Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 60 (2017) 554 – 559

www.elsevier.com/locate/procedia

27th CIRP Design 2017

Prototype experiments: strategies and trade-offs

Sigmund A. Tronvoll*, Christer W. Elverum, Torgeir Welo

NTNU - Norwegian University of Technology and Science - Department of Mechanical and Industrial Engineering, Richard Birkelands vei 2B, 7491 Trondheim, Norway.

* Corresponding author. Tel.: +47-905-20-107. *E-mail address:* sigmund.tronvoll@ntnu.no

Abstract

Throughout the whole product development process, there is always a question on whether the proposed product is "up to its task", and often it is up to the engineer or designer to answer these questions. In many cases, this calls for experiments in form of prototype testing, to explore, verify, and validate the product performance. This paper connects the overall approach of the development process, in form of point-based, set-based, and agile strategies, and connects them to what seems to be the fundamental tradeoff in prototype experiments, exemplified by real cases from an industrial-academic development project.

Peer-review under responsibility of the scientific committee of the 27th CIRP Design Conference

Keywords: agile development; set-based engineering; point-based engineering; prototyping; experimentation

1. Background and introduction

As product development methodologies come and go, some aspects of product development remain invariant. Redesign, rework or iterative work cycles in its simplest form exists in all engineering activities [1], as changing measures of a machine drawing due to interference of parts, choosing different color of a part after seeing a rendered CAD model, or larger failures as rebuilding a late-stage prototype due to a weak design.

Going back to the basics, engineering design is about fulfilling a need. Some of these can be parametrized into equations, and mathematically solved, resulting in a solution that will work. An example could be deciding the needed cross section for a cantilever beam to hold a certain load without overshooting its yield strength. However, the transition from needs to design for something as simple as a bar stool proves difficult when considering all possible inflicting parameters as fatigue life, material defects and user preferences and behavior.

When digging into the problem, designers and engineers therefore introduce simplifications and assumptions to confine the problem into a neater package of solvable bits and pieces of problems and sub-problems. Based on these models, a qualified (or unqualified) guess for a suitable solution is composed and evaluated against its requirements through some form of experiment. As these are in fact guesses, the resulting design will sometimes fail, either due to lack of understanding of what the product should do or withstand, or due to overly crude simplifications and assumptions. The solution must therefore subsequently be redesigned and tested again. The cost of such rework often depends on the level of commitment introduced after the initial work is done, and tend to be more expensive the further you get into the process [2]. Or more accurately; the sunk cost of the initial faulty work that must be discarded could have been substantial, while rework is the means of correcting those faults.

These, almost unavoidable, cycles are the background for the term iterative design cycles, which are celebrated and formalized in some cultures (as *design thinking* [3] and *agile development* [4]) and doomed in others (as *quality function deployment* [5] and *total quality management* [6]). The difference is not whether they exist in different cultures, but how they are perceived. Does the culture emphasize tuning each design decision to perfection before committing to it, or do they acknowledge that early design decisions will be flawed, and therefore iterates with larger changes between each cycle? Do the culture lean towards "do it right the first time" or "just do it"?

Peer-review under responsibility of the scientific committee of the 27th CIRP Design Conference
doi:10.1016/j.procir.2017.01.049

To increase confidence in solutions, engineers often utilize different dimensioning tools as *allowable stress design* and *limit state design*. In addition to this, engineers and designers try to reduce the cost of rework through design strategy and improved analysis/tests of design proposals. Elverum, Welo and Tronvoll [7] have earlier proposed guidelines for choosing prototyping methods for design and evaluation, based on contextual factors. To be able to expand on these contextual guidelines, this paper investigates and clarifies the trade-offs associated with choosing approach for experimenting with prototypes. Two examples from an academic/industrial collaborative project are displayed, to show how the iterative nature of product development, together with the strategy of the product development task, affects the experiment trade-off.

2. The four strategies

Before going into how prototype experiments are conducted, it is important to set their purpose in a strategic context. In what overall development strategy are the experiments utilized? There are in in principle four different approaches to progress a design task. The choice depends on the extent to which a proposed solution is refined/fixated, and what backup plans are built into the product or the process:

1. **Point-based design** - Make sure that the chosen solution would be suitable/work, and stick with it. Use as much resources as needed to be able to proceed with a design, only when you have achieved a very high confidence in its performance. If the design appears to be missing its target, redo the process (no backup plan), similar to trying to implement the stage/phase-gate approach as a development process rather than a management tool [8].
2. **Set-based solution array -** Create an array (set) of solutions/designs that potentially could perform at certain level and hope that at least one would be suitable. Screening/convergence is based on continuous or stage-wise estimation of performance and gradual elimination of weak alternatives (as opposed to searching for the best alternatives) [9].
3. **Performance set investigation -** Instead of choosing multiple solutions, choose the most promising one, and investigate its performance thoroughly so that it's capability could be represented as a range of performance rather than the traditional compliance with requirements. Postpone committing to design decisions, which will constrain the design, until they can be validated. This will eventually lead to a gradual convergence of the design [9,10]. The capabilities and performance ranges are also named sets in some literature.
4. **Flexible design** - Design a best-guess solution so that eventual necessary design changes are easily implementable. Fixate, but allow for change (for example through extensive use of modularization). Validate when possible and change direction if needed [4]. This is the only strategy actually designed for iterating, while the others try to mitigate uncertainty in other ways.

The twofold use of 'set' is only to point out that they are essential to set-based concurrent engineering, in which they are combined [11]. As strategy 2 and 3 stems from studies of Toyota, they are naturally described as components of Lean Product Development [9,12]. There is currently a lack of naming convention on the two aspects of set-based strategies, so their naming in this paper does only reflect their function.

The term flexible could as well be replaced by agile, as used in the software industry. However, the second and third principle are also useful tools to achieve an agile development process [13] for physical goods.

Although the strategies have been graphically presented in various ways in earlier work (as set-based in Smith [13] and agile-like in Steinert and Leifer [14]), the essential difference between the strategies boils down to how the concepts evolve in a design space/time diagram, as seen in figure 1.

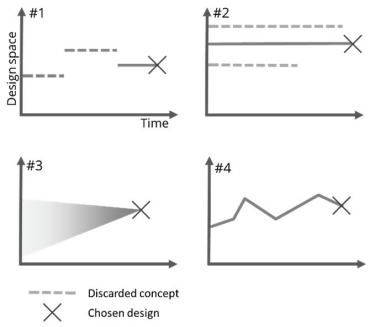

Figure 1: Design progress of the four different strategies. 1) Point based design, 2) Set based solution array, 3) Performance set investigation, 4) Flexible design.

Design teams rarely stick with one of the strategies exclusively, as this would be mentally difficult and impractical. Furthermore, the differences between the strategies are fuzzy when it comes to real-world application as they are mostly combined. What is certain is that designers and engineers tend to intentionally or unintentionally change between the different strategies. It would be rational to differentiate between systems design and component design, where systems are often designed and evaluated using strategy 3 and 4, while the components are designed using strategies 1-3. This is due to the number of variables/components in larger systems, which introduces a lot of uncertainty in the start and would need the flexibility of performance set investigation and flexible design to handle this uncertainty. On the other hand, keeping a sufficient set-based solution array for large systems is cumbersome, and point based design would likely fail due to the uncertainty. Contrary, functions of sub-components are easier to identify, easier to create solution sets of, cheaper to replace, and less cumbersome to fully redesign if they fail. Other researchers have also identified that as the development process is getting closer to finish, the process often tends to approach point-based [13].

3. Iteration through prototype experiments

Iterative development cycles have been a widespread concept for a long time (see Figure 2), probably first formally introduced by Simon [15] as the *generate-test cycle* for explaining how artificial intelligence might contribute to engineering and design. This has later been expanded to the *design-build-test cycle* from [16] to fit for the problem solving cycles in the Japanese automotive industry, or the *design-build-run-analyze cycle* from Thomke [17] stressing the analyze part as an essential activity to learn from the cycle.

The basic of iterative development is however the fact that design tasks consist of cycles with divergent and convergent activities, where one propose, create and test solutions, and subsequently re-create, re-design and re-test if the initial results are not satisfying.

Experimentation, borrowing Smith's description [13] can be described as "you provoke a situation and see how it responds". The reason for explicitly using the term *prototype experiments* rather than prototypes as an isolated artefact, is that a prototype is made for evaluation through some form of experiment. A prototype experiment often targets generating knowledge about different attributes of a proposed design which is not identified by simple reflection. This could be screening of solutions, milestone tests, fulfilment of requirements, proof of concept/manufacturing, integration etc.

In contrast to many researchers which are using the term prototype as a tangible artefact, as opposed to virtual prototypes, we choose in this paper to keep the term as open as possible. Ulrich and Eppinger's definition of prototypes as "… an approximation of the product along one or more dimensions." [18] would to a large extent cover most product development experiments.

Clark et al. [16] identified that different prototype experiments were used in the Japanese automotive industry, during so called "problem solving cycles", in classical set-based engineering; "Inside the problem solving cycles, alternative solutions are created or retrieved, and their consequences are simulated through physical, mental, or computer experiments". It must be noted that the term *computer experiment* should be rephrased to *analytic calculations,* as this is what they are, only as a system of many calculations done at the same time. In this paper we disregard investigating mental experiments, although probably the least cost intensive, as this is a matter of brain power and not strategic experimentation.

The choice of arms would depend on the problem, the wanted form of the output and the capability of the development team, but the main goal must be finding the consequence of a solution through the most favorable type of experiment.

An important fact about prototype experiments is that in addition to including an incomplete model of the product (the prototype), the test also most often includes an incomplete model of its environment [19], and unanticipated behavior often tend to happen when either of the models are replaced by a more comprehensive one [17,18]. The test environment, in contrast to the prototype, can in theory span over an infinite range of scenarios, as it is difficult for a product developer to know in advance how the product will be used. In addition to the variety of the scenarios and load cases, the uncertainty of their duration and occurrence throughout their life span makes all prototype testing incomplete. This is especially significant for consumer durables, as they in contrast to capital goods, are rarely designed for each individual buyer/user and also suffer from less contact with producer/designer, both before, under and after purchase.

Generalized, the test environment attributes are possible to fit within one or more of the categories of human interaction, physical environment, and product structure. Often, many experiments must be performed to target the span of anticipated scenarios and types of environment the product is supposed to be subjected to. Drawing the analogy to the iterative design-build-test cycle, this would be extended with a parallel loop with iterations around the test environment [20] (see figure 2), to illustrate how the environment is iteratively changed to test a prototype for different cases.

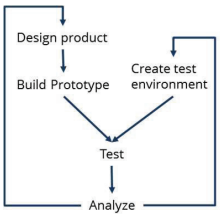

Figure 2: The iterative prototype experiment cycle

Some test setups are created for being able to test multiple prototypes, which allows for experimenting with different solutions which has been shown to be useful for set-based techniques [10].

4. Dimensions of experimental tests in product development

Assessing the important attributes of an experimental test, we could say that the most important result is the increase in the industrial performance parameters, or the general (not project specific) usefulness of the experiment. An experiment should therefore aim for either of these improvements:

- Reduce development cost
- Shorten development time
- Increase total product quality
- Create reusable physical assets
- Create reusable knowledge

Point 1-3 are the general product development performance parameters introduced by Clark and Fujimoto [21], which target the single project performance, while parameter 4 and 5 target the project-to-project asset transfer emphasized in *knowledge based engineering* [22], and *front loading* tactics [23]. All these factors are also emphasized in Lean Product Development [12], focusing on project performance, knowledge, and asset generation, especially for the use in continuous improvement (Kaizen).

However, these are the measures of an outcome, and not the dimensions of the experiment itself, and are difficult to assess on forehand. The dimensions of an experiment should capture how the experiment recreates its real physical counterpart. To what cost, in what way and to what extent does an experiment, present the behavior and performance of the real product in a real environment. The factors of influencing this are:

1. Iteration cost (What is the cost of the experiment)
2. Iteration time (What is the time used on an experiment)
3. Approximation level (How correct is the result)
4. User level (How easy is it to use)
5. Result presentation implicit/explicit (How easy it is to draw conclusions from the data)
6. Experiment flexibility (How easy it is to change conditions)

Dimension 1-3 are quite directly linked to general process performance, while 4-5 are parameters less focused on through research. However, dimension 4-6 might hold the key to why designers, engineers, and managers choose the approach they do.

4.1. Iteration cost

As one of the attributes directly linked to general product development performance, this is a natural component of the experimentation trade-off. However, it must be noted that the overall process performance and the experiment iteration cost is not necessarily directly correlating. If taking into account that approximately 80% of the life cycle costs of a product are committed to after spending 20% of the product development cost [2], there are reasons to believe that somewhat costly experiments in the early phase, that could be giving high learning value could often be beneficial.

Some forms of experimentation, as simulations, can often be iterated again and again, with negligible marginal cost, while destructive testing of physical prototypes often have the same marginal as initial cost.

4.2. Iteration time

As with iteration cost, this is a quite obvious part of any performance measure. And as identified by Thomke et. al.[23], using time and resources on extensive testing and knowledge creating in the start of a project (termed *front-loading*), could result in time savings for the overall project performance.

Reproducing all loads and durations acting on a consumer durable throughout it's safe-life is rarely done in product development, as their safe--life do often extend the product development phase by orders of magnitude. One does therefore often create a more compact load scheme (or scenario), or test multiple similar prototypes on different cases in parallel. Especially in software industry, where users are often granted permission to use not-yet-launched products (as a part of the later stages of the development; *beta testing),* and in return get user statistics, error messages, and bugs which the software exhibits. As an example, Windows 7 had 8 million individual, non-paid, testers [24]. That did not only free up time for the development team, but made a much more extensive test than any in-house testing facilities could have done.

4.3. Level of approximation

This is one the main counterarguments against computer simulations. Engineers, and maybe especially managers will often prefer tangible experiments for tangible artefacts as exemplified by Iansiti and MacCormack [25]. The world is transient, non-linear, non-conservative, multi-physical, stochastic, and continuous in extent. Reducing this to a, very often, single physics, steady state simulation, is an extreme simplification, and often requires very high knowledge about the situation and limitations of the simulation procedure. If these qualifications are not present, this would often lead to crude results (crap-in, crap out).

Lean management relies heavily on observing the real problems in order to understand and solve them. This has been named *San-Gen Shugi* (*the three reals*): *Gemba*, *Gembutsu* and *Genjitsu*, which translates to *the real place*, *real item* and real *situation/data* [26]. In case of production, this is possible, as these are physical entities, while for product development it can be a bit more difficult due to the fact that all the "real physicals" only exist in the future, after product deployment.

Some have given "dimensions" to the approximation on whether the prototype is either rich/low on functions/attributes and how accurate the representation of these functions are. This has been named by some as fidelity vs. resolution [27] or horizontal vs. vertical prototyping [28,29]. Ulrich and Eppinger [18] has a different way of describing prototypes, as whether they are focused/comprehensive (the amount of functions implemented), and analytical/physical, giving a dimension on whether they are to be tested in the real world, or by analytic calculations/estimations.

4.4. User level

What knowledge and capabilities is needed to conduct the experiment is important when choosing mode of experimentation. This would often be one of the main indicators for whether the experiment can be conducted in-house or must be outsourced to other companies. This is where physical experiments have one of its great advantages. As physical prototyping methods and physical testing most often consists of familiar and tangible processes, it will be a higher possibility that the whole product development team understands the construction of the prototype, the design of the experiment and the implications of the results.

Choosing a toolset common for the whole design team (or the team is able to learn within reasonable time) for experimentation, could be beneficial as it not only allows the team to ask questions about the product performance, but also enables them to answer them.

4.5. Results presentation

Although performance measures, qualitative analysis and visual appearance can be described in words and numbers, a thorough analysis and understanding of the results often require

a more graphical appearance, in form of graphs, plots, videos and statements. Often, to get a feel for how the prototype performs, the result is then frequently showcased against some baseline performance (market leaders, nominal values, legislative values etc.), which often calls for experimenting with in market products also.

4.6. Experiment flexibility

This parameter targets the possibility of changing the experiment, either in terms of the product or the test environment. Fully functional prototypes facilitate testing a large part of their life-cycle scenarios, while a focused prototype might only fit for a very specific case.

5. Examples from an academic-industrial collaborative project

As a part of improving and developing products for property level flood protection, two parallel development tasks were performed, where one was mainly targeting improving water tightness of temporary flood barriers which could be implemented into the already existing product and new products, and the other case targeted developing all-new type of flood protection barriers.

6. Improving water tightness

This was a project initiated to improve the water tightness of self-stabilizing bookstand flood barriers, as shown in Bjerkholt and Lindholm [30]. The first round of experiments was done as a screening of sealing solutions against each other, in a set-based solution array driven development phase.

Figure 3 - First experiment setup of investigating sealing performance

The first experimental setup was designed to be able to replicate a general flood barrier, and the team used a plywood tank and half scale flood barrier. The ground was replicated using pebbles glued on with bed-liner, and the replica of the modules was constructed to accommodate a wide range of sealing solutions. There was no measuring of the leakage, but rather a visual estimation of much/less/no leakage, which was found to be sufficient for the experiment.

The team found two distinct sealing solutions which they favored, but in addition, gained a lot of knowledge about the coupling between leakage and structural stability.

The second experiment was based on the results from the first one, using one of the sealing solutions (a foam gasket), and explored the influence of water pressure vs. gasket pressure, in addition to being able to test different widths of the gasket. The new setup isolated the key sealing mechanism, to be able to draw quantifiable data, and neglected the overall failure effects observed when testing with a more realistic setup. The setup consisted of a bucket with a texture plate encircling a draining hole in the bottom, and on top of this a plate with a glued on circular gasket as seen in figure 8. The setup also incorporated automatic water leveling and input water measuring to be able to log performance data.

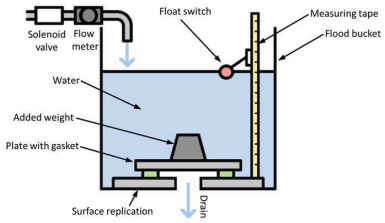

Figure 4: Second experiment setup [31], giving a more rigorous testing of performance parameters of a more constrained type of sealing solution.

The latter experiment reduced the flexibility of the setup, and would only accommodate seals of “gasket like” functionality. But in return allowed for exploring the effect of gasket pressure and water pressure on the leakage rate, [31]. This makes the latter experiment more of an investigation of the performance set of the chosen solution, rather than the set-based solution array nature of the first experiment.

So in terms of tradeoffs committed, somewhat higher iteration time and iteration cost of the setup (due to the data logging system), heightened user level, reduced flexibility, in trade of more explicit data. The approximation in the setup is decreased in terms of data extraction, but it neglects the potential instability issues found using the first setup, making it less accurate in terms of overall accuracy.

6.1. All new concept

This was a project initialized to develop a flood barrier for the consumer market (rather than business to business which is the company’s main market segment). The proposed product consisted of tripods holding a canvas (as seen in figure 5).

Figure 5: First prototypes, and tests, of canvas-tripod solution.

The first round of experiments was a *classical proof-of-concept prototype*. As there were uncertainties about the structural integrity of the concept, the overall purpose was to investigate whether it would work or not. The product was built in half scale, and tested in a steel tank. Due to the number of uncertainties, as a potential need of a rod for holding the canvas between the modules, length of canvas, tripod feet spread, flexibility of tripod, flexibility in all these factors were designed into the product. This was achieved using a high degree of modularity, allowed for features to be altered,

removed and introduced in multiple iterations (although keeping most of the concept fixed) throughout the experiment.

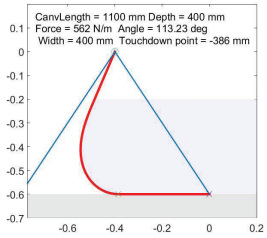

Figure 6: Second round of experimentation – Computer simulation of geometry and loading of canvas.

After getting some peculiar failure modes in the physical testing, the team wanted to know more about the loads and geometry of canvas subjected to a hydraulic pressure. Especially what canvas lengths would be safe, or alternatively introduce unstable behavior.

Although a simplified 2D form of the problem is governed by a very simple equation (cylindrical hoop stress equation) it results in a non-linear partial differential equation. Due to the apparent complexity in the problem, this could not be solved easily in normal structural simulation software, and was therefore hard coded in MATLAB. The results gave a clear overview of the limit between unstable and safe behavior of the barrier. As this experiment was limited to a single solution, for exploring its capabilities thoroughly, this leans toward a *performance-set investigation* type of experiment.

The tradeoff committed was increased user level, decreased flexibility, increased approximation, in trade of a more extensive results presentation, including canvas tension in addition to instability, less iteration cost and time.

7. Concluding remarks

Throughout a development project, designers and engineers shift their strategy to be able to answer the questions at hand. A lot of potential might lie in choosing the most appropriate method of experimentation within that strategy, which in turn calls for investigating their advantages, pitfalls, limitations and implications. The trade-off given when confining a real world problem into a prototype experiment, is characterized by six attributes; iteration time, iteration cost, approximation level user level, result presentation and experiment flexibility. These factors do not say much about the success of the overall product development, but they do give some key points for how the experiment can transform design parameters into valid data to support decision-making.

8. Aknowledgements

This research is funded by The Research Council of Norway, and done in collaboration with AquaFence AS.

9. References

[1] Smith RP, Tjandra P. Experimental observation of iteration in engineering design. Research in Engineering Design 1998;10:107–17.
[2] Haskins C, Forsberg K, Krueger M, Walden D, Hamelin D. Systems engineering handbook. INCOSE, 2006.
[3] Dow SP, Klemmer SR. The Efficacy of Prototyping Under Time Constraints. In: Meinel C, Leifer L, Plattner H, editors. Design Thinking, Springer Berlin Heidelberg; 2011, p. 111–28.
[4] Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, et al. Manifesto for agile software development 2001.
[5] Mohamed Zairi, Mohamed A. Youssef. Quality function deployment: a main pillar for successful total quality management and product development. International Journal of Quality & Reliability Management 1995;12:9–23.
[6] Harari O. Ten reasons why TQM doesn't work. Management Review 1993;82:33.
[7] Elverum CW, Welo T, Tronvoll S. Prototyping in New Product Development: Strategy Considerations. Procedia CIRP 2016;50:117–22.
[8] Cooper RG, Kleinschmidt EJ. New product processes at leading industrial firms. Industrial Marketing Management 1991;20:137–47.
[9] Ward A, Liker JK, Cristiano JJ, Sobek DK. The second Toyota paradox: How delaying decisions can make better cars faster. Sloan Management Review 1995;36:43.
[10] Kennedy BM, Sobek DK, Kennedy MN. Reducing Rework by Applying Set-Based Practices Early in the Systems Engineering Process. Systems Engineering 2014;17:278–96.
[11] Sobek DK, Ward AC, Liker JK. Toyota's Principles of Set-Based Concurrent Engineering. Sloan Management Review 1999;40:67–83.
[13] Welo T. On the application of lean principles in Product Development: a commentary on models and practices. International Journal of Product Development 2011;13:316–43.
[13] Smith PG. Flexible product development: building agility for changing markets. John Wiley & Sons; 2007.
[14] Steinert M, Leifer LJ. "Finding One"s Way': Re-Discovering a Hunter-Gatherer Model based on Wayfaring. International Journal of Engineering Education 2012;28:251.
[15] Simon HA. The sciences of the artificial. Cambridge, Mass.: MIT Press; 1969.
[16] Clark KB, Fujimoto T. Lead time in automobile product development explaining the Japanese advantage. Journal of Engineering and Technology Management 1989;6:25–58.
[17] Thomke SH. Managing experimentation in the design of new products. Management Science 1998;44:743–762.
[18] Ulrich KT, Eppinger SD. Product design and development. 5th ed. McGraw-Hill; 2012.
[20] Thomke SH. Simulation, learning and R&D performance: Evidence from automotive development. Research Policy 1998;27:55–74.
[20] Tronvoll SA, Elverum CW, Welo T. Test Environments in Engineering Design: A conceptual study. DS 85-1: Proceedings of NordDesign 2016, Volume 1, 2016.
[21] Clark KB, Fujimoto T. Product Development Performance: Strategy, Organization, and Management in the World Auto Industry. Harvard Business Press; 1991.
[23] Studer R, Benjamins VR, Fensel D. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 1998;25:161–97.
[24] Thomke S, Fujimoto T. The Effect of "Front-Loading" Problem-Solving on Product Development Performance. Journal of Product Innovation Management 2000;17:128–42.
[24] Protalinski E. Windows 7 had 8 million testers, biggest beta ever. Ars Technica 2009. www.arstechnica.com (accessed November 8, 2016).
[25] Iansiti M, MacCormack AD. Team New Zealand (A). Harvard Business Publising, 1996.
[26] Hill AV. The Encyclopedia of Operations Management: A Field Manual and Glossary of Operations Management Terms and Concepts. FT Press; 2012.
[27] McCurdy M, Connors C, Pyrzak G, Kanefsky B, Vera A. Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. Proceedings of the SIGCHI Conference, ACM; 2006, p. 1233–1242.
[28] Beaudouin-Lafon M, Mackay W. Prototyping tools and techniques. Human Computer Interaction-Development Process 2003:122–142.
[29] Floyd C. A Systematic Look at Prototyping. In: Budde R, Kuhlenkamp K, Mathiassen L, Züllighoven H, editors. Approaches to Prototyping, Springer Berlin Heidelberg; 1984, p. 1–18.
[30] Bjerkholt JT, Lindholm OG. An Innovative Semi-permanent Flood Protection Structure-Alternative to Sandbags and Supplements to Conventional Earth Embankments. Advances in Urban Flood Management 2007:461.
[31] Haaland KV, Walderhaug ØB. Master Thesis. Prototyping and testing of novel flood protection systems. 2016.