NTNU
Norwegian University of
Science and Technology

City Guide: Service Composition in an Urban Environment

Eirik Blakstad

Master of Science in Computer Science
Submission date: June 2009
Supervisor: John Krogstie, IDI
Co-supervisor: Jacqueline Floch, SINTEF

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

As a starting point in the UbiCompForAll project, a set of service scenarios in an urban environment have been identified. One of them, the city guide scenario, relates to flexible city guiding. The scenario assumes that a guide can easily create city tour services customized to the particular needs of tourist groups. Different functionalities such as information about points of interest, navigation support and communication facilities might be provided. The tourists use handheld devices while visiting the city to access these customized services. This assessment will investigate the issues of service composition and customization. As a case study, it will refine the city guide scenario and propose a set of alternative scenarios or stories. It will identify and specify relevant service building blocks that can be reused, composed and customized in these alternative scenarios. Selected building blocks will be realized to demonstrate composition in the scenario. Technology available in Wireless Trondheim should be used as far as possible (e.g. for positioning). City guide services developed in earlier assessments should also be exploited.

Assignment given: 15. January 2009
Supervisor: John Krogstie, IDI

Abstract

The aim of this thesis has been to develop and validate the user interface of a service composition tool for creating mobile tourist services in a city environment. The project has been conducted in cooperation with SINTEF and the Norwegian research project UbiCompForAll.

As part of this thesis, a number of scenarios were created initially, showing what kind of functionalities accessed using mobile devices might be useful for tourists in the city. Based on these scenarios, the graphical user interface of a composition service tool was created and then assessed through the involvement of several users using a paper prototyping approach.

The results from the paper prototyping assessment showed that the proposed interface is generally easy to understand and use, although there were a number of improvements necessary in the areas of information abstraction, module naming and letting users be able to see the final product of the composition.

In addition, as part of the assessment of the proposed interface, the realizability of the services specified using this interface was evaluated. An architecture for the service derived from the specification developed by the user was defined. This enabled us to check that all the information necessary for creating the composed services can be provided when using the proposed user interface.

Preface

This report documents the project work performed by Eirik Blakstad as the Master's Thesis for the Masters degree in Technology at The Norwegian University of Technology and Science (NTNU).

The subject matter of the thesis was defined through the UbiCompForAll research project, coordinated by SINTEF in Trondheim. UbiCompForAll started in October 2008, and the name is short for "Ubiquitous service Composition For All users".

The project was defined by Jacqueline Floch, coordinator of the UbiCompForAll project.

I would like to thank my main advisor at SINTEF, Jacqueline Floch, as well as Erlend Stav at SINTEF and my supervisor at NTNU, John Krogstie, for their support and feedback.

Trondheim, June 18th 2009

Eirik Blakstad

Contents

Abstract I

Preface III

Figures IX

Chapter 1: Introduction 1

1.1 Project background 1

1.2 Research questions 1

1.3 Goal and problem definition 1

1.4 Report outline 2

Chapter 2: Background 3

2.1 Ubiquitous Computing 3

2.2 Services and service composition 3

2.2.1 End-user programming 4

2.3 Wireless Trondheim 6

2.4 Existing mobile city guide solutions 7

2.4.1 myMytileneCity 7

2.4.2 The GUIDE project 8

2.4.3 What makes the work of this thesis unique? 9

Chapter 3: Research agenda and evaluation approach 11

3.1 Design Science 11

3.2 Evaluation approach 13

3.2.1 Scenarios 13

3.2.2 Graphical user interface 13

3.2.3 Implementation and architecture 13

Chapter 4: Requirement specification and scenarios 15

4.1 Usage scenarios 15

4.1.1 Scenario 1: Corporate group 15

4.1.2 Scenario 2: Arriving by car 17

4.1.3 Identified building blocks 18

4.2 Composition scenarios 18

4.3 Feedback from tourism experts 19
4.4 Requirements specification 20
4.4.1 Composed service requirements 20
4.4.2 Functional requirements for composition tool 20
4.4.3 Non-functional requirements for composition tool 20
Chapter 5: Service composition tool 21
5.1 General theory 21
5.2 Basic components 21
5.3 User interface 22
5.3.1 Initial interface 22
5.3.2 Revised interface 23
5.3.3 Interface in practice 25
Chapter 6: Assessment through paper prototyping 29
6.1 The test 29
6.1.1 Test subjects 30
6.1.2 The process 30
6.2 Results from tests and uncovered issues 31
6.2.1 First session 31
6.2.2 Second session 32
6.2.3 Third session 34
6.2.4 Test conclusions 34
Chapter 7: Assessment through realizability study 37
7.1 Service realization 37
7.1.1 Service meta-model 38
7.2 Service logic 39
7.3 Service framework architecture 40
7.4 Validation 41
Chapter 8: Conclusion and further work 45
8.1 Achievements 45
8.1.1 Results 45
8.1.2 Research questions 45

8.1.3 Fulfillment of the requirements 45
8.2 Discussion 46
8.3 Further work 46
Appendix A: References 49

Figures

Figure 1: Example of visual programming interface for creating web applications [Tersus 2009] 6

Figure 2: The myMytileneCity setup 7

Figure 3: The GUIDE system's interface and guiding functionality 9

Figure 4: Initial user interface draft 23

Figure 5: Create menu 23

Figure 6: The main types of elements 24

Figure 7: Other user interface elements 25

Figure 8: Creating a new guided tour 26

Figure 9: User clicks "Add people" 27

Figure 10: User clicks "Add route" 28

Figure 11: Conducting the paper prototype test 30

Figure 12: New property page when adding person to a group 32

Figure 13: The "Places" element before and after changes 33

Figure 14: Final version of the Event element 34

Figure 15: Realizability components 37

Figure 16: Meta-model showing classes making up the service 38

Figure 17: Architecture with reusable service components 40

Figure 18: Sequence diagram for operation 1, logging on 42

Figure 19: Sequence diagram for operation 2, seeing your friends on a map 43

Figure 20: Sequence diagram for operation 3, going to the next place in a route 44

Chapter 1: Introduction

This chapter briefly introduces the background and context of this thesis, it describes the motivation behind it, thoroughly defines the problem and gives an overview of this report.

1.1 Project background

The UbiCompForAll research project is financed by the Norwegian Research Council. It is coordinated by SINTEF and involves several academic and industry partners, including NTNU and Wireless Trondheim AS. The aim of UbiCompForAll is to provide support for ordinary end users to help them compose services adapted to their needs in intelligent environments [UbiCompForAll 2009].

The UbiCompForAll project is divided into two phases. The first phase is the engineering phase, wherein the focus is on developing engineering solutions and assessing them through the development and demonstration of pilot services. Later, in the second part, real services will be developed in an experimentation phase, and end users will be involved in the assessment. The end result of the UbiCompForAll project should be an extensive infrastructure that is intuitive for ordinary end users, while still sophisticated enough to handle all aspects of service composition.

As part of the UbiCompForAll project, a number of different scenarios that require further exploration have been identified. One of these scenarios, the *city guide scenario*, is related to tourism and people visiting a city. The different ways groups of tourists and other visitors can use modern technology as a tool to guide them around a city and how to create such tools through end-user programming are the main parts of this scenario.

The intention of this thesis is to design and assess a user interface for a service composition tool in that particular scenario.

1.2 Research questions

Two research questions have been identified, which this thesis intends to answer

- How can we design end-user tools that enable specification of services related to the «city guide» scenario?
- Can services that are specified by the user be realized in practice?

1.3 Goal and problem definition

This thesis will focus on tourism and the city guide scenario and the goal of this project is to develop and validate an interface for a service composition tool in this service scenario. The interface should be both simple in use, extensive and adaptable. This goal will be met by developing a number of scenarios to help create the user interface. Further, the user interface will go through usability testing to assess and improve it.

Firstly, a number of usage and composition scenarios will be developed. The usage scenarios describe the main ways tourists can be supported by software on their mobile devices when visiting a city, and the composition scenarios show how this software is composed and facilitated for individual users and groups. The usage scenarios describe potential variants of the services showing the relevance of adapting and composing services to different user needs.

These scenarios also help identifying building blocks that will be part of the composition environment. Building blocks in this case can mean any modules in a system, and in most cases will offer a service, for instance a map service that generates and sends map tiles.

A graphical user interface for the tool used to compose tourism related services will then be developed and go through usability testing involving end-users.

The results of the usability test will be exploited to further improve this interface, and as a final step in the assessment a simple architecture of a complete system will be created, to check that all the information necessary for creating the composed services can be provided when using the proposed user interface, and to see how the composition tool would be part of a bigger architecture.

By assessing the user interface using well-executed and recognized evaluation methods we hope to properly answer the first research question, showing how an interface can enable the specification of services in the city guide scenario.

The next part of the assessment, where an architecture will be created, should help answer the second research question, showing whether the services specified by the users are realizable in practice.

1.4 Report outline

Chapter 2: Background is an introduction to concepts relevant to the thesis and shows similar solutions.

Chapter 3: Research agenda and evaluation approach discusses the types of research and assessment that will be done in this thesis.

Chapter 4: Requirement specification and scenarios introduces a number of scenarios and lists requirements of the composition tool.

Chapter 5: Service composition tool describes the choices behind the user interface for the composition tool and the process of creating it.

Chapter 6: Assessment through paper prototyping goes through the paper prototype assessment of the user interface.

Chapter 7: Assessment through realizability study discusses the realizability of the service specified using the paper prototype and describes a system framework

Chapter 8: Discussion and further work concludes this thesis, sums up the results and describes the further work that can be done based on it.

Chapter 2: Background

This chapter introduces concepts relevant to the thesis. It will also describe similar existing solutions and the state of the art.

We will first introduce the areas of ubiquitous computing, service composition and end-user programming, which are all mentioned in the goals for the UbiCompForAll project. We then present Wireless Trondheim, which has been chosen as the experimentation infrastructure of UbiCompForAll. Finally, some projects for developing similar mobile city guide solutions are presented, which will be of interest for the usage scenarios that will be developed in this thesis.

2.1 Ubiquitous Computing

> *The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*
>
> **Mark Weiser [Weiser 1991]**

Mark Weiser largely defined the concept of ubiquitous computing, and this quote goes a long way of summing up what it stands for. His vision was that technology would eventually be so integrated into our daily life that we are not even aware of the fact that we are using it, that there would be computers surrounding us everywhere without us even knowing they were there.

This can still seem like visions of things to come, but in many areas the principles of ubiquitous computing have started to appear, if not yet as a completely integrated part of our environment. Modern mobile devices and wireless networks such as Wireless Trondheim [Andresen 2007] give people access to updated information and services almost everywhere they go.

However, an important part of ubiquitous computing is that the ability to access information should be like second nature, without having to navigate through difficult technological hurdles. A real research effort thus needs to be undertaken to integrate software with this existing hardware, with the focus being on how users can be assisted in the most transparent way, and not on the technology itself.

2.2 Services and service composition

Service composition is the process of creating a new solution based on a number of existing services. It aims at providing effective ways to create, run, adapt, and maintain services that are reliant on other services in some way.

In order for service composition to deliver on its promises, there is a need for development tools incorporating high-level abstractions for facilitating, or even automating, the tasks associated with

service composition. Hence, these tools should provide the infrastructure for enabling the design and execution of composite services [Benatallah 2005].

The term “service composition” can in some cases be attributed to automatic discovery and composition of services done by a computer program. However, in the context of this thesis the composition of services is always initiated by an end-user. Nonetheless, the user will still be guided by the program when composing, which makes it more of a supported process than a completely manual composition would be. Automatic composition as a form of service composition is beyond the scope of this thesis.

A service in this sense can be any software component, data or hardware resource on a device that is accessible by others. One example of a service can be the GPS on a mobile device, which will provide the location coordinates of that device [Chakraborty 2005].

Service composition will often involve discovering, integrating and executing existing services. It can also mean using and combining existing services to create a new service that will fit better to the needs of the user. One way of doing this composition is through end-user programming.

2.2.1 End-user programming

End-user programming is a way for users without professional programming experience to create, modify or integrate software applications to fulfill their own requirements. This is usually done with high-level and visual tools that simplify complex components for the user and uses simple terms related to the task when identifying the building blocks of the system.

The term “end-user” will in this case reference the person who is creating the system, and not necessarily the person who will be using it. In this project, the end-user can for instance be a person at the local tourist office, while the person using the system will be the visitor to the city.

Software for end-user programming can be divided into a number of categories, based on the user interface they feature [Stav 2006].

1. **General purpose programming languages**
   These are generally only appropriate for professional programmers, and even though efforts like Microsoft’s Visual Basic have simplified parts of the process, for instance by featuring an interface builder. However, the underlying logic must still be managed by the end-user using traditional programming which makes this method unsuitable for most people.

2. **Scripting**
   Scripting languages are generally easier to learn and simpler to use than traditional programming languages. One example of a widespread use of scripting language in this context is Microsoft’s Visual Basic for Applications (VBA). This is used in various Microsoft Office programs to let the user create powerful macros that can become relatively powerful solutions.

3. **Macro Recording**

Another way to create macros is to record sequences of user actions which can then be repeated. This is very simple, but is often not suitable because the actions will be too specific and it's difficult to modify a prerecorded macro because it is stored as a script.

4. **Programming by example**

   In many ways similar to macro recording, programming by example (PBE) aims to make macros suitable for more general tasks. The programmer demonstrates what she intends to do, and uses things like example rows and guide objects to interact with items that will have different values at runtime. The system uses a form of AI to then derive a generalisation of the problem from the examples.

5. **Grid based**

   In grid based programming, the application is basically a grid with a number of "actors" in the cells that interact in different ways. The actors will follow a specified number of rules and everything is laid out clearly with no hidden logic.

6. **Visual programming**

   A visual programming language (VPL) can be any type of programming language that lets the user create programs graphically, by manipulating different elements, instead of specifying everything textually. This also includes dataflow programming, wherein a program can be modeled as a graph of the data flowing between the operations.

   VPLs can be divided into visually transformed languages, which are regular non-visual languages with a visual interface placed over them, and naturally visual languages, which only have the inherent visual interface with no equivalent textual representation.

   Current trends in visual programming aim to integrate visual programming with dataflow programming languages in order to better model program states and support the growing need for parallelization [Johnston 2004].

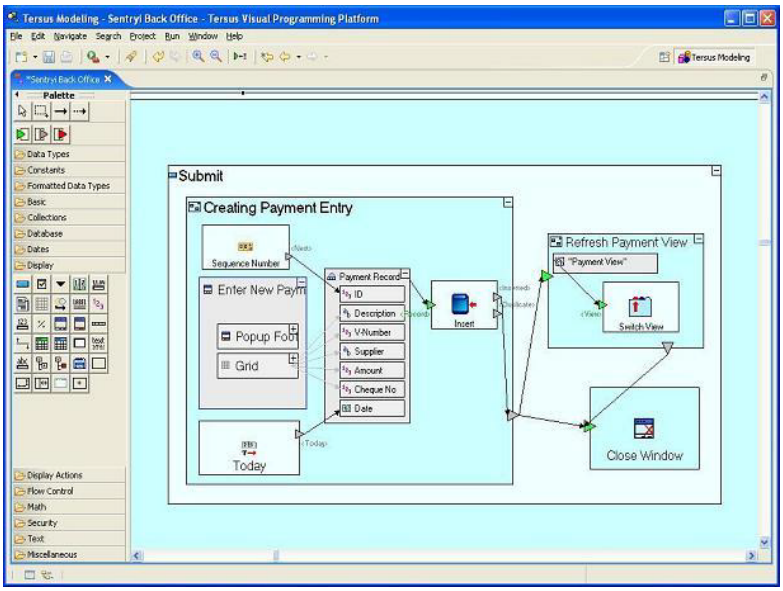


Figure 1: Example of visual programming interface for creating web applications [Tersus 2009]

Even though many VPLs are able to represent all aspects of a program visually, these programs can be harder to read and work with than those that use text for labels and some atomic operations.

7. **Other approaches**

   There are also other ways to create applications for end-user programming. One of the most successful approaches has been spreadsheets. Using simple formulas to work on rows and columns of a grid, the range of applications suitable for spreadsheets is quite limited, but this also makes the concept simple to understand for users.

The visual programming approach will be the basis of the interface to be created for this project.

2.3 Wireless Trondheim

Wireless Trondheim [Andresen 2007] is a Wi-Fi wireless computer network encompassing much of the Trondheim city center, especially the more popular public areas.

Created as a coalition between NTNU and local officials and businesses, it serves as a huge laboratory that can work as a testing ground for new services and equipment, and it gives people in Trondheim a low-cost way to access the Internet from cafes, restaurants and public spaces in the city. Coverage is not complete, but it does cover the most popular shopping and dining areas of the city.

2.4 Existing mobile city guide solutions

This section will look at some of the present solutions for mobile city guides and related service composition.

The field of mobile tourism is relatively new, as a result of the relatively recent growth of the user base with high capability mobile phones. However, using mobile devices to support tourists is not in itself a new idea, as mobile audio guides have been used in museum environments for years.

There are several providers of mobile location- and tourism based software at the moment. However, most of these either give a single pre-created solution, with little or no opportunities for individual tailoring, or they aim to create an individual user through location aware systems and intelligent agents that guess user preferences. Neither of these approaches are what this project intends to do.

There are however some projects that do offer similar types of user tailoring to the one our project seeks to create:

2.4.1 myMytileneCity

The result of the myMytileneCity project [Kenteris 2007], was a system that let users select the tourist content they were interested in from a website and then generate a standalone J2ME application for their mobile phone.

It differed from traditional approaches in that it worked "offline", not requiring a live network connection to receive data, because everything was compiled into the application at design time before the application was "pushed" to the mobile device.

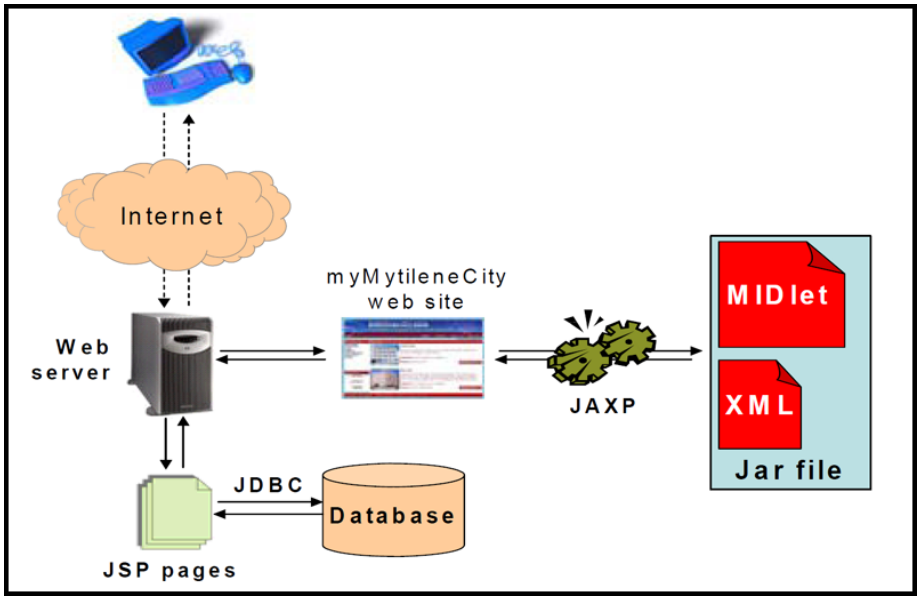


Figure 2: The myMytileneCity setup

This system has some similarities to the type of system we aim to create for our project; mainly that a service is specified individually from a separate user interface, and based on this a program for their mobile device is created.

However, even though there is an element of service composition, it is quite limited in scope and ambition, with the user mostly selecting from a number of options on a website through a “wizard” type guide.

The project’s explicit aim to create network-independent offline applications is also different from the goals of our project. While the reasoning for this was the high cost of WAP type traffic, for our project, the cheap and high bandwidth network capabilities of Wireless Trondheim can assist in creating highly dynamic applications that constantly communicate with the world.

2.4.2 The GUIDE project

GUIDE [Cheverst 2000] was an early attempt at creating a tailored tool for visitors to a city, combining mobile computing with wireless infrastructure. It took place in the city of Lancaster, UK and let users get information through a browser based interface. Interactive services and dynamic information was used, running on big and clunky portable devices using an older cell-based wireless network.

This early project had some interesting results, but technology has changed much since the project began, making some of the findings outdated.

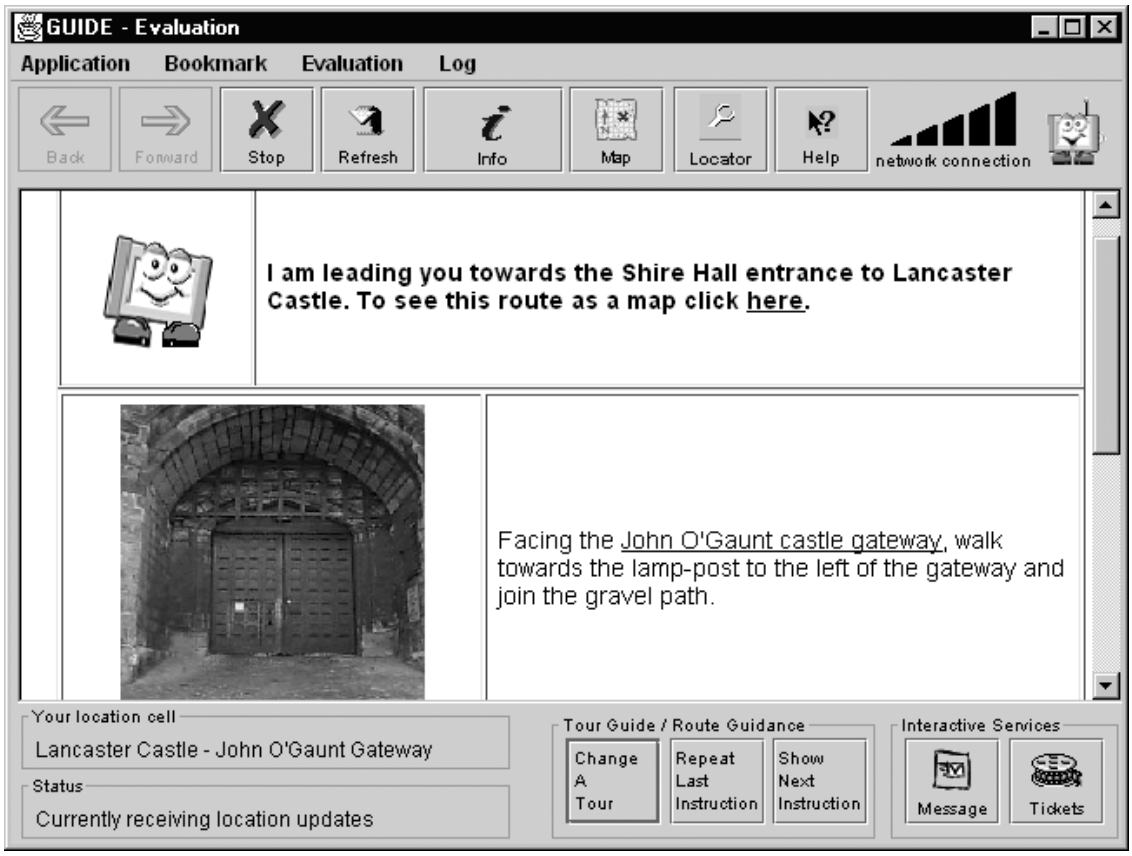


Figure 3: The GUIDE system's interface and guiding functionality

There was a certain amount of tailoring, where users could create personalized guides, but everything had to be done through the mobile unit's web-like user interface, and the amount of tailoring opportunities was limited.

Conclusions from trials during the project found quite a large user acceptance for this technology, although the system's flexibility also confused and bewildered users, indicating a need to have users be able to choose the level of functionality they require.

2.4.3 What makes the work of this thesis unique?

So far, most existing tourism software will normally only give a single pre-created solution that is common to all end-users, with few or no opportunities for individual tailoring. Other systems aim to let the users themselves control every aspect of the application, but they are not created for regular guided tourist groups.

Tailoring software is a complicated process, and will often require domain specific programs to be successful. This project will develop and test a user interface for such a program, the ideas and results from this thesis can then be further developed into a functional architecture.

There have been some previous projects with similar aims as our thesis. The results of these projects are interesting, and are valuable as research material, but as described in the previous subchapters, these projects have generally not had the same goals as this project has.

Chapter 3: Research agenda and evaluation approach

This chapter discusses the types of research and evaluation that will be conducted in this project.

3.1 Design Science

The research method used for this project is design science. This is a research methodology that offers guidelines for evaluating IT-related research projects. The focus is on improvement of the functional performance of an artifact, where the artifact can be a number of different things. There are seven specific guidelines for evaluation and iteration provided in [Hevner 2004]:

Guideline	Description
1: Design as an Artifact	Design science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
2: Problem Relevance	The objective of design science research is to develop technology-based solutions to important and relevant business problems.
3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations and/or design methodologies.
5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

***Table 3.1**: Evaluation and Iteration Guidelines* from [Hevner 2004]

Following these guidelines, design science can be applied to this project as following:

- **Guideline 1** demands an actual artifact be produced.
  In this case the artifacts will be the paper prototype of the composition software user interface, and the architecture created in order to verify the completeness of the user interface.

- **Guideline 2** requires the problem to be a relevant and important business problem.
  The initial work done by the UbiCompForAll project has identified a number of areas that require further study, of which this the city guide area this thesis is based on is one.

The growing amount of intelligent objects in our environment also introduces new challenges in having the services accessible for everyday users. Research in this area is still limited, and it is thought to be an area that will see significant progress in the near future.

The ICT Work Programme from the European Commission has initiated funding for research into the areas of "Service Architectures and Platforms for the Future Internet ", which covers the areas of this thesis, specifically defined as "Service front ends enabling communities of networked users easily to compose, configure, share and use services and providing device and context aware service adaptations" [ICT 2009].

- **Guideline 3** stresses the need to use well-executed evaluation methods.
  For this project, the main evaluation method is usability testing using paper prototypes. This is a tried and tested method of evaluating user interfaces, as confirmed by a July 2002 survey of usability specialists, where 86% considered paper prototyping "essential" or "useful" [Snyder 2003].
  Basing the prototypes on recommendations from respected authors in the field ensures the quality of the testing is upheld.

  As an additional evaluation, creating a basic architecture for the system will demonstrate how the user interface leads to a complete application. This way of evaluating the interface through a proof-of-concept process is also a tried and tested method.

- **Guideline 4** calls for the design-science research to provide a clear and verifiable contribution.
  The background in chapter 2 lists similar existing services for mobile tourism, but there is a clear lack of applications for creating individually tailored solutions within this domain.

- **Guideline 5** requires the use of rigorous methods.
  In this project, the methods for construction and evaluation are chosen based on a study of various options given the requirements, and informed input from the project advisors.

  Literature that is considered *the* standard of providing guidance in creating and evaluating paper prototypes will be used to ensure methods are withheld to the highest standards [Snyder 2003].

- **Guideline 6** expresses the need to utilize the findings and use them to create a better product.
  This is very important, and for this project the interface prototype will be further evaluated in itself by checking the realizability of the specified service. To that end, we propose a service architecture, to check that enough information has been provided during composition to create a service. This architecture will also serve as a basis for further research at later stage in other student projects (e.g. programming projects.)

- **Guideline 7** will be followed by explaining uncommon technical terms used and keeping in mind the inteded audience while writing the report.

The result of this project will mainly be evaluated through usability testing using paper prototypes. The prototypes will undergo minor changes during testing. Further, an architecture will be proposed to help check that all the information necessary for creating the composed services can be provided when using the proposed user interface.

3.2 Evaluation approach

There are several different ways in which the artifacts of this project will be evaluated.

3.2.1 Scenarios

The created scenarios will be evaluated and improved by interviewing professionals in the tourism industry. The feedback from these people, executives at tourist offices, is essential in ensuring that the scenarios are realistic and reflect common needs and behaviors of the tourists visiting the city. After creating the scenarios and modifying them based on the discussions with industry professionals, the scenarios will serve as a basis for creating the graphical interface and identifying relevant building blocks and functionality of the system.

3.2.2 Graphical user interface

The graphical user interfaces will be evaluated through usability testing. In this thesis, we have exploited paper prototyping.

Paper prototyping is a form of evaluation of a graphical user interface where representative users interact with a paper version of the interface. The users will follow a precreated scenario which describes the tasks the users should try to accomplish. When the user is interacting with the user interface, a person will act as the computer and change the paper slides to show the result of the interaction.

This form of testing makes it easy to detect usability issues early in the design process, and rapidly create updated versions of the user interface. Conducting these tests early in the process is essential, to make sure the design is as flexible as possible, and fundamental design elements still can be changed.

3.2.3 Implementation and architecture

The final part of the evaluation will be to check if it is possible to create the desired services by using the graphical user interface. That is, to ensure that the user interface gives sufficient input data that the system can be built in the background. This will be done by specifying programmatically the different classes, variables and links in the system and map out how the input from the GUI will affect these classes and contribute to the new system.

In addition, a basic architecture of a completed system will be presented, along with a proposed way of converting the data from the end-user programming into a runable service. There will also be a comparison with similar existing solutions to see how the model and architecture differs.

Chapter 4: Requirement specification and scenarios

This chapter goes deeper into the details of the project, giving some scenarios to demonstrate use and listing the requirements of the composed services and the composition tool.

4.1 Usage scenarios

In order to better understand the services that will be composed, and to identify requirements and features of the composed system, two scenarios have been created. These should show how users will use the composed services in practice, and be representative for standard usage patterns. It should be possible to define different variants of the services, thus creating a need for adapting and composing services to different user needs. In addition, they should help identify the individual components that should be available for the end-user programmer to create the programs with.

Further, important knowledge to extract from the scenarios is building blocks that are used in different types of services. Building blocks can be services that are re-used in different ways during composition into new solutions. To help identify these building blocks, some variants to each scenario have been created. These variants will show how the same components can be used in different contexts.

4.1.1 Scenario 1: Corporate group

It is a Saturday morning in May, and members of Statoil's marketing offices around the world are in the city for a conference. 10 of the members have some free time and want a guided tour around the city.

One of the group members is Mona, who works at Statoil's marketing office in Germany. She shares similar interests with the other people in the group, some of which are close colleagues of her, and they all decide to walk the same basic route.

They are using a city guide service for their visit in the city, created by Alice at the local tourist office. The group expressed an interest in seeing the most popular sights, but also had some special requests to see sights showing examples of special architecture in the city. They plan to spend around 6 hours walking around the town, but won't necessarily always walk together as a single group.

Mona starts her mobile device and opens the "City Guide" application. She can see a map with directions to the first sight on their agenda, the church ruins of Olavskirken. Most of the group walks together to the first stop on their route, following the map on their device with directions in their own language.

At their first stop, there are no local guides that speak German, and all posted information is in English or Norwegian. Mona uses her mobile device to give her information in German about the different artifacts at the church ruins.

After touring the ruins for a while, Mona is ready to move on to the next sight. She presses the "Next unvisited" button, and automatically gets directions to the next planned sight. She enjoys having the

ability to spend as much or little time as she wants at the different sights, instead of always having to walk with the entire group.

The group has planned to meet for lunch at noon. At the tourist office, Alice had recommended a restaurant which they all agreed upon, and programmed in a meeting at noon, with automatic reminders and walking directions sent to all members.

At 11:27, while Mona is touring NTNU, she gets a message from the system that lunch is in 33 minutes at "Mormors Stue". It shows her the directions to the eating place, and estimates the time to get there at 23 minutes at normal walking pace. The alerts are timed to give the group members enough time to get to the eating place depending on how far away they are.

After lunch, the group continues their sightseeing. Mona decides to split from the other tourists and visit an art gallery instead. She uses the "Culture" part of the system to search for nearby museums, and gets directions to "Trondheim Kunstmuseum".

Finally, Mona is tired of walking around and wants to get to their hotel. The system shows it is 2 km away by foot. This is a bit too far to walk right now, so presses the "Taxi" button on her device, which orders a taxi to her current location. The system informs her the taxi will arrive within 10 minutes. She gets in the taxi and tells the driver the name of the hotel.

Variants of this scenario:

- Tourists could instead of walking together as one big group, be divided into smaller groups of 2-5 people which share the same route and can use intra-group communication
- If some close friends are together in a bigger group, they can specify in the system who their close friends are and have the device show only them on a map, instead of the bigger group
- A group leader has the ability to see which waypoints have been visited, and e.g. in the case of a school trip can verify that all students have visited all required sights
- The route can be "locked in" requiring people to visit sights in a certain sequence, or they can be "free", letting people visit the points they want to in the order they want
- Tourists can see on the map if others from the group are nearby, and meet for e.g. an impromptu lunch. The system can also send messages to the X nearest people to schedule a meeting
- The system can record the route walked by the users, so they can share this route with friends or retrace their steps later, or to "tag" the location of photos they've taken
- There can be a quiz-type event associated with each waypoint/sight, which for instance can be used as a trivia contest
- One can get automatic reminders of big events that are about to start nearby

4.1.2 Scenario 2: Arriving by car

Before she, her husband and their two kids of 10 and 13 go on a road trip to Trondheim, Caroline composes a service to help them navigate the city, using her computer at home. She does not want to make a preset route of sights to visit, but wants instead to be able to spontaneously search for and find places to visit. However, to quickly find a parking spot when she arrives, she includes the guiding functionality and adds the parking lots that are nearest to the city center.

When they get close to the city, Caroline has the system guide them to the best parking lot.

Caroline and her husband want to visit some art galleries. To keep the kids from getting bored, Caroline wants to find something for them to do in the meantime. She opens her mobile device and checks out what is happening in the city today, looking for an appropriate activity to send the kids to.

She notices “Vitensenteret” is open, and the kids seem eager to visit the place, so she sends them on their way, giving the oldest kid a mobile phone to help them find their way, and to let Caroline keep track of exactly where the kids are and be able to communicate with them.

While walking to the first gallery, Caroline falls and scrubs her knee. She notices it is bleeding, so she needs to find a place to buy band-aids. Using the “Find Nearest” service on her mobile, she finds the nearest pharmacy which luckily is only 100m away, where she buys some band-aids and disinfectant.

After walking around in some galleries for a while, Caroline gets an alert from the system that the kids have left “Vitensenteret”. She can see the kids are moving on the map, and sends them her current location so they can meet up at the gallery before deciding on where to eat dinner.

After meeting up, they decide to eat at the Peppes Pizza restaurant, and the system guides them there. After visiting the restaurant, Caroline was so pleased with her meal and the staff that she wanted to write a review. She opened the feedback part of the service, gave the restaurant a score of 9/10 and wrote a glowing review.

When they are ready to return to their car, the system lets them know exactly which route to take to get back to the parking lot.

Variants of this scenario:

- Instead of looking through the list of restaurants and picking one, the system can be set to suggest restaurants to visit automatically when the family walks near them, and one can specify beforehand what kind of restaurants are interesting/relevant
- In a similar vein, the system can also be set to automatically suggest interesting sights nearby to visit

- In addition to the parents seeing where the kids are, the system could also show the kids where the parents are
- The “Find nearest”-service can also find the nearest restroom/kiosk/parking/tourist info office etc.

4.1.3 Identified building blocks

Based on the scenarios in 4.1.1 and 4.1.2, a number of common building blocks have been identified. These will serve as a basis when the building blocks to use for the service composition tool will be chosen.

Related to location:

- Location (GPS/WiFi positioning/etc)
- Map (with different types supported)
- Navigation (by foot/by car)
- A route creator

Data on waypoints:

- General points of interest
- Tourist attractions/historical places
- Places to eat, drink, go out
- Detailed POI information (incl audio/text-to-speech)
- Multi-language support

Time/calendar:

- Reminder type “Event” building block
- Calendar/daily schedule

User and groups:

- User manager keeping track of logins etc.
- Group manager to organize users
- Group messaging and other communication

Interaction/feedback:

- Info on current events in the city (“which movies are playing?”)
- Ticket ordering for movies/other events
- Bus/taxi services

4.2 Composition scenarios

To give the tourists a better and more individual experience, Alice at the local tourist office was asked to create a specialized mobile service for the Statoil corporate group visiting the city.

To create the service, Alice opens the Service Composer and uses the “Guided tour” element as a starting point. She adds a “Group”, and adds all the group members to it. The members will not be

divided so they all go in the same group. She then adds the Language property for each group member and sets it to their preferred language.

To satisfy the wishes of which sights the group members want to see, she specifies for the “Main Guide” that it should include all sights related to “architecture” as well as the sights in the “Main sights” group. She then adds functions for “Group messages” and “Eating”, before she finally schedules lunch for the group and schedules reminders for all group members

4.3 Feedback from tourism experts

As part of assessing the scenarios and other parts of the project, meetings were held with people at the tourist information offices in Trondheim and Bergen. In Trondheim, with managing director of Trondheim Tourist Information, Jens Fredrik von der Lippe, and in Bergen, with manager of the Tourist Information in Bergen, Inger Balean.

In the meetings, the tourism professionals were first presented with the general concept of tourists using mobile devices to assist them, and then with the concept of composition and individual tailoring. Further, the scenarios were discussed and general issues surrounding issues like target groups were brought up.

From the meetings, a number of points and observations were made:

- A general feeling that a program on a mobile device could not be a replacement for human guides. The human interaction, the ability to ask questions and get immediate feedback, the connection to local people were all important elements that are hard to recreate in a mobile service.
- The original scenarios involved tourists arriving in the city as part of a package holiday. This was thought of as unrealistic, as these types of trips were organized far in advance, were on a tight schedule and did not allow for much individual tailoring.
- A better target group would be visitors coming to the city as part of a business conference or meetings. These people would often have spare time to explore the city, and could want to go in smaller groups for a more individualized experience. They could also be more comfortable with new technology and willing to try out new things.
  In addition, meetings and conferences with business travelers bring in a lot more money to the city than regular tourists, so new technology could work as a selling point when trying to get conferences to the city.
- Individual tourists would also be a worthwile target group for this type of service. Tourist offices often have tourists coming in asking for recommendations on what to do, often seeking suggestions based on their interests. This could be a good place for a tailored mobile service.

All in all, the people at the tourist offices were initially somewhat skeptical to the general concept, but after discussing it and properly explaining it they did see a potential in such a service.

4.4 Requirements specification

In this section, the identified requirements are introduced. First, the identified requirements for the mobile software are briefly listed. From these requirements, as well as the composition scenarios, the input from the tourist office representatives and the UbiCompForAll project leaders, the functional and non-functional requirements for the composition software are identified and listed and designated with priorities of high, medium or low.

4.4.1 Composed service requirements

- Users must be able to see a map
- Users must be able to walk both freely along the route or in a set order
- Users must be allowed to visit sights independently of each other
- Positions of other people in the group must be visible in the map
- Sights defined in the route must be visible on the map
- A way of seeing the route visually to the selected destination must be visible on the map
- Users must be able to get information in their own language
- Users must be able to go to other sights than the ones in the route
- Users must be able to call a taxi
- There should be options for surveillance of selected users (e.g. to look after kids)
- Users must be able to send messages to other users

4.4.2 Functional requirements for composition tool

Number	Requirement	Priority
1	It must be possible to create guided tours	H
2	It must be possible to organize persons in a group	H
3	The guided tour must have options to specify either walking freely or in a set order	M
4	There must be support for multiple languages	M
5	There must be a way to organize sights in a route	H
6	User interface must be based on visual programming principles	H
7	Users must be able to add and define new types of elements	M

4.4.3 Non-functional requirements for composition tool

Number	Requirement	Priority
8	System must have high usability, making it easy to understand and use for all intended users	H
9	The tool must be easily extendable to accommodate new types of elements (extensibility)	M

Chapter 5: Service composition tool

This chapter introduces and discusses the choices made in regard to the user interface of the service composition tool. It will also show the process of creating and refining this interface before the evaluation process.

5.1 General theory

After finalizing the scenarios, the next step was to create the user interface for a service composition application. This involved two main areas of exploration. First, the various components necessary to fulfill the scenarios and requirements would be chosen, and then the graphical user interface featuring the components would be created.

It was decided to select an environment flexible enough to ensure as wide a variety of different composed services as practically possible could be made. This would also include solutions involving services and components that had not yet been thought of or created, that should still be supported seamlessly by the user interface.

This decision introduces a number of challenges. Because another requirement was that the design should be easily understood and used by people not familiar with software development, the added complexity of a more flexible system should ideally still not be detrimental to the usability of the system.

5.2 Basic components

Using the created scenarios as a basis, a number of modules or classes were specified, with the emphasis on having all the necessary components to create all the service types described in the scenarios.

One important decision was to experiment a so called “Event” module that could be used quite flexibly as a way of allowing various types of messages and other data to be transferred between modules. Having an event-type component available can help users easier understand the principles of programming, as findings suggest people tend to think of events, and responses to events, when solving problems naturally, before they have been exposed to programming [Myers 2004].

Another decision was to allow for a library of different components to be used in the composition. It would be possible to add and remove components from the library, and to create new components from scratch to be used in the application framework.

The basic components we would use initially were these:

- **Tour**
  - A tour would consist of a Route, a Group and a Map. In addition it would have various properties associated with it.
- **Route**
  - A route would be a collection of sights, and various properties.

- **Group**
  - A group would be a colletion of people, and various properties.
- **Sight**
  - A sight could be any place of interest, including both tourist sights and restaurants. It would have location data with coordinates to allow it to be displayed on a map, in addition to many other properties.
- **Person**
  - A person would have a number of properties, including constantly updating location coordinates.
- **Event**
  - An event would allow messages and data to pass between modules. It would connect between any two modules, with properties depending on the type of event chosen, but for instance it could pass along location information and reminders about a place to eat to a person.
- **Map**
  - A map, allowing for different types of maps to be used, according to requirements for things like detail level, street names and satellite photos.

5.3 User interface

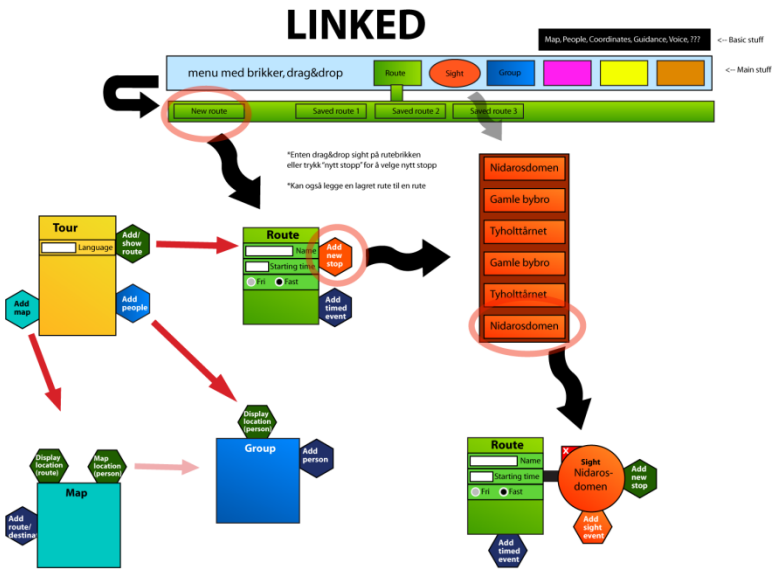
From the list of the different types of interfaces used in end-user programming shown in chapter 2, the decision was made to base the user interface for this task on the principles of visual programming.

The choice of trying out a visual interface was based on an interest expressed from the UbiCompForAll project to explore visual formalisms in this context of end-user programming.

In addition, the type of visual interface chosen was a quite simple one, which was a choice we made based on the assumption that the relatively uncomplicated nature of the composed solutions would make a simple interface sufficient, and that a simpler interface would help optimize usability for users unfamiliar with this and other types of programming.

5.3.1 Initial interface

After first identifying the different components needed, an initial draft of an interface was created.



Route+Map+People+Guiding=Tour

Figure 4: Initial user interface draft

This interface features the various modules as differently colored boxes that could be chosen from a menu. Every component had a number of "leaves" that would allow it to connect to other components, forming links. The type and number of leaves would depend on the component. The very first interface did not yet have the "Event" module, but it soon became apparent that for added flexibility, a generic module that could link any two modules was needed.

5.3.2 Revised interface

After some sessions discussing the interface with the project supervisors at SINTEF, the first draft evolved into a revised interface, which would work as a basis for the first paper prototype.

The main idea behind this interface is that the user opens various elements from a menu. Represented as a self-contained "box", these elements let the user specify various properties related to the element, and it allows the user to "link" an element to other elements using the leaf-like buttons on either side. The type of element one is allowed to link to is specified by the color and text of the leaf-button.

These are the main interface components that were created:

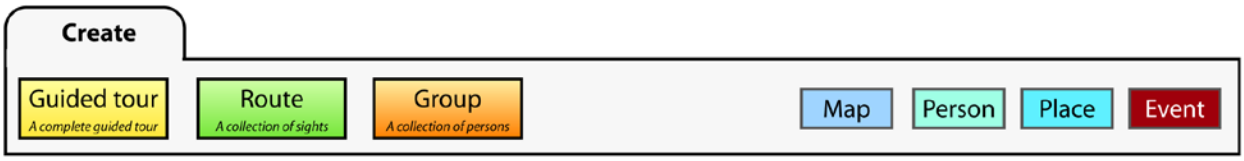


Figure 5: Create menu

The **Create** component is the initial menu where the user will find a list of different elements to create.

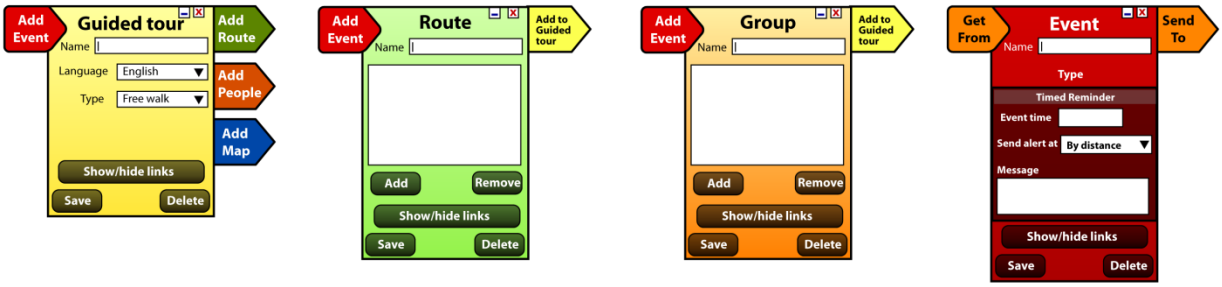


Figure 6: The main types of elements

In figure 6, the main elements used during the prototyping are shown.

Guided tour is the main element in this context. It is the basic element for creating guided tours, which are the most common type of service created for the tourists. A guided tour can be any type of service where people receive a list of places to visit with directions on how to get there. To complete a tour, this element needs to be linked to a **Route**, a **Group** and a **Map**. In addition it can be linked to an arbitrary number of **Event**s.

Route is basically a collection of places. In this context, a place can be any type of waypoint that can be displayed in a map, for instance a tourist attraction or a restaurant. A route is created by adding any number of places and sorting them in the preferred order.

Group is a collection of people. People are the basic users of the finished service, and one person can belong to several groups. Groups are created by adding persons to it from a list.

Map is just a map. Having the guided tours support different maps can be important depending on the types of places it would cover.

Event is a special type of element that enables any two elements to connect to each other and exchange data. The Event element has two “link” leaves, where one would be used as input to get data from one element, and the other as output to send the data.

The properties of the Event element lets one choose between a number of different behaviors, the illustrated one shows how one can receive a reminder to go somewhere, with the input being the place to go to and the output being the people who will receive the reminder.

After creating a new element, the user would fill in the name of the element and specify various options. Then she would click the different leaves of the element to link it to other elements.

Elements like **Route** and **Group** can also be created directly from the **Create** menu, instead of via the **Guided tour** element. This ensures that the elements can be created independently, and saved and reused in other compositions.

When links are created, they show up visually in the interface as links between elements. To avoid clutter, elements can be minimized and links can be turned off by clicking the “Show/hide links” button. When this button is clicked, it will show/hide all the elements that were linked to that element.

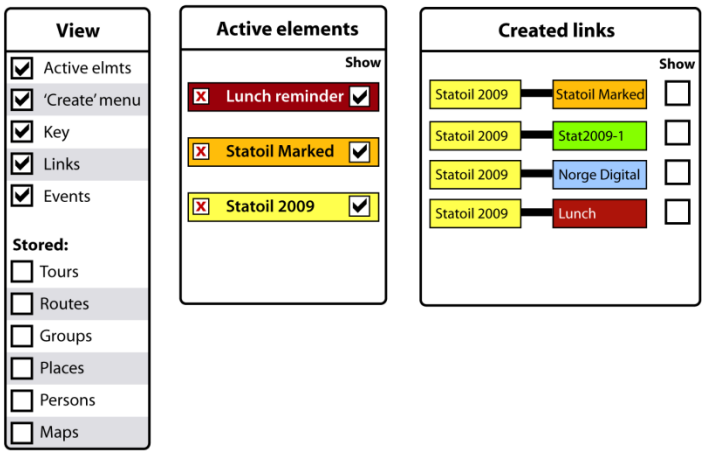


Figure 7: Other user interface elements

Various other user interface elements were also created, to assist the user when creating things. A **View** item would let the user turn on or off the various user interface elements. The **Active elements** list would show all open elements and let the user show or hide them to avoid clutter. The **Created links** list would show all links that had been created during the composition and let the user show them.

5.3.3 Interface in practice

To illustrate how this interface would work in a real-life situation, some steps of the script used for the paper prototyping assessment are shown with the accompanying user interface for fulfilling that step:

1. ***“You want to create a new guided tour for a group of visitors to the city.”***
   Here, the user will normally click the “Guided tour” option from the Create menu which opens a new, blank “Guided tour” element, as shown in figure 8.

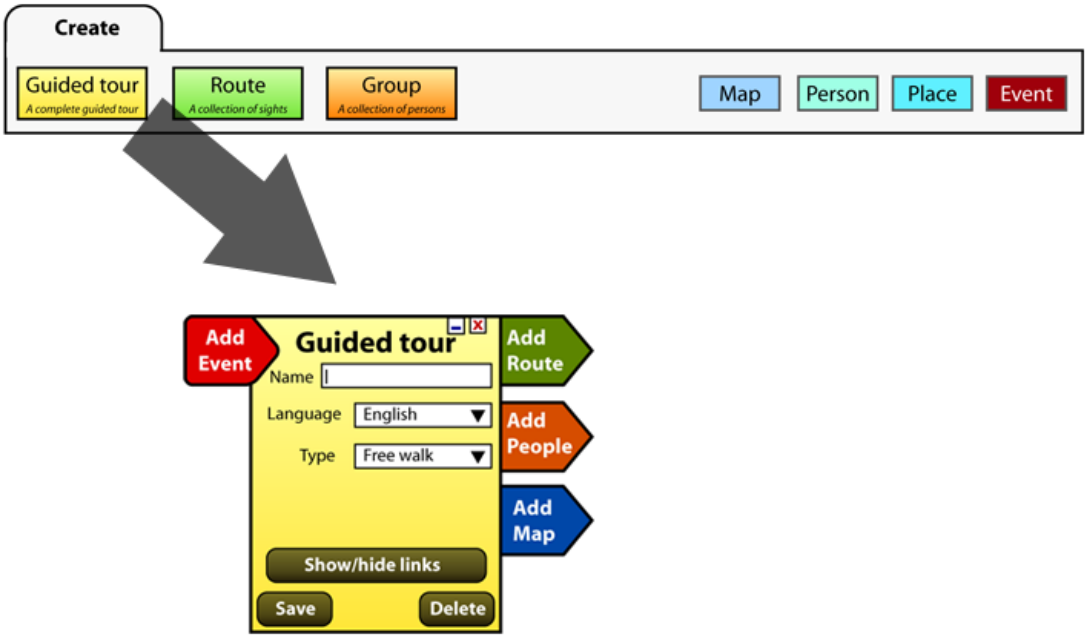


Figure 8: Creating a new guided tour

2. ***"The group consists of four persons: Per Pettersen, Ola Olsen, Lise Larsen and Trine Trulsen. They are in the city for a conference initiated by Statoil and have one day available for sightseeing."***

   Now the user can either create a new group from the Create menu (this must later be linked to the appropriate "Guided tour" element,) or she can click the "Add people" button on the newly created "Guided tour" element (later renamed "Add group".)

   In the latter case, a list of already existing groups will be shown, with the option to add a new, blank group. The link to the guided tour element will be automatically added to this group.

   The next step will be to click the "Add" button in the group element to open the "Stores persons" dialog which lets you add people to the group. If the person is not in the system, a new person can be added by clicking the create button.

   The latter process is shown in figure 9.

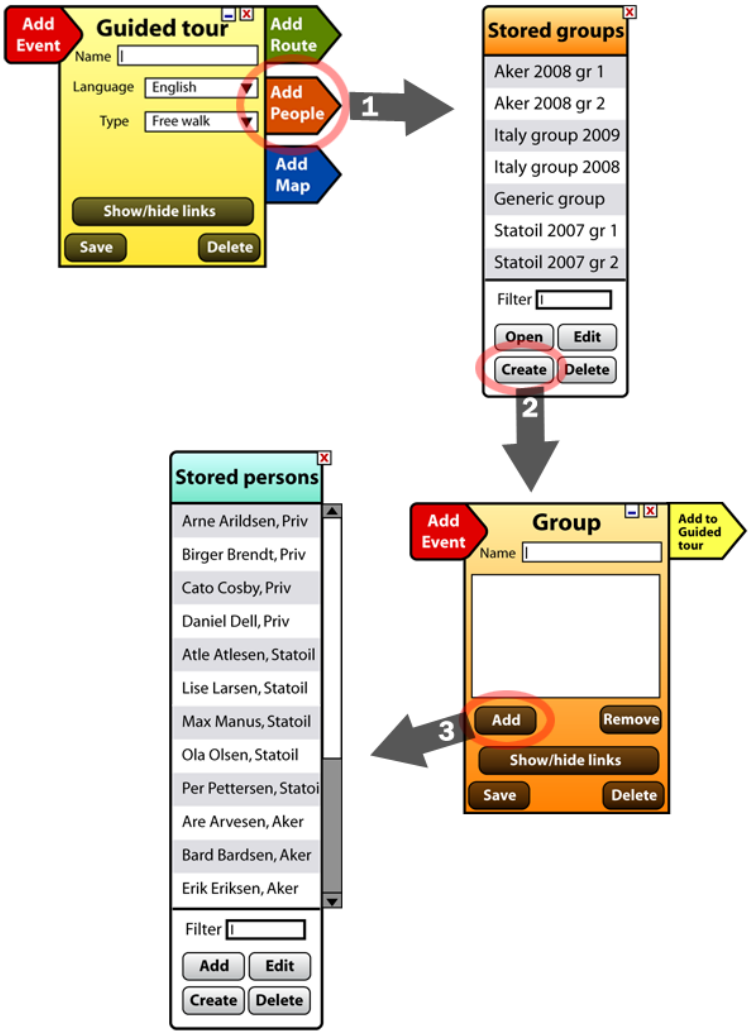


Figure 9: User clicks "Add people" which opens the stored groups list, then clicks "Create" to create a new Group element and then clicks “Add” to add people to the group.

3. ***"The group has expressed a number of requests about what they are interested in. You have picked five places they shall visit: The Nidaros Cathedral, Festningen, Trondheim Art Museum, Sverresborg Trøndelag Folkemuseum and the Tyholt tower."***
   This requires the user to create a new "Route" element. As before, this can be done either by using the Create menu and link to the guided tour later, or clicking directly on "Add route" on the guided tour element. As with adding a group, clicking the add route button opens the "Stored routes" dialog, where the user must either choose a pre-created route or click the create button to create a new one.

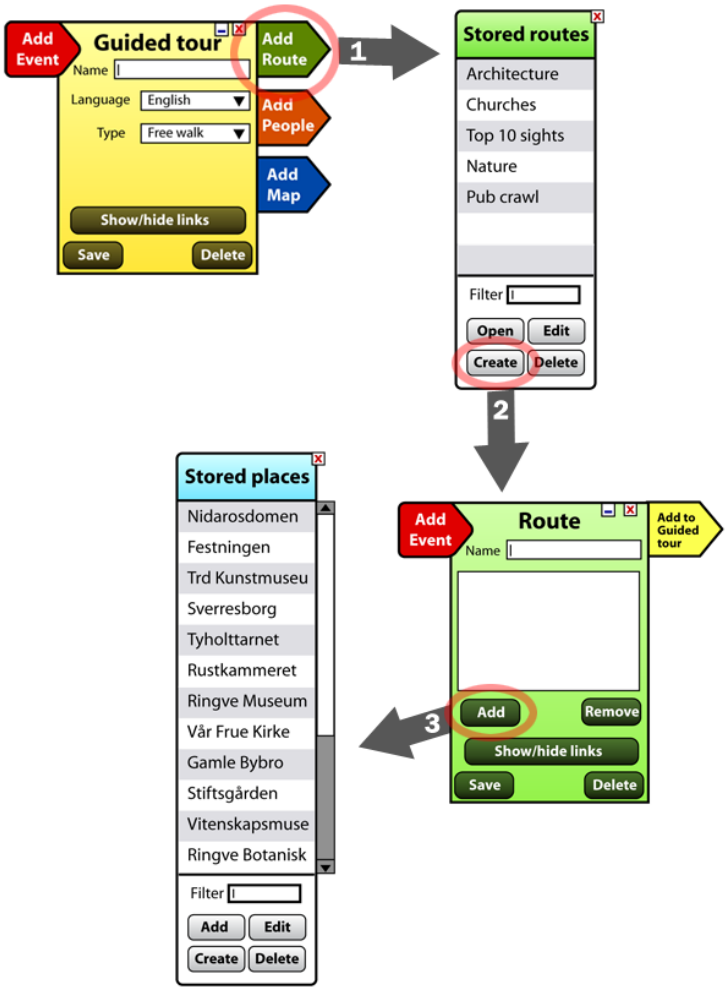


Figure 10: User clicks "Add route" which opens the stored routes list, then clicks "Create" to create a new Route element and then clicks "Add" to add places to the route.

Chapter 6: Assessment through paper prototyping

This chapter presents the usability assessment of the user interface.

A paper prototype is an interface created early in the design process that allows it to be tested on potential end users [Snyder 2003].

A paper prototype does not have to be particularly good looking or refined, as long as the basic concepts are evident. However, in this case the prototypes were designed on a computer, which made them look more finished than a hand drawn prototype would.

The revised interface introduced in section 5.3.2 went through three different sessions of assessment, and each time 2 to 3 different people performed the test. After each test session, the prototypes were given minor updates based on feedback from the testers.

6.1 The test

The subjects had to perform a number of tasks which involved creating a solution for a number of people visiting the city. They were given the following script, which was based on the composition scenario and the first usage scenario:

> *You want to create a new complete guided tour for a group of visitors to the city.*
>
> - *The group consists of four persons: Per Pettersen, Ola Olsen, Lise Larsen and Trine Trulsen. They are in the city for a conference initiated by Statoil and have one day available for sightseeing. Per is the leader of the group, which is called "Statoil Marked", and everyone wants to see each other on the map.*
> - *The group has expressed a number of requests about what they are interested in. You have picked five places they shall visit: The Nidaros Cathedral, Festningen, Trondheim Art Museum, Sverresborg Trøndelag Folkemuseum and the Tyholt tower. The route should be called Stat2009-1.*
> - *You realize the Nidaros Cathedral and Trondheim Art Museum are right next to each other, so you change the order of the route to ensure Trondheim Art Museum is put right after the Nidaros Cathedral.*
> - *You have already ordered lunch for the group at the Tyholt Tower at noon, and you want to give the group a reminder to ensure they are able to get there in time.*
> - *You want the group to use the map from Norge Digital, as this is the most detailed one.*

Some of the details in this script refer to specific property fields in the application. After familiarizing themselves with the script, the test subjects got a basic introduction to the principles of paper prototyping, and instructions on how to act and interact with the paper interface.

6.1.1 Test subjects

In choosing the subjects to perform the assessment, a decision was made to try to include several persons who had little to no knowledge about software development. Because the aim of both this project and the UbiCompForAll project in general is to have non-IT professionals be able to use the composition tools, we have chosen to involve persons with no knowledge of software development, although they still have general experience in using mobile phones and PCs.

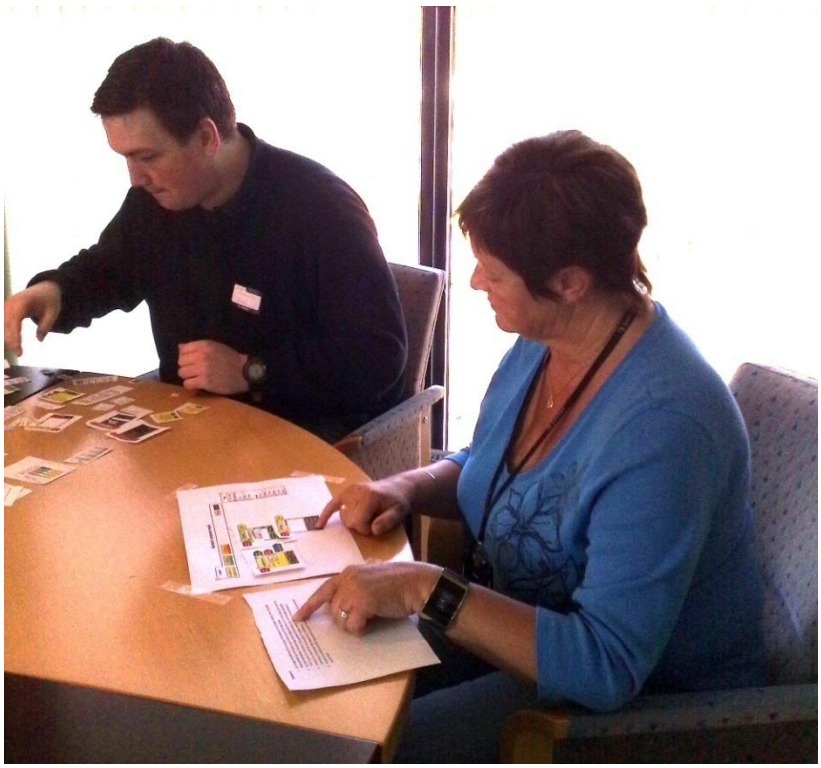


Figure 11: Conducting the paper prototype test

6.1.2 The process

The proposed user interface went through three iterations of assessment, each involving two users. Between iterations, revisions were made to the user interface as a response to issues that were uncovered.

During each test, three people were involved:

- The tester, who was performing the task given by interacting with the user interface.
- The person acting as the computer, who updated the interface according to input from the tester by changing out paper elements.

- The observer, who would take note of all issues coming up during testing, or other noteworthy events. Particularly situations were things seemed to “break down”, and testers were unable to understand or continue parts of the test, were important to document.

It was important that besides the introduction, the tester would not receive any help or guidance from other people, even if they asked for it. Even though the paper prototype looked far from a real computer based interface, the experience when using it should be as close as possible for the tester.

6.2 Results from tests and uncovered issues

As pointed out in [Snyder 2003], detailed reports from paper prototyping tests are rarely useful or practical. The most important thing is to clearly communicate the top issues that arose during the tests, for instance by listing them with a short summary. Using this list, the interface designer can then perform the required changes to the user interface.

In the following sub-chapters, the top issues arising during each test session are listed, including the changes to the interface done to overcome them.

6.2.1 First session

In the first session, two persons were tested. They had both good knowledge of software development and also worked in fields related to the testing.

Functional issues uncovered in this session:

- Users did not understand and were confused by some of the more technical options, for instance option to “Create map”, as only expert users would need such features.
- Some issues with naming; not clear what the word “Place” entails.
- Difficult to find items in a list, for instance a person in a list of persons, because items were not alphabetized.
- Some confusion surrounding the “Event” element.
- Wishes to see the result of the composition in an emulator type window.

Changes made before the next session:

- Create a system where regular users do not have access to advanced options - introduce different abstraction levels, where only domain experts have access to complicated options, and regular users only get the most used regular options. In practice, this involved removing complicated and unnecessary elements from the interface, elements that would not be needed by regular users.

- When adding a person to a group, users will have to select the properties of the person related to that particular group.

Figure 12: New property page when adding person to a group

- For lists of items, for instance the list of sights or list of people, always have them alphabetically sorted to make them easier to navigate.

The issue with not having abstraction levels quickly became apparent, and the next version sought to remove much of the confusion by giving regular users a simpler version of the interface, while leaving the advanced version for domain experts. For instance, in this new version, the option for users to create or edit new "Map" and "Place" elements was removed.

6.2.2 Second session

In the second session, both people familiar and unfamiliar with software development were tested.

Functional issues uncovered in this session:

- Unclear how to edit a person's group properties versus editing the person itself.
- More issues with naming of objects, for instance not clear that clicking "Add people" on a "Guided tour" object will in fact add a "Group" object,
- When searching for items in a list, users would overlook the "Filter" option that let them easily search for items in the list.
- Users were unclear on when they were able to edit the properties of a module and when they were not.
- Users often had questions about specifics around elements – what they were for and what they did.
- Users were not sure if places in a route were sorted, and how to change the order of places.
- More confusion surrounding the "Event" element.

Changes made before the next session:

- Changed naming of leaves to accurately reflect the type of module they will link to, for instance “Add group” instead of “Add people”.
- Rename “Filter” function to “Search” function, to see if it would get more use.
- Stricter limits on when something was editable were enforced. Users would now always have to click “Edit” on an object before being able to change any options.
- Added “Help”-function to all elements in the form of a “?”-button that would that would explain the element’s functionality when clicked on.
- Items with a specific order, like places in a route, were numbered to make it easier to see they were ordered. In addition, arrows to move items up and down in the list were added.

Figure 13: The "Places" element before and after changes based on the first two prototype tests. Changes include simpler and more logical button layout, renaming the “Filter” function, alphabetized list and added “?” button. The Add/View buttons will be grayed out when an item is not selected in the list.

As in the first test, there was some confusion in how to use the “Event” element. The added Help-functionality helped to a certain degree, but it turned out to be difficult to design the “Event”-element to be both as user friendly and as flexible as was wanted. In the end, flexibility won through; as it was thought simplifying the “Event” element too much would render it too unusable in practice.

Figure 14: Final version of the Event element, already somewhat simplified for the test

6.2.3 Third session

In the third session, two persons unfamiliar with software development were tested.

Functional issues uncovered in this session:

- Users want to see the route they create live in a map, and to be able to manipulate the route directly in the map instead of as a simple list of places.
- Users also still wish to see the result of what they create live in an emulator.
- Events are still difficult to understand. May need to create a simpler version after all, and use the existing Event functionality for more advanced users.
- Users did not use the help functionality. Using a question mark as symbol for help may not be logical for most users.
- Users rarely used the "Show links" functionality to display the connections between elements, and were not sure what it did. Users may need further instructions in how to use this function.

6.2.4 Test conclusions

The proposed interface of the paper prototype tests was generally easy to understand and use, as the subjects were usually able to compose the solution as intended based on the script given.

However, a number of issues with the user interface were uncovered:

- The need for information abstraction became very evident, as most of the advanced functionality would not ever be applicable to the average user, and only served as a confusing element. This could be accomplished by specifying different user levels, where advanced domain experts would be given access to all options, while the average user would have access to a limited set of options that would be sufficient in creating most or all common types of solutions.

- Several users requested a way to see the result of what they create directly, for instance through an emulator type element. Similarly, users requested an option to be able to create routes directly in a map, instantly seeing how the routes look and having the ability to change route properties from a map-type interface.

  This shows a clear potential of extending the interface to include such functionality. There seems to be a good case made for the added usability such features would bring with them, and letting the users see directly the results of what they do is in tune with basic user interface principles.

- The need to clearly define and label terms used in the interface, such as “Guided tour” and “Route” was necessary for users to understand what the different elements were used for.

The choice to have a completely visual tool seems to have been positive, as users were mostly successful in creating the solutions they intended to. For the users this project was inteded for, a complicated visual solution or a more text-based solution would probably require users to receive a lot more introduction on how to use the tools. Correspondingly, with a simple visual solution users were able to understand most of the concepts with hardly any introduction at all.

As a general observation, the testers in the third iteration spent significantly less time to complete the task than the testers of the first and second iterations, the time spent going down from 1 - 1.5 hours to about 40 minutes. While one can not draw any conclusions from such a small sample size, this can be an indication that the interface revisions make the interface easier to use.

Chapter 7: Assessment through realizability study

This chapter discusses the realizability of the services specified using the proposed interface. It presents a service framework for the realization of the service derived from the specification modelled by the users. It also discusses how some of the core functionalities in a composed service can be realized.

7.1 Service realization

The following figure illustrates the main concepts related to the realization of a service composed by the user:

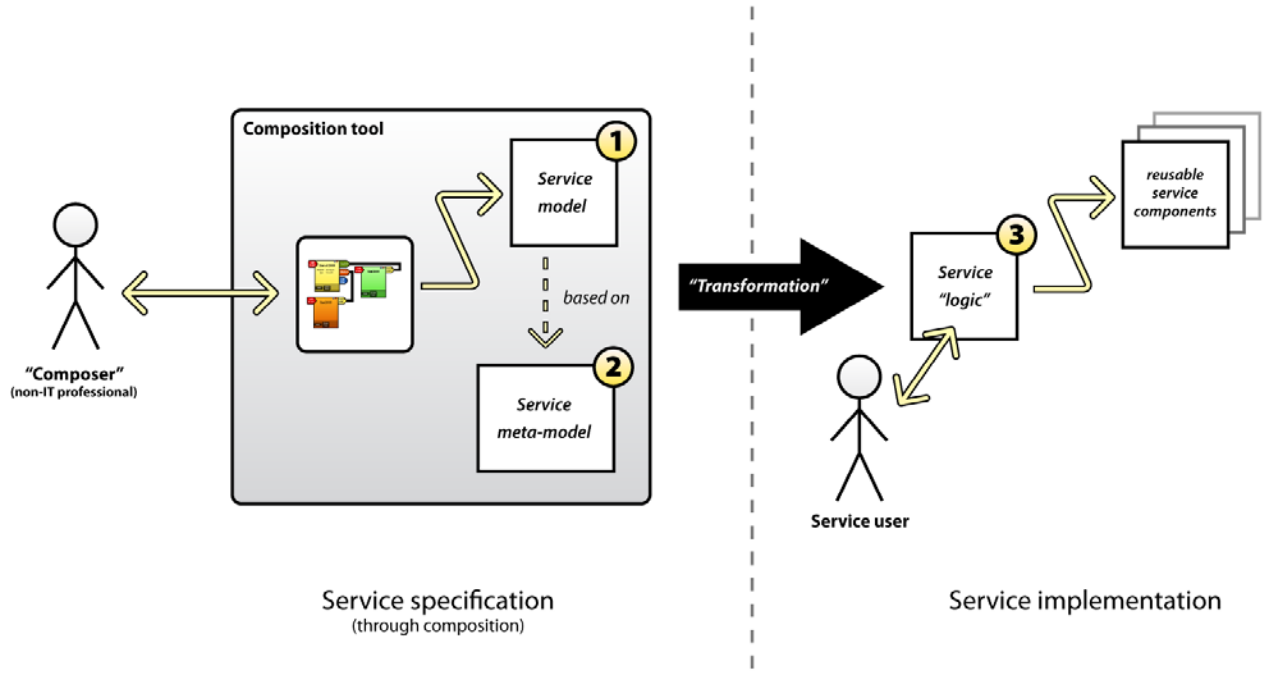


Figure 15: Realizability components

The main entities of this figure are:

1. **Service model:** A description of the service composed by the composer, i.e. the user of the composition tool
2. **Service meta-model:** A model defining the concepts needed to represent a service. The service model is defined using the concepts of the meta-model, as described in section 7.1.1.
3. **Service logic:** The code that realizes the composition defined by the composer. It interacts with a set of reusable service components that provide reusable behavior in the service. The service logic and the reusable components thus make up the service implementation.

7.1.1 Service meta-model

The following figure shows the service meta-model, represented as a UML class diagram, used to model concepts and their relationships. It shows how to conceptually represent the service components, with the classes necessary to represent the various services:

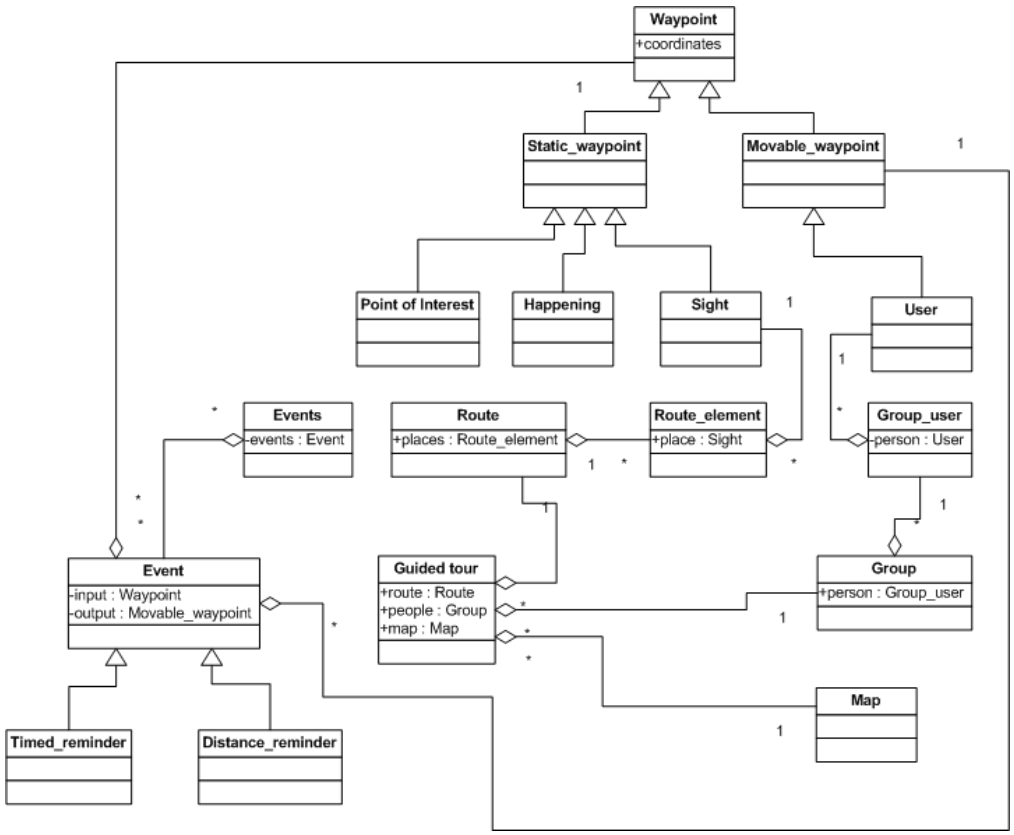


Figure 16: Meta-model showing classes making up the service. The composition uses this meta-model for representing a model for the service composition specified by the user.

The main entities of this figure are:

Waypoint: This is the basic class for any object that has a location or position associated with it as coordinates. These are further divided into **Static** waypoints, which are things that generally do not move, like churches and restaurants, and **Movable** waypoints, which need constantly updated positions; this will mostly apply to persons but can in theory also apply to objects like buses and taxis in an extended system. In this case, the defined static waypoints are **Point of Interest**, which can be stores, restaurants etc, **Happenings**, which is any time limited event happening in the city like an exhibition, and **Sights**, which are general tourist sights.

Group: A group consists of **Group_user** objects, which contain a **User** and some properties on how that user relates to the specific group.

Route: A route consists of **Route_element** objects, which contain a **Sight** and some properties on how that sight relates to the specific route. As an extension, Route_element objects should contain any static waypoints, not just sights.

Guided tour: A guided tour has a **Route** object, a **Group** object and a **Map** object.

Events: Event objects take any **Waypoint** as input and outputs messages to any **Movable_waypoint**, for instance an input can be a restaurant and the output can be a person getting a message about being near that restaurant.

7.2 Service logic

The service logic implements the core service composition behavior, essentially serving as the "client software" on the mobile device.

There are two main approaches for realizing the service logic: 1. using an interpreter that interprets the service model, 2. using a precompiled program. In the latter case, the "transformation" consists of generating the program from the service model. In the first case, no transformation - or only a minor transformation to the service model - is needed, as the local "virtual machine" on the device will interpret it directly.

There are a number of criteria to consider when deciding on the realization approach:

- **Configurability at runtime:** A precompiled program would be close to impossible to change at runtime on the mobile device. However, when using the interpreter approach, it is possible to let the service users themselves change parts of the service model directly on the mobile device. This could be exploited in future versions of this architecture.

- **Performance**: Because it is created and compiled after the service composition phase, and then loaded onto the mobile device, a precompiled program should be faster than an interpreted. In addition, the footprint should be smaller, because the generic interpreter program needs to take into account all the functionality that might be executed by the service core, while a precompiled program only has to implement the needed functionality as is.
  This performance advantage will be benefitial for slower and smaller devices, and even with the increased space and processing capabilities of new mobile phones it could make a noticeable practical difference.

- **Error handling:** A precompiled program should have a lesser potential for errors since it is not user modifiable. This in turn is also a drawback of the interpreter approach, where the added flexibility will create more potential for unforeseen errors in the service logic.

- **Compatibility:** The interpreter approach could make it easier to support a wider variety of devices, since the service model will be platform independent. However, there is still the need

for a generic interpreter program that needs to be adapted to the different platforms, which would be the same process a precompiled program would go through.

For this system, the “local interpreter” way of designing the software seems like the best choice. The added flexibility and ease of adding new versions can be assumed to make up for the drawbacks. Performance should not be a big issue with this relatively simple type of software, and modern mobile devices have ample space available, but the added flexibility would introduce a higher possibility of errors in the program, which will require a more thorough approach during testing.

7.3 Service framework architecture

The following figure shows the general architecture of the service implementation framework including a set of reusable service components:

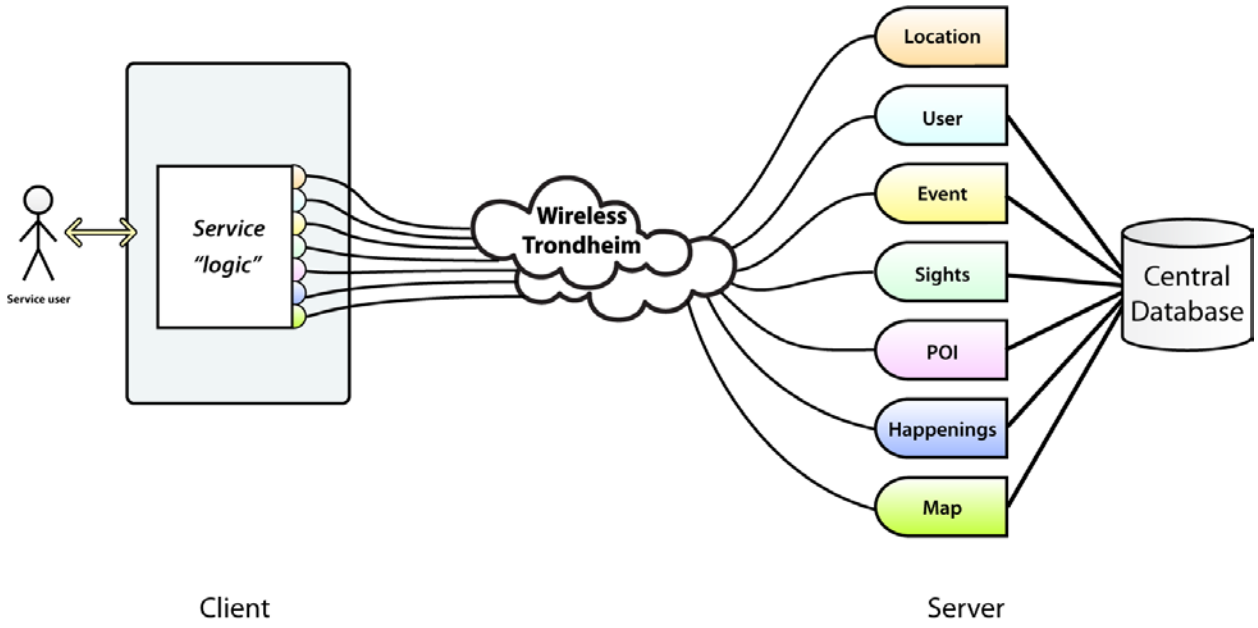


Figure 17: Architecture with reusable service components

The reusable service components shown in this architecture are:

Location: Enables one to get the coordinates of a user, generally through WiFi positioning, but secondarily through GPS if outside the coverage area of Wireless Trondheim.

User: Manages user and group functionality, including logging in, intra group messaging and who you are able to see on a map.

Event: Manages all events and the data flow between components and services connected by events.

Sights: Contains a list of tourist attractions and sights with associated data stored in a database, this data can include pictures, textual information and audio guides.

POI (Points of Interest): General points of interest, like stores and restaurants, with videos, user reviews etc.

Happenings: Manages all current events in the city, like concerts, theatre performances and exhibitions.

Map: Is responsible for storing and sending map tiles requested by the service logic.

Because the solution is intended to run inside the zone of the Wireless Trondheim WiFi network, we have posited that a lot of the services will be external to the local device. The increased bandwidth afforded by a WiFi network should fulfill the additional requirements this puts on the communication link, and the local system can then be kept as simple as possible.

7.4 Validation

For validating the service created by the composition interface, we look at a number of core functions of the prototype and show how they would result in interaction between the different building blocks of a completed solution. We can then find whether the data gotten from the composition phase is sufficient in creating the entire solution for the device, and if the components chosen as building blocks will be the right ones.

The basic set of operations we have identified to help validate the service are:

1) Logging on and registration to a group.
2) Seeing your friends on a map.
3) Going to the next point of interest.

For all of these, we see the user having a “main menu” screen which lets her do a variety of actions as determined by the composition.

1) Logging on

1. User turns on the mobile device and starts the guiding program.
2. User is met with login-screen and enters username and password.
3. Username and password is sent to the “User” module.
4. User module authenticates the user and stores the device ID, then checks to see if the user is assigned to a group.

5. If user belongs to a group, the user module sends the other group members the user's new online status.
6. User module checks with the "Event" module if any events are assigned to the user.
7. User module sends all the data including all online friend IDs from the assigned group back to the user. (User module stores the device ID and uses it as an identifier during all further communication from the mobile device to the external services.)
8. The user module will begin sending periodic messages to the "Location" module to learn the position of the user as soon as she is on the Wireless Trondheim network (through WiFi positioning,) and will continue keeping that position updated as long as the user is logged on.
9. After the user is logged in/authenticated, she will have access to the full menu of the program, with the options available depending on the composition.

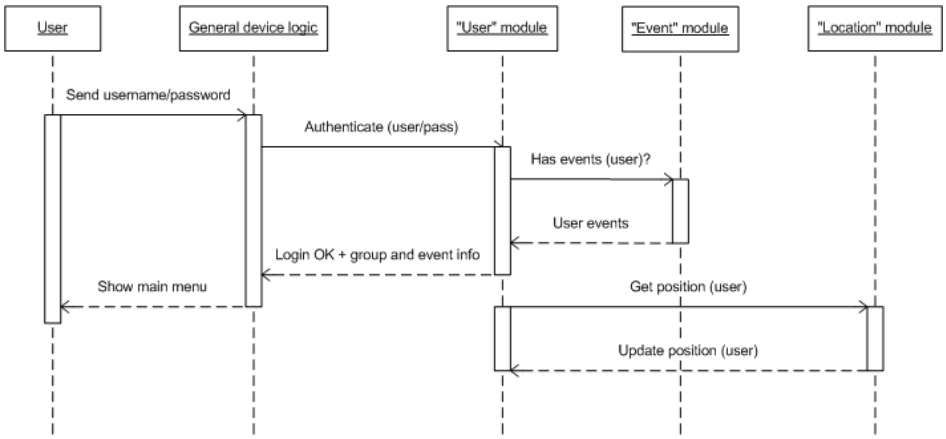


Figure 18: Sequence diagram for operation 1, logging on

2) Seeing your friends on a map

1. User selects the "map" function in the main menu.
2. The device sends a request to the "User" module to get the positions of the user and all of the friends that are set to be visible on the user's map.
3. The device then sends a request to the "Map" module with the position.
4. The map module sends applicable map tiles based on the user's current position back to the user.
5. The map tiles are shown and the user and friends are drawn on as dots on the map locally.
6. Periodically the device will send a request to the user module to get updated positions for the map, to see everyone moving around.
7. The program will automatically request new map tiles from the map module when the user is getting close to the edge of the map, or the user is moving the map around or zooming in/out.

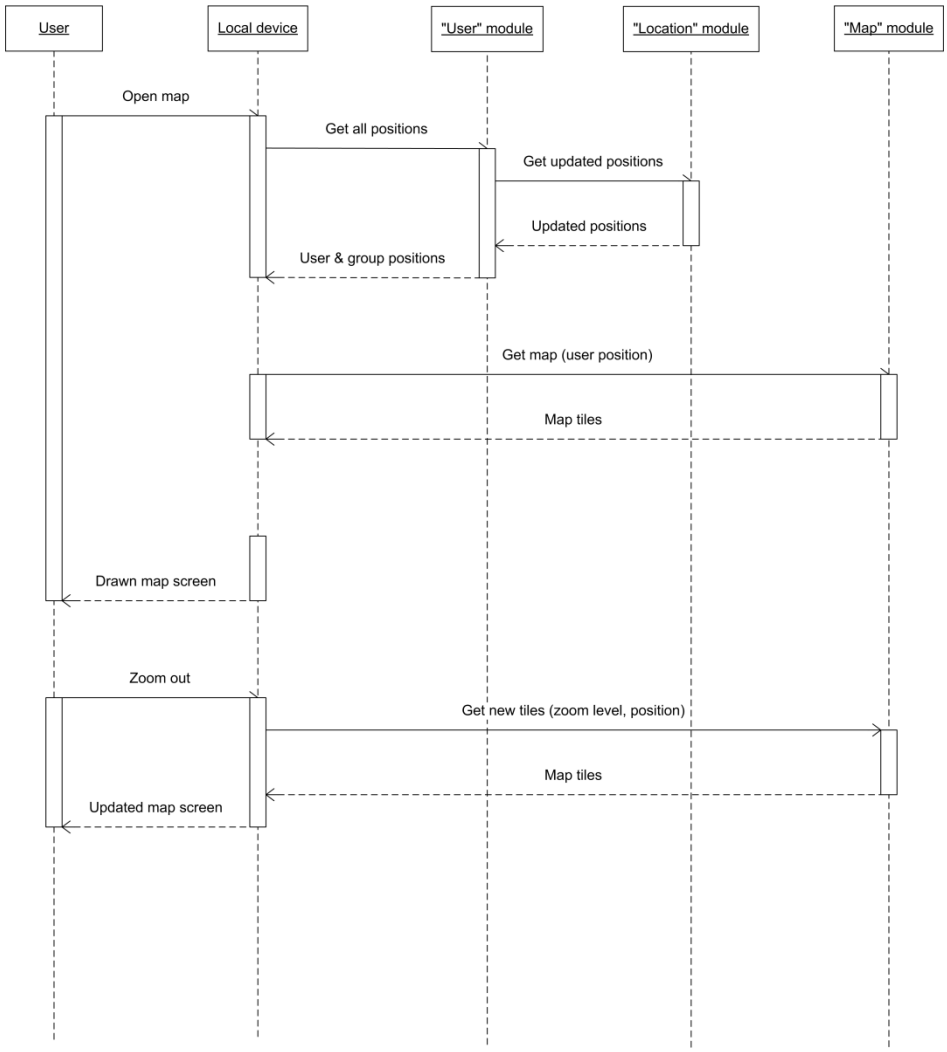


Figure 19: Sequence diagram for operation 2, seeing your friends on a map

3) Going to the next place in a route

1. If the user is on a “free walk”, she has the choice of seeing all unvisited places, or to go to the closest one. In this example, she wants to go to the closest one.
2. She opens the route viewer. The local “Tour” module contains a list of all places to visit, and it knows where the user has already been. The user gets a list of the places in the route, with the unvisited ones at the top, sorted by distance.
3. She clicks the top one to select it (for instance “Nidaros Cathedral”.) This activates new buttons related to the choice: “Go there”, “Show on map” and “View information”.
4. She clicks “Go there” to get guided directions on how to walk there.
5. The tour module knows the place to go to, but not he directions there. It sends a request to the “Map” module for directions from the user’s current position to the sight, and the map module returns a set of directions to the tour module.

6. The device uses the directions to guide the user to the place, showing the route as a line on the map.
7. When she arrives, the screen on her device changes to show information about the place, based on the types of information available.
8. The user can then click on the different types of information available, for instance text information, audio guiding, videos, user reviews, photographs and more. The user selects to listen to an audio guide about the sight.

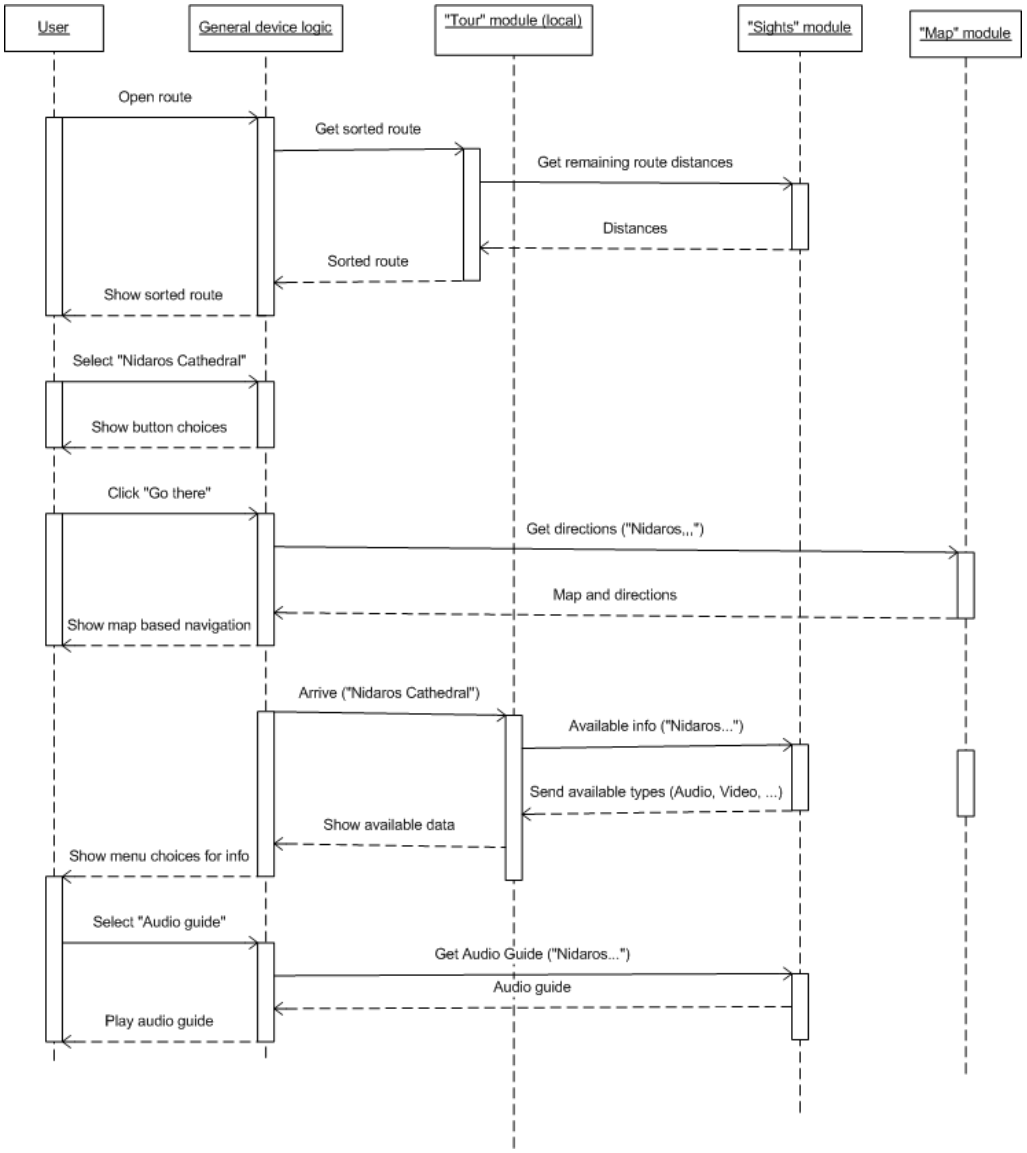


Figure 20: Sequence diagram for operation 3, going to the next place in a route

Chapter 8: Conclusion and further work

This chapter summarizes the results achived in this thesis. It also discusses the work approach and suggests further directions of work.

8.1 Achievements

8.1.1 Results

Two main artifacts have been created for this thesis, a user interface for a composition tool and a service framework.

The user interface was successful in helping users create new services related to the created scenarios. Paper prototype evaluations showed that the interface was generally easy to understand and use, as most users were able to use the interface to create their services. The testing also uncovered some usability issues with the tool, which showed some possible areas of improvement.

Further, the results of the composition were successfully incorporated into an architecture. We specified how the results from the composition would be transformed into a service logic that could run on a mobile device, and we showed how this service logic would interact with the identified building blocks.

8.1.2 Research questions

The first research question was: “How can we design end-user tools that enable specification of services related to the «city guide» scenario?”
The proposed interface has been shown to be a valid design for an end-user tool, and users were indeed able to specify services related to the city guide scenario.

The second research question was: “Can services that are specified by the user be realized in practice?” We have shown that the service model resulting from the composition can be transformed into a runable service logic by using the proposed meta-model and a transformation process that lets a local interpreter on a mobile device present a complete service to the user.

8.1.3 Fulfillment of the requirements

Based on the work done in the previous chapters, we can determine how the requirements were met.

Requirements **1** – **5** were general functions of the composition tool that were shown to be fulfilled by the paper prototype assessments: Composers were able to (**1**) create guided tours, (**2**) organize persons in groups , (**3**) specify the tour to be a “free walk”, (**4**) specify languages and (**5**) organize sights in routes.

Requirement **6**, that the interface must be based on visual programming principles, was fulfilled.

Requirement **7**, specifying that users must be able to add and define new types of elements was not tested. Still, the functionality was kept in mind while designing the user interface and incorporated into the design, by creating the generic “Event” object, and allowing for the specification and addition of new components in the interface.

Requirement **8**, a non-functional requirement of usability was shown to generally be fulfilled through the paper prototype assessments.

Requirement **9**, a non-functional requirement of extensibility was not tested but incorporated into the design.

8.2 Discussion

This work has served as an initial investigation into user interfaces for composition tools in the UbiCompForAll project and has helped give an initial understanding of which properties these interfaces require to fulfill requirements of usability and extensability.

In addition, the proposed architecture has been a first step towards realizing these services in practice. It can be used as a starting point for future investigation into these areas, when more complex systems are being created for more extensive testing.

Because the UbiCompForAll project is still in its initial stages, this thesis is one of the first attempts within the project at assessing a composition tool and the related components. As the project moves forward, similar studies will have to be done for composition tools related to other scenarios and areas, and hopefully this thesis will be a valuable reference for this further research.

This has also been a first foray into paper prototype assessments for the UbiCompForAll project. The experience gained from these assessments will be important, as the testing and development of user interfaces for a variety of different scenarios is a significant part of UbiCompForAll.

Throughout this process we have learned a great deal about composition tools, user interfaces and paper prototyping. Based on the things we know now, some things could have been done differently, for instance, the paper prototype assessment was more time consuming than previously assumed, and should have taken place earlier. This would have also given more time to further develop the service framework and look into realizing some service components. However, we still feel we have had valuable results from this thesis work.

8.3 Further work

Further work based on this thesis can be focused on realizing the service framework in practice and to further test the proposed architecture. In addition, a number of extensions to the user interface could be the basis of further research.

The first step could be to begin building an actual composition tool and the associated components. Further, the transformation from the user service specification to the executable service logic could be realized.

To help test this framework, some of the identified building blocks should also be realized and prototype software for mobile devices should be created, for instance using the Android platform.

The next step would be to introduce extensions to the composition tool. Since a number of ways to improve and extend the functionality of the interface of the service composition tool were discovered, these can serve as a basis for the implementation into the composition tool. Specifically, these items could be further specified and be incorporated into the tool:

- Information abstraction, having different user interface complexity for different user roles
- Giving users a way to create and edit routes visually using a map
- Letting users see the result of the composition “live” in an emulator

Finally, one can look into extending the service functionality itself, by identifying and introducing more reusable service components.

Appendix A: References

[Andresen 2007] Andresen, S.H., J. Krogstie, and T. Jelle; *Lab and Research Activities at Wireless Trondheim, in Proceedings of the 4.th IEEE International Symposium on Wireless Communication Systems* (ISWCS'07) M. Pätzold, Y. Jiang, and Y. Zhang, Editors. 2007, IEEE: Trondheim, Norway. p. 385-389.

[Benatallah 2005] Benatallah, Boualem and Dijkman, Remco and Dumas, Marlon and Maamar, Zakaria; *Service Composition: Concepts, Techniques, Tools and Trends*. In: Stojanovic, Zoran and Dahanayake, Ajantha, (eds.) Service-Oriented Software System Engineering: Challenges and Practices, 2005, IDEA Group, Hershey, PA, pp. 48-66. http://eprints.eemcs.utwente.nl/7048/ accessed June 18, 2009

[Chakraborty 2005] Chakraborty, D., Joshi, A., Finin, T., and Yesha,Y.; *Service composition for mobile environment*. Mobile Networks and Applications 4, 10 (August 2005), p. 435–451. http://ebiquity.umbc.edu/paper/html/id/175/Service-Composition-for-Mobile-Environments, accessed June 18, 2009

[Cheverst 2000] Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C.; *Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences*. Proceedings of CHI 2000, Netherlands, April 2000, pp 17-24. http://www.guide.lancs.ac.uk/CHIpaper.pdf, accessed June 18, 2009.

[Hevner 2004] Hevner, March and Jinsoo; *Design Science in Information Systems Research*, MIS Quarterly Vol. 28 2004, pp. 75-105.

[ICT 2009] ICT Work Programme 2009-10 http://cordis.europa.eu/fp7/ict/ , accessed June 18, 2009.

[Johnston 2004] Johnston, W., Hanna, J.R., Millar, R.; *Advances in Dataflow Programming Languages,* ACM Computing Surveys, Vol. 36, No. 1, March 2004, pp. 1–34. http://www.ittc.ku.edu/~rsass/rcreading/johnston04.pdf, accessed June 18, 2009.

[Kenteris 2007] Kenteris, M., Gavalas, D., Economou, D. (2007); *An innovative mobile electronic tourist guide application*. Personal and Ubiquitous Computing, Vol. 13, No. 2, February 2009, p. 103-118. http://www.springerlink.com/content/887gj235177w7781/, accessed June 18, 2009.

[Myers 2004] Myers, B. A., Pane, J. F., Ko, A.; *Natural programming languages and environments.* Communications of the ACM, September 2004, Vol. 47, No. 9. http://portal.acm.org/citation.cfm?doid=1015864.1015888, accessed June 18, 2009.

[Tersus 2009] Tersus Software Ltd. http://www.tersus.com, accessed May 31 2009.

[Snyder 2003] Snyder, C.; *Paper Prototyping – The fast and easy way to design and refine user interfaces*. Morgan Kaufmann Publishers, London, 2003.

[Stav 2006] Stav, E.; *An Approach to Developing Extensible Application Composition Environments for End Users*. Doctoral thesis, Norwegian University of Science and Technology, 2006

[UbiCompForAll 2009] The UbiCompForAll research project, *UbiCompForAll - Ubiquitous service composition for all users*, http://www.ubicompforall.org, accessed June 17, 2009.

[Weiser 1991] Weiser, M.; *The Computer for the 21st Century,* Scientific American vol. 264 issue 3, 1991, pp 94 – 104.