

References

Allison, R. S., Harris, L. R., Jenkin, M., Jasiobedzka, U., & Zacher, J. E. (2001). Tolerance of temporal delay in virtual environments. *Proceedings IEEE Virtual Reality 2001*, 247–254. https://doi.org/10.1109/VR.2001.913793

Ando, N., Lee, J.-H., & Hashimoto, H. (1999). A study on influence of time delay in teleoperation—Quantitative evaluation on time perception and operability of human operator. *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, *5*, 1111–1116 vol.5. https://doi.org/10.1109/ICSMC.1999.815712

Appelqvist, P., Knuuttila, J., & Ahtiainen, J. (2007). Development of an Unmanned Ground Vehicle for task-oriented operation—Considerations on teleoperation and delay. *2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 1–6. https://doi.org/10.1109/AIM.2007.4412567

Baldwin, J., Basu, A., & Zhang, H. (1999). Panoramic video with predictive windows for telepresence applications. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, *3*, 1922–1927 vol.3. https://doi.org/10.1109/ROBOT.1999.770389

Bejczy, A. K., Kim, W. S., & Venema, S. C. (1990). The phantom robot: Predictive displays for teleoperation with time delay. *, IEEE International Conference on Robotics and Automation Proceedings*, 546–551 vol.1. https://doi.org/10.1109/ROBOT.1990.126037

Chen, J. Y. C., Haas, E. C., & Barnes, M. J. (2007). Human Performance Issues and User Interface Design for Teleoperated Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *37*(6), 1231–1245. https://doi.org/10.1109/TSMCC.2007.905819

Davis, J., Smyth, C., & McDowell, K. (2010). The Effects of Time Lag on Driving Performance and a Possible Mitigation. *IEEE Transactions on Robotics*, *26*(3), 590–593. https://doi.org/10.1109/TRO.2010.2046695

Dorais, G., Bonasso, R. P., Kortenkamp, D., Pell, B., & Schreckenghost, D. (1999). *Adjustable autonomy for human-centered autonomous systems*. 16–35.

Draper, J. V., Kaber, D. B., & Usher, J. M. (1998). Telepresence. *Human Factors*, *40*(3), 354–375. https://doi.org/10.1518/001872098779591386

Dye, M. W. G., Green, C. S., & Bavelier, D. (2009). Increasing Speed of Processing With Action Video Games. *Current Directions in Psychological Science*, *18*(6), 321–326. https://doi.org/10.1111/j.1467-8721.2009.01660.x

ENDSLEY, M. R. (1999). Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics*, *42*(3), 462–492. https://doi.org/10.1080/001401399185595

Fabrizio, M. D., Lee, B. R., Chan, D. Y., Stoianovici, D., Jarrett, T. W., Yang, C., & Kavoussi, L. R. (2000). Effect of Time Delay on Surgical Performance During Telesurgical Manipulation. *Journal of Endourology*, *14*(2), 133–138. https://doi.org/10.1089/end.2000.14.133

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, *47*(6), 381. https://doi.org/10.1037/h0055392

Goodrich, M. A., Olsen, D. R., Crandall, J. W., & Palmer, T. J. (2001). Experiments in adjustable autonomy. *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, 1624–1629.

Green, C. S., & Bavelier, D. (2003). Action video game modifies visual selective attention. *Nature*, *423*(6939), 534–537. https://doi.org/10.1038/nature01647

Hart, S. G. (2006). Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *50*(9), 904–908. https://doi.org/10.1177/154193120605000909

Hart, S. G., & Staveland, L. E. (1988a). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in Psychology*, *52*, 139–183.

Hart, S. G., & Staveland, L. E. (1988b). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in Psychology*, *52*, 139–183.

Hendy, K. C., Hamilton, K. M., & Landry, L. N. (1993). Measuring subjective workload: When is one scale better than many? *Human Factors*, *35*(4), 579–601.

Hill, S. G., Iavecchia, H. P., Byers, J. C., Bittner, A. C., Zaklade, A. L., & Christ, R. E. (1992). Comparison of Four Subjective Workload Rating Scales. *Human Factors*, *34*(4), 429–439. https://doi.org/10.1177/001872089203400405

Hu, H., Perez, C., Sun, H., & Jagersand, M. (2016). Performance of Predictive Display Teleoperation under Different Delays with Different Degree of Freedoms. *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, 380–384. https://doi.org/10.1109/ISAI.2016.0087

IBM SPSS Statistics 25. (2017). IBM Corp.

Kirk, R. (2013). *Experimental Design: Procedures for the Behavioral Sciences*. SAGE Publications, Inc. https://doi.org/10.4135/9781483384733

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (pp. 87–90). IOS Press. https://doi.org/10.3233/978-1-61499-649-1-87

Krishnaiah, P. R. (1980). *Analysis of variance* (Vol. 1). North-Holland.

Lane, J. C., Carignan, C. R., Sullivan, B. R., Akin, D. L., Hunt, T., & Cohen, R. (2002). Effects of time delay on telerobotic control of neutral buoyancy vehicles. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, *3*, 2874–2879 vol.3. https://doi.org/10.1109/ROBOT.2002.1013668

Lim, J., Wu, W., Wang, J., Detre, J. A., Dinges, D. F., & Rao, H. (2010). Imaging brain fatigue from sustained mental workload: An ASL perfusion study of the time-on-task effect. *NeuroImage*, *49*(4), 3426–3435. https://doi.org/10.1016/j.neuroimage.2009.11.020

Lovi, D., Birkbeck, N., Herdocia, A. H., Rachmielowski, A., Jägersand, M., & Cobzaş, D. (2010). Predictive display for mobile manipulators in unknown environments using online vision-based monocular modeling and localization. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5792–5798. https://doi.org/10.1109/IROS.2010.5649522

Lu, S., Zhang, M. Y., Ersal, T., & Yang, X. J. (2018). Effects of a Delay Compensation Aid on Teleoperation of Unmanned Ground Vehicles. *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 179–180. https://doi.org/10.1145/3173386.3177064

Luck, J. P., McDermott, P. L., Allender, L., & Russell, D. C. (2006). An Investigation of Real World Control of Robotic Assets Under Communication Latency. *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, 202–209. https://doi.org/10.1145/1121241.1121277

Lum, M. J. H., Rosen, J., King, H., Friedman, D. C. W., Lendvay, T. S., Wright, A. S., Sinanan, M. N., & Hannaford, B. (2009). Teleoperation in surgical robotics – network latency effects on surgical performance. *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 6860–6863. https://doi.org/10.1109/IEMBS.2009.5333120

Ma, R., & Kaber, D. B. (2006). Presence, workload and performance effects of synthetic environment design factors. *International Journal of Human-Computer Studies*, *64*(6), 541–552. https://doi.org/10.1016/j.ijhcs.2005.12.003

MacKenzie, I. S., & Ware, C. (1993). Lag As a Determinant of Human Performance in Interactive Systems. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, 488–493. https://doi.org/10.1145/169059.169431

Mathan, S., Hyndman, A., Fischer, K., Blatz, J., & Brams, D. (1996). Efficacy of a Predictive Display, Steering Device, and Vehicle Body Representation in the Operation of a Lunar Vehicle. *Conference Companion on Human Factors in Computing Systems*, 71–72. https://doi.org/10.1145/257089.257147

Matheson, A., Donmez, B., Rehmatullah, F., Jasiobedzki, P., Ng, H.-K., Panwar, V., & Li, M. (2013). The Effects of Predictive Displays on Performance in Driving Tasks with Multi-Second Latency: Aiding Tele-Operation of Lunar Rovers. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *57*(1), 21–25. https://doi.org/10.1177/1541931213571007

Maxwell, S. E., & Delaney, H. D. (2003). *Designing experiments and analyzing data: A model comparison perspective*. Routledge.

Miller, D. P., & Machulis, K. (2005). *Visual aids for lunar rover tele-operation*. proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space, edited by R. Battrick, ESA Publishing, Noordwijk, Netherlands.

Moffat, S. D., Hampson, E., & Hatzipantelis, M. (1998). Navigation in a "Virtual" Maze: Sex Differences and Correlation With Psychometric Measures of Spatial Ability in Humans. *Evolution and Human Behavior*, *19*(2), 73–87. https://doi.org/10.1016/S1090-5138(97)00104-9

Neumeier, S., Wintersberger, P., Frison, A.-K., Becher, A., Facchi, C., & Riener, A. (2019). Teleoperation: The Holy Grail to Solve Problems of Automated Driving? Sure, but Latency Matters. *Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 186–197. https://doi.org/10.1145/3342197.3344534

Nielsen, C. W., Goodrich, M. A., & Ricks, R. W. (2007). Ecological Interfaces for Improving Mobile Robot Teleoperation. *IEEE Transactions on Robotics*, *23*(5), 927–941. https://doi.org/10.1109/TRO.2007.907479

Oboe, R., & Fiorini, P. (1998). A Design and Control Environment for Internet-Based Telerobotics. *The International Journal of Robotics Research*, *17*(4), 433–449. https://doi.org/10.1177/027836499801700408

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2008). Situation Awareness, Mental Workload, and Trust in Automation: Viable, Empirically Supported Cognitive Engineering Constructs. *Journal of Cognitive Engineering and Decision Making*, *2*(2), 140–160. https://doi.org/10.1518/155534308X284417

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perez, M., Xu, S., Chauhan, S., Tanaka, A., Simpson, K., Abdul-Muhsin, H., & Smith, R. (2016). Impact of delay on telesurgical performance: Study on the robotic simulator dV-Trainer. *International Journal of Computer Assisted Radiology and Surgery*, *11*(4), 581–587. https://doi.org/10.1007/s11548-015-1306-y

Rachmielowski, A., Birkbeck, N., & Jägersand, M. (2010). Performance evaluation of monocular predictive display. *2010 IEEE International Conference on Robotics and Automation*, 5309–5314. https://doi.org/10.1109/ROBOT.2010.5509652

Richardson, A. E., Powers, M. E., & Bousquet, L. G. (2011). Video game experience predicts virtual, but not real navigation performance. *Computers in Human Behavior*, *27*(1), 552–560. https://doi.org/10.1016/j.chb.2010.10.003

Ricks, B., Nielsen, C. W., & Goodrich, M. A. (2004). Ecological displays for robot interaction: A new perspective. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, *3*, 2855–2860 vol.3. https://doi.org/10.1109/IROS.2004.1389842

Schmider, E., Ziegler, M., Danay, E., Beyer, L., & Bühner, M. (2010). Is It Really Robust? *Methodology*, *6*(4), 147–151. https://doi.org/10.1027/1614-2241/a000016

Schutte, P. C. (2017). How to make the most of your human: Design considerations for human–machine interactions. *Cognition, Technology & Work*, *19*(2), 233–249. https://doi.org/10.1007/s10111-017-0418-2

Sheridan, T. B. (1995). Teleoperation, telerobotics and telepresence: A progress report. *Control Engineering Practice*, *3*(2), 205–214. https://doi.org/10.1016/0967-0661(94)00078-U

Tukey, J. W. (1977). *Exploratory data analysis* (Vol. 2). Reading, Mass.

Vidulich, M. A., & Tsang, P. S. (1987). Absolute Magnitude Estimation and Relative Judgement Approaches to Subjective Workload Assessment. *Proceedings of the Human Factors Society Annual Meeting*, *31*(9), 1057–1061. https://doi.org/10.1177/154193128703100930

Wickens, T. D., & Keppel, G. (2004). *Design and analysis: A researcher's handbook*. Pearson Prentice-Hall.

Xu, S., Perez, M., Yang, K., Perrenot, C., Felblinger, J., & Hubert, J. (2014). Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dV-Trainer® simulator. *Surgical Endoscopy*, *28*(9), 2569–2576. https://doi.org/10.1007/s00464-014-3504-z

Zhang, Y., & Li, H. (2016). Handling qualities evaluation of predictive display model for rendezvous and docking in lunar orbit with large time delay. *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, 742–747. https://doi.org/10.1109/CGNCC.2016.7828878

Zheng, Y., Brudnak, M. J., Jayakumar, P., Stein, J. L., & Ersal, T. (2016). An Experimental Evaluation of a Model-Free Predictor Framework in Teleoperated Vehicles**This work was supported by the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-14-2-0001 U.S. Army Tank Automotive Research, Development and Engineering Center (TARDEC) Warren, MI. UNCLASSIFIED: Distribution Statement A. Approved for public release. #27479. *IFAC-PapersOnLine*, *49*(10), 157–164. https://doi.org/10.1016/j.ifacol.2016.07.513

Appendix A

Table A1. Descriptive Statistics Subjective variables.

Variable	C1. Latency		C2. Latency w/PD		C3. Baseline	
	Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**
Subjective Overall Workload	5.263	0.197	4.927	0.192	3.488	0.193
Mental Demand 0-10	5.667	0.273	5.509	0.301	3.561	0.271
Physical Demand 0-10	2.877	0.285	2.842	0.293	2.175	0.245
Temporal Demand 0-10	5.842	0.277	5.667	0.280	5.386	0.307
Subjective Performance 0-10[1)]	5.526	0.307	4.737	0.274	2.702	0.214
Effort 0-10	6.018	0.260	5.772	0.266	4.667	0.278
Frustration 0-10	5.625	0.319	4.946	0.275	2.446	0.243

Table A2. One-way repeated measured ANOVA F-test.

Variable	N	Outliers[ae]	Normality	Sphericity	Epsilon (ε)	F-statistic	Sig.	Sample effect size	Population effect size[c]
Subjective Overall Workload	57	Yes (1)	Yes	Yes $\chi^2(2) = 3.787$, p = 0.151	-	*F*(2, 112) = 68.322	p< 0.001**	$\eta^2 = 0.55$	$\omega^2 = 0.441$
(outlier removed)	56	No	Yes	Yes $\chi^2(2) = 3.999$, p = 0.135	-	*F*(2, 110) = 68.311	p< 0.001**	$\eta^2 = 0.55$	$\omega^2 = 0.445$
Mental demand	57	No	Yes[d]	No $\chi^2(2) =$ 9.962198, p = 0.007	ε = 0.858[b]	F(1.716, 96.082) = 41.286	p<0.001**	0.424	$\omega^2 = 0.32$
Physical demand	57	Yes (1) [f]	Yes[d]	Yes	-	F(2, 112) = 6.474	p = 0.002**	$\eta^2 =$ 0.104	$\omega^2 = 0.060$

(outlier removed)				χ2(2) = 0.357, p = 0.837.					
	56	No	Yes[d]	Yes χ2(2) = 1.198 , p = 0.549.	-	F(2, 110) = 5.601	p = 0.005**	η^2 = 0.092	ω^2 = 0.052
Temporal demand	57	Yes (2)[f]	Yes[d]	No χ2(2) = 6.498, p = 0.039.	ε = 0.900[b]	F(1.799, 100.771) = 1.690	p = 0.192	η^2 = 0.029	ω^2 = 0.008
(outlier removed)	55	No	Yes[d]	No χ2(2) = 6.504, p = 0.039.	ε = 0.896	F(1.793, 96.819) = 1.686	p = 0.193	η^2 = 0.030	ω^2 = 0.008
Subjective performance	57	Yes (2)[f]	Yes[d]	Yes χ2(2) = 1.552, p = 0.460.	-	F(2, 112) = 78.578	p < 0.001**	η^2 = 0.584	ω^2 = 0.476
(outlier removed)	55	Yes (3)[g]	Yes[d]	Yes χ2(2) = 1.972, p = 0.373.	-	F(2, 108) = 81.030	p < 0.001**	η^2 = 0.600	ω^2 = 0.492
Effort	57	Yes (1)[f]	Yes[d]	Yes χ2(2) = 1.143, p = 0.565	-	*F*(2, 112) = 19.641	p < 0.001**	η^2 = 0.260	ω^2 = 0.179
(outlier removed)	56	No	Yes[d]	Yes χ2(2) = 1.277, p = 0.528	-	*F*(2, 110) = 18.627	p < 0.001**	η^2 = 0.253	ω^2 = 0.173
Frustration	56	No[h]	Yes[d]	Yes χ2(2) = 0.519, p = 0.771	-	F(2,112) = 63.275	p < 0.001**	η^2 = 0.535	ω^2 = 0.426

*: p < 0.05, **: p < 0.01

a) Number of outliers in parentheses.

b) A Greenhause Geisser correction was applied.

c) Calculated according to (Wickens & Keppel, 2004).

d) Visual inspection of Normal Q-Q Plots and histograms for all three conditions.

e) Visual inspection of a boxplot.

f) Outliers was kept in the first ANOVA as there was no reason for excluding them and a Friedman test with pairwise comparisons using a Bonferroni correction was carried out for comparison purposes, as this test is less affected by outliers. Results were corroborated. We also reran the analysis with outliers excluded, which resulted in somewhat higher effect size. The overall result was the same.

g) Excluding initial outliers did not yield a dataset without outliers. Further outlier removal was not conducted to avoid constructing a highly reduced, and thus unrepresentative dataset.

h) There were three outliers in the sample with N=57. One outlier in C2. Latency w/PD, reported the highest frustration while feeling like their performed the worst with PD. We assume this was due to not understanding what PD was trying to do and thus we removed this participant from the analysis of Frustration. The two outliers in C3. Baseline reported high frustration in all three conditions and were therefore kept. When excluding the abovementioned participant and rerunning the analysis, there were not outliers in the data. N=56 data points were used for this specific analysis.

ISBN 9788232651245 (printed ver.)
ISBN 9788232651252 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)