Aalto University
School of Science
Degree Programme in Information Networks

Katriina Heiskanen

# Parametric Modelling in Website Design: A Solution for Efficient Requirements Specification and Fast Development

Master's Thesis

Helsinki, 30th May, 2018

Supervisor: Professor Marko Nieminen

Thesis advisor: Katri Jauhiainen, M.Sc. (Tech)

Aalto University
School of Science

|  |  |  |
|---|---|---|
| **Author** | Katriina Heiskanen | |
| **Title of thesis** | Parametric Modelling in Website Design: A Solution for Efficient Requirements Specification and Fast Development | |
| **Master's programme** | Information Networks | |
| **Thesis supervisor** | Professor Marko Nieminen | |
| **Major/Code** | Information Networks/SCI3047 | |
| **Department** | Computer Science | |
| **Thesis advisor** | Katri Jauhiainen, M.Sc. (Tech) | |
| **Date**<br>30th May, 2018 | **Number of pages**<br>vi + 58 | **Language**<br>English |

## Abstract

This thesis addresses two challenges in website design and software development in general. First, development of websites is slow and requires special knowledge. Second, successful requirements engineering is essential for success of the project, but specifying the requirements is often time-consuming. Parametric modelling has long been utilized in mechanical engineering, architecture and other 3D modelling, but not much in web design. This thesis brings the parametric modelling to the context of web development to find a new solution to support efficient requirements engineering and fast development of websites in the context of the case company.

This thesis studies possible ways of describing websites as parametric models and parametrizing website requirements, and the role of a parametric model of a website in supporting the requirements engineering process. Data about the requirements engineering process and the website requirements was collected with interviews and project document analysis. Based on the research results and literature, a parametric model of a website was developed.

The research results and the parametric model created show, that parametric modelling is a great tool also for website design. Parametric model enables fast prototyping of the website. Prototypes in requirements engineering facilitate better communication and mutual understanding, which are crucial for successful requirements engineering. The parametric model also structures the requirements engineering process and supports iterative specification of requirements. Parametric model simplifies the website development, and changes to the website can be made fast to meet the customer's changing requirements.

**Keywords** Parametric modeling, website design, requirements specification

Aalto-yliopisto
Perustieteiden
korkeakoulu

| | | |
|---|---|---|
| **Tekijä** | Katriina Heiskanen | |
| **Työn nimi** | Parametrinen mallintaminen nettisivujen suunnittelussa: Ratkaisu tehokkaaseen vaatimusmäärittelyyn ja nopeaan kehittämiseen | |
| **Koulutusohjelma** | Informaatioverkostot | |
| **Työn valvoja** | Professori Marko Nieminen | |
| **Pääaine/Koodi** | Informaatioverkostot/SCI3047 | |
| **Laitos** | Tietotekniikka | |
| **Työn ohjaaja** | Katri Jauhiainen, DI | |
| **Päivämäärä** | **Sivumäärä** | **Kieli** |
| 30.5.2018 | vi + 58 | englanti |

**Tiivistelmä**

Tämä diplomityö keskittyy kahteen verkkosivustojen suunnittelun ja ohjelmistokehityksen haasteeseen. Ensiksi, verkkosivujen kehittäminen on hidasta ja vaatii erityisosaamista. Toiseksi, onnistunut vaatimusmäärittely on perusehto koko projektin onnistumiselle, mutta on usein aikaa vievää. Parametrista mallintamista on pitkään hyödynnetty konesuunnittelussa, arkkitehtuurissa ja muussa 3D-mallintamisessa, mutta ei juurikaan web-suunnittelussa. Tämä työ soveltaa parametrista mallinnusta web-kehityksessä sekä pyrkii löytämään ratkaisun, joka tukee tehokasta vaatimusmäärittelyä ja nopeaa verkkosivujen toteuttamista kohdeyrityksen kontekstissa.

Tämä työ tutkii mahdollisia tapoja kuvata verkkosivustoja parametrisina malleina, verkkosivuston vaatimuksien parametrisointia sekä verkkosivuston parametrisen mallin roolia vaatimusmäärittelyprosessissa. Aineistoa vaatimusmäärittelyprosessista ja verkkosivustojen vaatimuksista kerättiin haastatteluiden avulla ja projektidokumentteja tutkimalla. Parametrinen malli verkkosivustosta kehitettiin tutkimustuloksiin ja kirjallisuuteen pohjautuen.

Tutkimustulokset sekä toteutettu parametrinen malli osoittavat, että parametrinen mallintaminen on oiva työkalu myös verkkosivustojen suunnittelussa. Parametrinen malli mahdollistaa verkkosivuston nopean prototypoinnin, mikä tukee parempaa kommunikointia ja yhteisymmärrystä vaatimusmäärittelyssä, joka on erityisen tärkeää onnistuneen vaatimusmäärittelyn kannalta. Parametrinen malli myös jäsentää vaatimusmäärittelyprosessia ja tukee iteratiivistä vaatimusmäärittelyä. Parametrinen malli yksinkertaistaa verkkosivustojen toteutusta, ja mahdollistaa muutosten tekemisen nopeasti asiakkaiden vaatimusten muuttuessa.

**Asiasanat** Parametrinen mallinnus, verkkosivuston suunnittelu, vaatimusmäärittely

# Acknowledgements

So, here it is! The last, the hardest and the most educational project of my university studies: Master's thesis. This is something I could never have achieved alone. So many people have helped me during this journey, and you all deserve the greatest thanks.

First, I would like to thank Kaj Forsström and my thesis advisor Katri Jauhiainen for giving me the possibility of doing this thesis project as a part of my work at Mascus. Without this possibility, my journey towards graduation would have been much more painful. I would also like to thank all the salespeople, who bothered to participate in the interviews during the busy SF times and gave me important data for the project. Also, thank you all at the Helsinki office for the horrible sense of humour that made me laugh every day at work.

Also, I am extremely grateful for the invaluable guidance and expertise provided by my thesis supervisor, professor Marko Nieminen. Without all that help, I could never have finished this project in five months, if ever.

Furthermore, I would like to thank everyone, who has been there supporting and listening, when I felt like I could never finish this project. Thank you, friends and rubber ducks. You are invaluable.

Last but not least, thank you, Mom and Dad, for always being there and believing in me. And for the ice creams of course.

Espoo, 30th May 2018

Katriina Heiskanen

# Table of Contents

# 1 Introduction

Web development technologies are evolving at high speed. During the recent years, websites have developed fast in many ways. New JavaScript libraries and HTML5 innovations have made the websites more interactive, the high increase of mobile devices has made them easily available, and disappearing of Flash, increased legislation and incognito browsing, among other things, have improved the privacy and security (Laperdrix *et al.*, 2016). This leads to the fact that websites built at the beginning of the decade are already technologically outdated. With the increasing importance of the web presence for succeeding in any business, the demand for website and web application providers is high.

Developing websites with the modern technologies requires special knowledge. Website builders, content management systems, models and frameworks have been created to make the website development process easier and faster, but the demand for new ways to make websites is still high as the technologies keep developing. This thesis aims to create a suggestion of a new model, which would lower the level of required resources to develop websites for customers in the context of the case company.

Another challenge in website development is succeeding in defining the requirements. Requirements engineering is a crucial part of any kind of software development. The success of requirements engineering process depends heavily on the success of the communication between application developers and other stakeholders (Saiedian and Dale, 2000). According to Saiedian and Dale (2000), a common problem in communication is the knowledge gap between the customer, the salespeople and the developers. Thus, this thesis aims also to provide a possible solution to facilitate better communication about the requirements and make the requirements engineering work more efficient.

## 1.1 Background and motivation

The case company is specialized in providing e-commerce solutions for heavy used equipment. In addition to their online marketplace, they provide solutions from fully customized integrated management systems and mobile applications to inspection applications and Facebook web shops. One of the solutions is a web shop, that is integrated into the customer's existing website. The web shop solution includes detailed search, product cards with contact form, and frontpage items like custom category links and featured products.

Often customers would need a full website from the case company instead of just the integrated web shop. They might have an outdated website that is not even responsive, or no website at all. Currently, the case company does not have a feasible solution for building full websites and responding to the increasing demand. Recently, some web sites have been built from the scratch according to the design guidelines provided by the customer, but it has been slow and ineffective due to the lack of proper development guidelines and methods. The case company has also ordered some websites from subcontractor, but in many cases the customer has ordered the website from other provider and only the web shop solution from the case company.

The inability to provide full websites built quickly by own developers is seen as lost profit for the company. Either another company gets the profits from the project, or the profits are low due to the ineffective website building process. The case company has chosen to use content management system provided by Voog[1] as a part of a new website building solution. The new solution is going to be using Voog API[2] for creating websites.

The first requirement for the new solution is, that creating full websites for web shops does not differ from creating just the web shop for an existing website. The components of the web shop are built as .NET controls that have changing parameter values defined in an XML file. For a basic web shop solution, the developer only needs to create an XML configuration file for the solution and define the parameter values for the components used in the solution. Usually the solutions require also custom

[1] Voog https://www.voog.com/ (Accessed on 27th May, 2018)

[2] Voog API https://www.voog.com/developers/api (Accessed on 27th May, 2018)

styling in CSS. Thus, also the new website solution needs to be built so that the customisable parts of the site are defined as parameters in the XML configuration file.

Other requirement for the solution is that the developers need to be able to build websites that fulfil the customers' requirements with as little customisation as possible. This means that websites meeting these requirements could be generated only by changing parameter values. Also, the new solution should be able to support sales process by making the requirements gathering easier for the sales people and by simplifying the whole development process.

## 1.2 Research problem and scope of the study

The goal of this thesis is to produce a proposal of the new solution for developing full websites for the case company's customers. As described in the previous section, the solution has several restrictive requirements. The website layout needs to be divided into parts, that can be built as separate .NET controls. All the editable properties of these parts should be defined as key-value pairs in an XML file. The new solution should be based on the general requirements for the websites, that remain the same in all the website projects. Also, it should adjust according to the case specific requirements.

Parametric design is based on the idea, that the designed objects consist of certain unchanging frames, and other characteristics, that chan ge inside these frames (ten Teije *et al.*, 2004). Parametric design has long been utilized in mechanical engineering, architecture and other 3D modelling, but not much in web design. Many models, frameworks and description languages for user interfaces and web applications are based on similar design thinking as parametric design but have a different approach to the problem. Based on this, the research problem of this thesis is:

*How can parametric design be utilized in website design?*

The website design process in this case consists of two different phases: 1) finding and defining the customer's own requirements for the website and 2) building the website according to these requirements. The research questions aim to find support for these phases, and to bind these phases together. This research has three research questions that are divided into sub-questions.

*RQ1: How can a website be described as a parametric model?*

The first research question considers the basic characteristics of a parametric model for any website. In the literature review, basics of parametric design and parametric models are explored, as well as modelling of web applications and user interfaces as descriptive models. Parametric modelling of a website is based on the review of these two subjects in this thesis.

*RQ2: How can website requirements be parametrized?*

The second research question considers the relationship between the parametric model of the website, and the requirements of the website. Customers, that the websites are built for, have also their own requirements for the website. Based on the empirical material, the similarities and differences of these requirements are analysed. Based on the analysis and the answer to the research question one, the customer requirements are parametrized and merged in to the generic parametric model of a website.

*RQ3: How can a parametric model of the website support finding and defining the customer specific requirements?*

As mentioned previously, one goal for the new solution would be additional support for the sales people, who are in charge of the communication with the customer and gathering and defining the customer's requirements. Based on the conceptual background gathered from literature and empirical research, the role of the parametric model in requirements engineering is defined.

## 1.3 Research process

The research process of this study started by defining the constraints and requirements of the case project. Based on this definition, a preliminary research problem was defined. To create a base of knowledge for the research, literature about parametric design and web development methods and models was reviewed. Combining knowledge about the literature and the case project, research problem and research questions were formulated. The final version of these was presented in the previous chapter.

When the research problem was defined, conceptual background about the subject was collected. Conceptual background includes previous research about requirements engineering in the web context, parametric design and modelling of web applications. Conceptual background is presented in chapter 2. Research questions were again refined at this stage.

After collecting the conceptual background, four people were interviewed to collect empirical data. The analysis of the data revealed that data collected with interviews is not sufficient to answer the research questions, so project document analysis was chosen to be an additional research method in the study. The research methods and data are described in chapter 3.

Results were collected from both interviews and project document analysis. The results of the research are presented in chapter 4. Based on the conceptual background and empirical results, a parametric model of a website is created. The parametric model, answers to research questions and conclusions are presented in chapter 5.

# 2 Conceptual background

This chapter provides an overview of previous research in requirements engineering of web applications, parametric design and modelling of web applications. The purpose of the conceptual background is to provide knowledge of the subject to conduct the empirical research, but also to evaluate the results and answer the research questions together with the empirical results. The first chapter comprises literature about requirements engineering of web applications is reviewed. The second chapter comprises literature about parametric design and modelling of web applications.

## 2.1 Requirements engineering of web applications

Requirements engineering is a crucial part of any kind of software development. In 'IEEE Standard Glossary of Software Engineering Terminology' (1990), a requirement is defined as "a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents". Well-specified requirements describe for developers what they should build and customers what to expect from the product, and also work as a base for validating if the system meets the needs of the customer (Escalona and Koch, 2004). Requirements engineering is an iterative process of finding, analysing, documenting and validating these requirements (Sommerville, 2010). The process is both iterative and co-operative, aiming to analyse the problem, document the results and evaluating the quality of the results produced (Ferreira and Loucopoulos, 2001).

To be able to build a software product, development team needs to gather knowledge about the problem domain and the requirements of the application (Escalona and Koch, 2004). Many practicalities and techniques have been developed to support this, but it is the responsibility of the development team to find the best way for each project, although the success also depends on the other stakeholders participating in the process (ibid.).

Generally, the requirements engineering process consists of three phases: requirements elicitation, requirements specification and requirements validation (Lowe and Hall, 1999). The iterative process is described in Figure 1. Different interpretations of the requirements engineering process have been presented. For example, (Sawyer and Kontonya, 2001) have included the requirements analysis and negotiation as a fourth activity. In this model, it is part of the requirements specification activity.

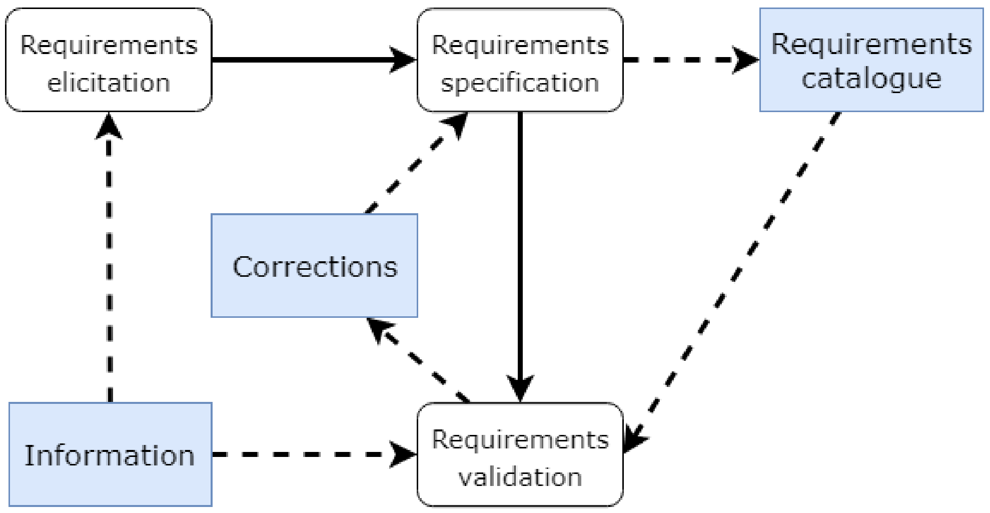

*Figure 1 Iterative process of requirements engineering (Escalona and Koch, 2004)*

The requirements engineering process described by Escalona and Koch (2004) starts with the requirements elicitation. In elicitation phase, information about the users and customers is collected from different sources, like documents and interviews. After that the requirements are defined based on the collected information in requirements specification phase.  In the last phase the requirements specifications are validated against the collected information to ensure that no inconsistencies, mistakes or undefined requirements exist. The process of specification and validation of the requirements is iterative, and it might take several iterations to get the final requirements.

Escalona and Koch (2004) point that the development of web applications differs from the development of other software systems. They describe that the main features of web applications are the navigational structure, the user interface and the personalization capability, while other characteristics of software applications are less significant. Nevertheless, different kinds of stakeholders are required to participate in the development process. Also, many developers of web applications, like web designers and analysts, do not have background in software engineering (Bolchini and Paolini, 2002).

Some characteristics of web applications require special attention. Users need to be able to navigate in the application intuitively (Olsina, 1999, referenced by Escalona and Koch, 2004), as interactivity is often in key role in web applications (Bolchini and Paolini, 2002). Also, multimedia and marketing aspects and high-level communication and business goals have to be taken into account in the requirements specification of web applications (Escalona and Koch, 2004; Bolchini and Paolini, 2002). These aspects have to be taken into account already in the requirements engineering process, not only in the design phase (Escalona *et al.*, 2002).

Inadequate requirement specification can lead to inability to meet the customers' needs and poor user experience (Brinck *et al.*, 2002). Bolchini and Paolini (2002) point that to succeed in the market, web applications need to be stakeholder-centred. They tell that the content and interaction in web applications need to be designed to meet the goals of the users as well as possible, but also the business and communication objectives have to be satisfied.

Escalona and Koch (2004) divide requirements of web applications into two categories: functional and non-functional requirements. Functional requirements describe the capabilities that the web application system should have. They are sub-classified to many different categories. Data requirements establish how the application stores and administrates information. Interaction requirements define how the user would interact with the application. Navigational requirements describe how the users need to be able to navigate in the system. Personalization requirements describe how the system needs to adapt to the different environments and according to the different user needs. Transactional requirements describe the requirements for the internal computations that do not consider interface or interaction aspects. In turn, non-functional requirements work as constraints for the solution. They describe general characteristics of the web application instead of the functionalities, like portability, reuse possibilities, usability, availability and performance of the web application.

### 2.1.1 Communication between stakeholders

According to Saiedian and Dale (2000), the success of requirements engineering process depends heavily on the success of the communication between application developers and other stakeholders. As described in the previous section, all the information to be used in specifying the requirements is collected in requirements elicitation phase. Requirements elicitation is also the most communication-intensive phase of the requirements engineering process (Saiedian and Dale, 2000). Thus, successful communication between the stakeholders is most important for the success of the whole project in the elicitation phase to build a solid base (ibid.).

Saiedian and Dale (2000) describe why successful communication of all the restrictions of the project is important for effective information gathering in requirements elicitation. Ensuring that the customer understands what kind of product is possible to be delivered with certain resources is key to defining such requirements that can be met. Otherwise the defined requirements or even the delivered product is likely to fail to meet the expectations of the customer. Very often the customers would expect things that cannot be provided, making it important to clearly communicate the need for compromises.

According to Saiedian and Dale (2000), communication can fail even if the right things are communicated but in a wrong way. They describe that all verbal and written communication need to be consistent and unambiguous. Desire to win the tender over competitors and overeagerness to be responsive can lead to unfulfilled customer expectations. Also, undervaluing the offered product and exceeding customers' expectations highly can lead to overly high expectations in the future or even create distrust.

Saiedian and Dale (2000) describe that often the customer and developer communicate through intermediaries. These indirect communication links often define the customers' goals and needs to designers and developers. One common example is marketing and sales link, in which salespeople act as intermediaries. Intermediaries often lack in understanding of the needs and constraints of the sides involved in communication, which can result in filtered or distorted messages. Different customers also have different communication preferences, which makes the situation even more complicated, as the intermediaries and developers need to adjust to that.

According to Saiedian and Dale (2000), a common problem in communication is the knowledge gap between different stakeholders. Often the customer does not have sufficient technical knowledge to fully understand all the aspects of the system that the developers are talking about. Thus, professionals need to adjust to the level of the customer's knowledge to ensure fluent communication. They still need to be careful not to underestimate the customer's technical skills.

Differences in problem perspective between the stakeholders challenges the communication, as Saiedian and Dale (2000) describe. Customers are not aware of all the technical constraints in software development. Thus, they do not know what they can ask for the product. On the contrary, developers are not aware of all the characteristics of the application domain that affect the development. They might be eager to develop new features that are useful and profitable to the company, but do not fully meet the needs of the customer.

Saiedian and Dale (2000) describe a set of skills that successful communication and information-gathering requires. Active participation (for example asking questions, re-stating and taking notes) in customer's conversation helps to focus attention to the right things. Customers are not often able to fully describe their needs and might feel stress when required to do so. To help the customer to articulate their requirements properly, they should be offered some representation of the product to facilitate the communication (Karten, 1994, p.66). These representation should work as a guide, and not make the customer feel like they are pressured to choose certain solution (Saiedian and Dale, 2000).

Karten (1994) describes that customers often mean different things that they say, so sceptic attitude towards the information gathering is important. Karten suggests different ways to ensure successful communication. Nothing should be assumed to be as it seems, and questions should be repeated and rephrased to find different perspectives. Also, if anything is unclear or seems inaccurate, asking for clarification is important. Customers might be telling what they think they should tell, and not their own opinion. To get a broader perspective and possible information gaps filled, information should be gathered from multiple sources.

### 2.1.2 Using prototypes in requirements communication

Johansson and Arvola (2007) point that prototypes work as structuring resources in communication between stakeholders. Prototypes in user interface and interaction design are commonly classified as high or low fidelity prototypes according to their level of fidelity (Römer *et al.*, 2001). The level of fidelity describes how well the prototype resembles the final product (Johansson and Arvola, 2007). Low fidelity prototypes usually have different kind of interaction, visual expression and level of detail than the final product (Walker *et al.*, 2002). High fidelity prototypes are more similar to the final product, have more realistic interaction, and communicate better the possibilities of the design (ibid.). Dynamic computer prototypes are examples of high fidelity prototypes that look and act like a finished product, and are used for detailed prototyping (Johansson and Arvola, 2007).

The level of fidelity of the prototype does not make a significant difference in the benefits achieved by prototyping, so researchers recommend to use the kind of prototype that best suits the situation (Johansson and Arvola, 2007; Walker *et al.*, 2002)). Walker *et al.* (2002) found that when using computer prototype instead of paper, users testing the prototype made significantly more comments, even though the amount of issues arisen did not change. That supports using high fidelity computer prototypes as a tool of communication in requirements engineering.

Rudd, Stern and Isensee (1996) state that high fidelity prototypes are great tools for marketing and sales. High fidelity prototypes provide a good basis for thorough evaluation of the product for the customers and can be used effectively to encourage customers to buy the product. Also, as changes to high fidelity prototypes can be made quickly, the customers can see fast how their wishes for the product can be implemented in the product.

Saiedian and Dale (2000) point that using visualizations of the product can help clarify and detail the understanding of the requirements. According to their research, prototyping has been identified to be a viable definition tool for user interface software products, as it helps in identification of real requirements and elimination of unnecessary requirements. Prototypes provide context which helps customers to better understand the system they want (Escalona and Koch, 2004). Andriole (1994) believes that prototyping is crucial for requirements specification. Schrage (2004) suggests that a prototype should be a main medium of communication between the development team and the customer. Käpyaho and Kauppinen (2015) state that prototyping helps in validating the requirements before implementation of the product, reducing the risk of the requirements being misunderstood.

Käpyaho and Kauppinen (2015) found out that prototyping ensures improved communication between stakeholders. Customers did not participate more in the project, but prototyping helped in reaching consensus by providing a common language for the communication. Prototypes help to ensure a good quality communication and to reach mutual understanding faster. Increasing the amount of communication is difficult, so reaching better quality is the most important way to improve communication. Other benefit of prototyping is that it works as a plan for the project that can be relied on when something is wanted to be changed. Prototypes also motivate stakeholders to focus on requirements work, as stakeholders need to discuss changes to requirements more concretely.

## 2.2 Parametric design and modelling of web applications

The basic idea of parametric design is dividing objects in to different parts, that consist of parameters and their values (ten Teije *et al.*, 2004). Many different ways of dividing user interfaces and web applications into parts that consist of different characteristics have been developed to make the design and development process easier. User interface description languages describe user interfaces with hierarchical classes and variables (Helms and Abrams, 2008). Declarative web application development means that a whole web application is developed with a declarative framework (Vuorimaa *et al.*, 2016). For example, only XML can be used to build the whole application, which means that the application needs to be divided to parts with different values. This kind of solutions also lower the required skill level of the developer.

### 2.2.1 Parametric design

Ten Teije *et al.* (2004) define parametric design as "a method for designing objects which is a simplification of general configuration". Like other design tasks, the requirements for the design are taken as an input, and a design satisfying these requirements is produced as an output. The objects that are configured in parametric design all need to have the same overall structure, that is defined in a form of preconfigured template. Variations of the configured objects can only be attained by defining the values for parameters within the template.

Woodbury (2010) describes parametric design thinking as thinking in three different ways: with abstraction, mathematically and algorithmically. Thinking with abstraction enables producing parallel alternatives and reusing parts of the parametric model. Thinking mathematically associates to the theorems and constructions that define the scripting language for design generation and representation. Thinking algorithmically refers to the functions in the scripting language, that can add, multiply, remove or modify parts in a parametric design.

According to Woodbury (2010), a parametric design environment requires different kind of design knowledge to predict the effects of the mathematical model and the diversity and structure it can create, and to move between the intended effect and the mathematical model that creates it. Thus, merely basic design knowledge is not sufficient for parametric design.

According to Oxman and Gu (2015), parametric design process has three special characteristics. First, designers design rules and define logical relationships between them to create visual models. Compared to traditional computer design, rule sets are the basic procedures in generating the visual models of parametric (Abdelsalam, 2009). Changing parameters within the rule sets and their relationships enables finding more alternative solutions for the design (Hernandez, 2006).

Secondly, parametric design allows the designers to modify the design at any stage of the process (Oxman and Gu, 2015). All the design procedures and activities are related to each other and unambiguously defined in the parametric model (Schumacher, 2008). The design process can be open and flexible, because the designers can return back to any stage and modify parameters or rules to achieve different results (Oxman and Gu, 2015). Thirdly, unlimited number of design alternatives can be developed simultaneously at any stage of the design process (ibid.).

Parametric design has several challenges compared to traditional computer design. Finding the valid ranges for certain parameters can be problematic, and if the designer fails to define these correctly, the whole design is prone to fail (Hoffmann and Kim, 2001). Aish and Woodbury (2005) also define several challenges and advantages of parametric design. In parametric design, designers need to model also the conceptual structure that guides variation in the design in addition to the artefact being designed, which increases the complexity of the design task. Parametrization may increase the complexity of the required design decisions. Also, the number of items that need attention in each task may increase, and more additional effort may be required.

Aish and Woodbury (2005) also point out some positive task, outcome and perceptual consequences that parametrization has for designers. Parametrization can help to find designs that adapt better to the context and supports discovery of new design ideas. It also reduces the time required to make changes to the design or reusing it. Thus, parametrization supports iterative design process, where requirements evolve in every iteration and designs need to be changed. Ten Teije *et al.* (2004) define some advantages of parametric design also for web service configuration. Re-use of preconfigured templates avoids repeating same tasks, and also works as encoding knowledge to create more sophisticated solution than when creating them from the scratch.

Parametric design has long been used in mechanical engineering, architecture and other 3D design (Aish and Woodbury, 2005). Ten Teije *et al.* (2004) have implemented web service configuration as parametric design and proved that parametric design is a useful tool also in web design context. They describe a complex web service as a fixed template, where possible component services are the parameter values of the template. The template describes the skeletal control structure, which defines how the component services should be composed. In the configuration process, detailed knowledge about the template and the components is exploited to build the required composite web service.

### 2.2.2 User interface description languages

Hanus and Kluß (2009) define user interface to be structured of three different kinds of elements: structure, functionality and layout. User interfaces have hierarchical *structure*, that consists of different components, such as input fields, buttons and text elements, and composed elements, such as rows and columns of components. The tree-like structure of user interfaces can be specified by an algebraic data type in a declarative language.

When the user interacts with the user interface elements for example by mouse clicks or key presses, the elements create events that call the application program to change the user interface according to the event call. These events and the user interface changes create the *functionality* part of the user interface, which is a graph-like logical structure that connects the events and different components of the user interface. *Layout* is the visual appearance of the user interface, that is built on the structure to visualize the components.

XML has been used to create user interface description languages (UIDLs) and models since 1990's (Helms and Abrams, 2008). Helms and Abrams (2008) point that one challenge in creating user interface description languages is to handle all the different kinds of user interfaces that can be built based on the same model: simple one page or complex enterprise applications, single version or constantly updating applications, and applications for certain screen or multi-device applications. They also describe that designing a user interface description language has a goal of being able to construct the user interface as efficiently as with any other language used for user interface implementations.

Helms and Abrams (2008) describe two approaches to creating UIDLs. The first approach is to create a single language representing both the model and interface design, that can be connected to any device or platform with a single interface description. The second approach is to use different languages to specify the different interface and presentation properties, such as a task model language, a user model language and a UIDL. UIML (User Interface Markup Language) is based on the second approach.

UIML is a user interface description language, that follows a minimalistic language design (Helms and Abrams, 2008). The 3.1 version has only 36 tags, and not all of them are used in one design. UIML is based on the use of generic tags, that can be used in any context, instead of case specific tags. User interface is constructed of parts, that have presentation properties that are described as tags in UIML. Classes and properties for these tags are defined in different vocabularies, that define the UI characteristics for the certain platform used.

Helms and Abrams (2008) describe that to define user interfaces effectively, the user interface definition needs to be separated into re-usable parts. In UIML, the partition of the user interface follows the Model View Controller (MVC) design pattern described in Figure 2. In the middle is interface layer, which consists of structure, style, content and behaviour. The interface layer interacts with the two other layers, presentation and logic. Presentation layer uses vocabularies to visualize the user interface in the certain environment. Presentation layer interacts with the device or platform the user interface is built on. Logic layer interacts with the back-end application and data sources, that provide functionality to the user interface. According to Helms and Abrams (2008), UIML defines the following aspects for the user interface:

- *Structure* (What parts comprise the UI?)
- *Style* (How each part if presented?)

- *Content* (What content each part has?)
- *Behaviour* (How does each part behave?)
- *Logic* (How is the part connected to outside world?)
- *Presentation* (How is the target mapped to the UI toolkit?)

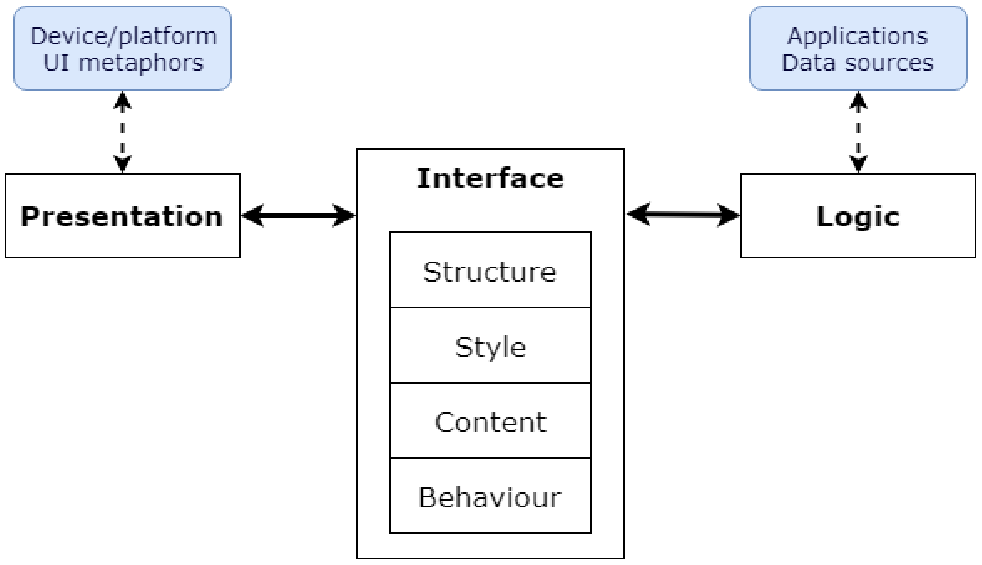

*Figure 2 Model-View-Controller Model based on Helms and Abrams's (2008) model*

Paterno' *et al.* (2009) created MARIA (Model-based lAnguage foR Interactive Applications) based on the lessons learned with previous UIDLs (like UIML) to better suit the new kind of web service oriented front-end. Implementing a user interface for a web service with MARIA is based on the design space described in Figure 3. The design space has four dimensions: abstraction level of the description of the user interface, time when the composition occurs, granularity of the user interface and aspects that are affected by the UI composition.

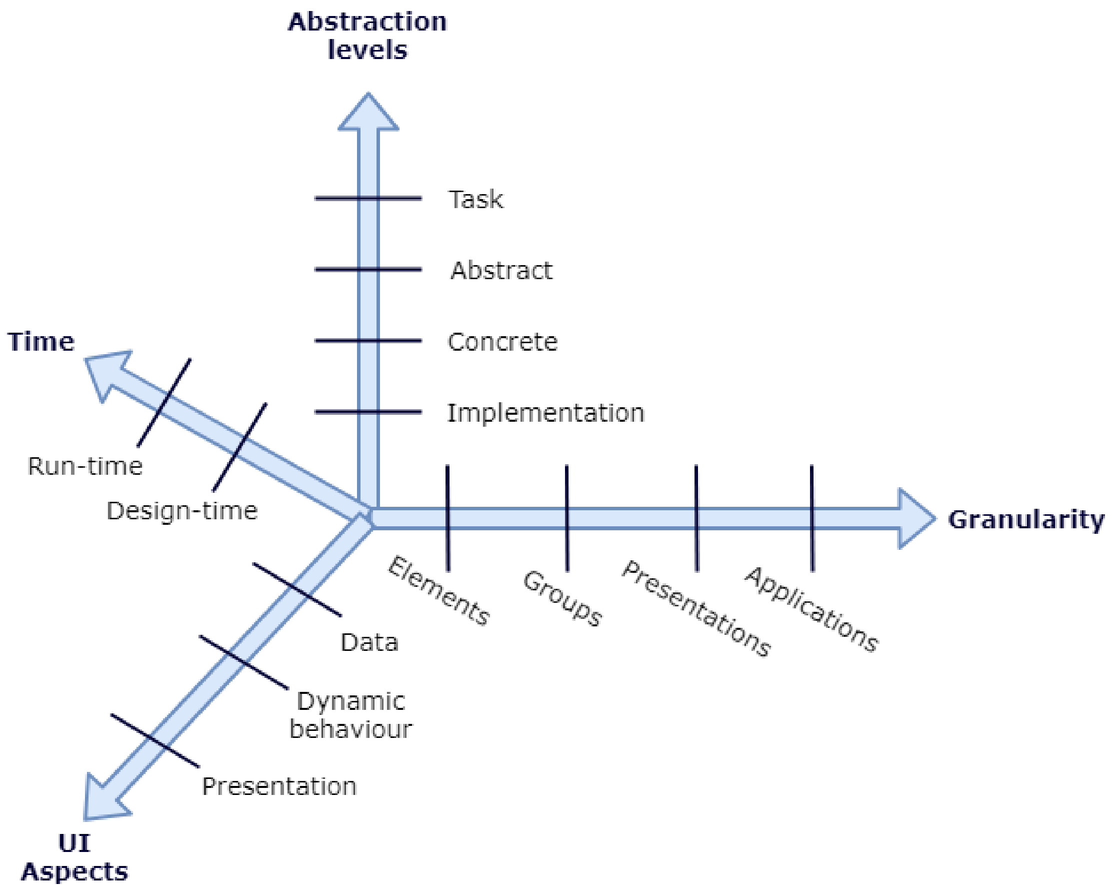

*Figure 3 Design space based on Paterno' et al.'s (2009) model*

Paterno' *et al.* (2009) describe that each dimension consists of different levels. The composition of the user interface can occur on different abstraction levels of the user interface: task and objects, abstract, concrete and implementation. Granularity describes the size of elements of the user interface: single elements like text fields, groups of objects like a whole contact form, presentation of elements and groups of elements like a whole web page, and a joined presentation of whole web application user interface.

Paterno' *et al.* (2009) also defined three different types of compositions depending on what aspects they affect: visible user interface objects, dynamic behaviour and data. With perceivable UI objects, for example the spatial relations between the composed elements need to be defined. Dynamic behaviour refers to the possible sequencing of user actions and system response, and dynamic behaviour of UI elements. The data refers to the data that is manipulated based on the events of the user interface.

Lastly, Paterno' *et al.* (2009) define two phases when the composition can occur:

design time and run time. With static web pages, the composition occurs in design time. In dynamic web applications, the user interface is changed dynamically during the execution of the application. The latter composition is used with most of the modern web sites and applications, as then it is possible to change the service UI based on the user actions or application events.

### 2.2.3 Declarative web application development

Modern web application development is based on the three-tier architecture: presentation tier, logic tier and data tier (Gustavo *et al.*, 2004). The presentation tier defines the user interface and is usually built with declarative languages such as HTML and CSS, and possibly some front-end logic scripts like JavaScript. The logic tier defines the server-side logic, that can be built with object-oriented language like Java and Python, or with scripting language like PHP and JavaScript. The data tier defines the application data, where Object-Relation Mapping or with SQL statements are used to access the data. The communication between the client and server side usually occurs with structures like XML or JSON.

Toffetti *et al.* (2011) describe three different Rich Internet Application development approaches: code-based, framework-based and model-driven methods. In code-based methods, developers code in technology-specific programming languages. Framework-based methods use more advanced libraries and code-generation tools, focusing usually on client-side development. Model-driven methods rely on automatic code-generation, thus being more comprehensive compared to the other two methods. Despite this, model-driven methods often are lacking in support for advanced web application features.

Vuorimaa *et al.* (2016) have a declarative approach to web application development, combining framework-based and model-based methods in their work. They have developed a framework that allows building whole, three-tier web applications with just one declarative language. This approach is similar to the one used in the case company for developing the web shops: the whole web application is built in XML code using ready back-end and front-end code blocks.

As Vuorimaa *et al.* (2016) describe, building web application requires different kinds of skills, as both imperative and declarative components need to be built. Just building the declarative static front-end of the application requires both HTML and CSS skills. Vuorimaa *et al.* (2016) have defined nine different levels of web application development, that describe the skill-sets of the developers:

Level 1: Customizing components
Level 2: WYSIWYG (What You See Is What You Get) editing
Level 3: Visual editing
Level 4: Mark-up authoring
Level 5: Snippet programming
Level 6: Single language programming
Level 7: Unified language programming
Level 8: Multiple language programming
Level 9: Multiple language and paradigm programming

Vuorimaa *et al.* (2016) have based their approach on level 7: unified language programming. Level 7 developers know one programming language or technology and are able to develop functional three-tier web applications with special toolkits or frameworks based on that language or technology. In Vuorimaa *et al.*'s solution declarative XML-based XForms language is used to cover all the three tiers of web application development. Declarative web development has certain advantages over imperative or multi-language development. First of all, using only single language on all three tiers can unifies the development process and reduces the amount of technologies involved Laine *et al.* (2011). Schmitz (2001) points that declarative languages are more accessible to non-professional programmers. Thus, using declarative framework would reduce the need of professional programmers in web application development.

XFormsDB framework (Vuorimaa *et al.*, 2016; Laine *et al.*, 2012) has tier-expanding architectural approach to web development, where client-side and server-side programming are unified to one model. It is based on XForms, a presentation-centric framework that uses Model-View-Controller architecture to separate the presentation, logic and data layers. Figure 4 describes the transition from a three-tier architecture with XForms presentation layer to presentation-centric XFormsDB with extended server-side and database functionality. The same architecture, that requires at least three different languages, can be achieved with only one language by using XFormsDB. In XFormsDB, each tier is described with different XML-based components.

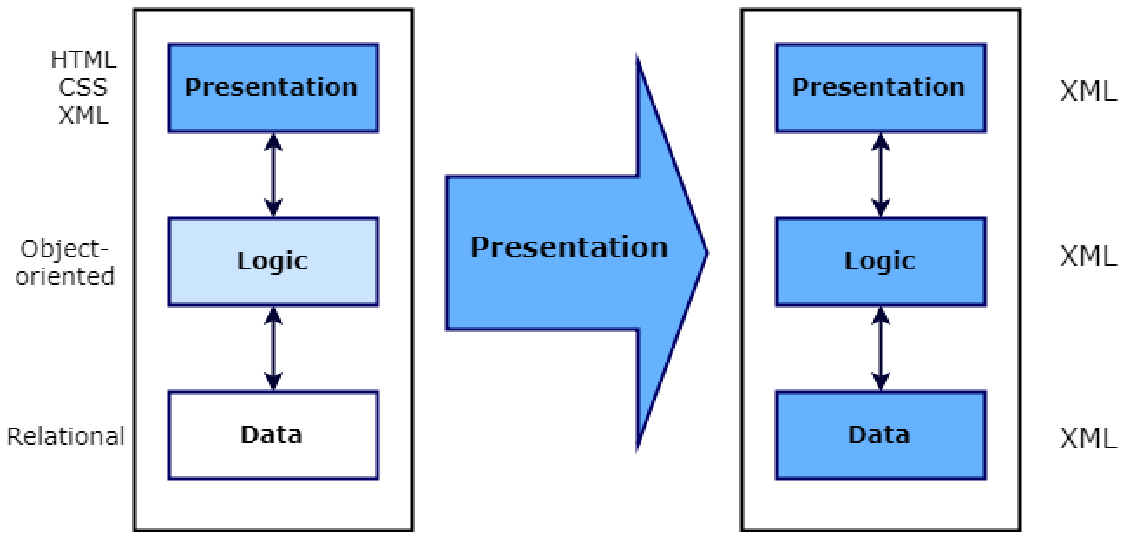

*Figure 4. The conventional three-tier web application architecture and presentation centric architectural expansion based on Laine et al.'s (2012) model.*

# 3 Methods and data

This chapters described how the empirical research in this thesis was conducted. The goal of the empirical study is to find data to answer the research questions together with the conceptual support presented in chapter 2. Based on Yin's (1994) work, two different methods were used to collect adequate data: interviews and project document analysis. Chapter 3.1 describes how the interviews were conducted. Chapter 3.2 describes the data analysis for the material gathered from the interviews. Chapter 3.3 describes the project documents and the analysis of them.

## 3.1 Interviews

Three interview participants were selected from the case company for the first round of interviews. The interview participants were salespeople from two different countries (Finland and Netherlands), that have many years of experience working in the case company and have sold many full websites for customers. Also, they have been responsible for gathering and defining the requirements the customer has for the website and been the communication link between the customer and the developer. After the first round of interviews, a fourth participant was chosen to take part in the study to support the results. The fourth participant was a salesperson from Germany, who had no success in selling a full website, but had a lot of experience in gathering and defining the customers' requirements.

The interviews were conducted as semi-structured interviews: the questions and the order of them were the same for each participant, but they could answer with their own words (Eskola, 1998). The interviews had two different kinds of questions: questions about customers and website projects in general, and project specific questions. The first two participants were asked questions about three different projects, and the third about one project. The fourth interview participant answered only general questions. The interview questions can be found in Appendix.

The goal of the interviews was to find information about the customers' requirements for the websites, the requirements gathering and website delivery process, and the salespeople's requirements for the new solution. Documents from previous website projects were read through to preface the construction of the question set. The original plan was to have six interview participants. After the first three interviews, it seemed that the interview participants were repeating each other's words. Data analysis (described in chapter 3.2) of the first three interviews supported this finding and revealed also that the interviews brought very little information about the concrete requirements the customers had for the websites. After analysing the first three interview, one interview was still made, which supported the findings from the first three interviews. Instead of making two more interviews, the research method was changed to document analysis to ensure the quality of the data.

## 3.2 Analysis of the interviews

All the interviews were recorded, and the recordings were transcribed. The analysis process followed the process of analysing qualitative data defined by Taylor-Powell and Renner (2003). The first step is to get to know the data and evaluate the quality of the date. The evaluation revealed, that the collected data might not be sufficient to support answering the research questions. The second step is to focus the analysis. Four key questions to focus the analysis were formed:

1. What requirements do the customers have for a website?
2. How does the requirements engineering process go?
3. How does the delivery process of the website go?
4. What would make these processes better?

The answers from each interview participants were organized by these questions to find consistencies and differences. Then, on the third step, the data for each of these questions is categorised. The identified categories per question are described in table 1. The categories describe answers to each question. Only emergent categories were used in this study. Emergent categories are defined after working with the data and finding recurring themes and issues (Taylor-Powell and Renner, 2003). The categorisation of the data was conducted with QDA Miner Lite software. The last step is to identify patterns and connection within and between categories. The results of this step are presented in chapter 4.

*Table 1 Categorisation of the data*

| Question | Categories |
|---|---|
| 1. What requirements do the customers have for a website? | No requirements, non-functional requirement, content requirement, example feature |
| 2. How does the requirements engineering process go? | Shows examples, asks questions, no structured process, requirements change |
| 3. How does the delivery process of the website go? | Content changes, style changes, structure changes, many rounds |
| 4. What would make these processes better? | Resources, templates, list of functions |

## 3.3 Project document analysis

Nine different website project cases were selected for the analysis:

- Three websites delivered by the case company's subcontractor
- Two website drafts made by the case company, that did not lead to a deal
- Three websites delivered by the case company
- One website project, where the customer decided to use another company for the website but the case company for the web shop

The available material for each case varied a lot, as the websites were built by different companies. The three projects, where the website was built by the subcontractor, emails between the subcontractor and the case company were analysed. In the projects, where the websites were built by the case company, the communication between the salesperson and the developers was analysed. To find detailed requirements for the websites, also the final websites were analysed from all the cases.

The document analysis process has three iterative steps: skimming, reading and interpretation (Bowen, 2009). Skimming and reading of most of the documents was done already before the interviews, as the interview questions were designed based on them. Before starting the document analysis, all the material was read through again. Interpretation followed the same steps as in the analysis of the data from the interviews: focusing the analysis by key questions and categorisation of the data. Three key questions to focus the analysis were formed:

1. What requirements did the customer have at the beginning?
2. How did the requirements change during the project?
3. What characteristics the final websites have?

Unlike in the analysis of the interviews, pre-set categories were used in the project document analysis. The pre-set categories were based on the findings from the interviews, skimming the material and reviewed literature. The website characteristics were divided into three sub-categories based on Hanus and Kluß's (2009) definition of user interface elements: structure, layout and functionality. The categorisation is described in table 2. The last step is to identify patterns and connection within and between categories. The results of this step are presented in chapter 4.

*Table 2 Categorisation of the project documents*

| **Question** | **Categories** |
|---|---|
| 1. What requirements did the customer have at the beginning? | Styling, pages, example, non-functional requirement, content |
| 2. How did the requirements change during the project? | Style change, layout change, content change |
| 3. What characteristics the final websites have? | **Structure**<br>Page, sub-page, page type, content block, content type, header, footer, navigation<br><br>**Layout**<br>Colours, text, spacing, content style<br><br>**Functionality**<br>Languages, movement, content functionality |

# 4 Results

This chapter presents the results of the empirical research and analysis, that were described in the previous chapter. Two different empirical research methods were used: interviews and project documentation collection. Material from these methods were analysed separately. The first section of this chapter presents the results from the interviews, and the second of the project document analysis.

## 4.1 Interviews

The goal for the interviews was to collect information about four different aspects: customers' requirements for a website, requirements gathering and definition, website delivery process and suggested improvements to the current processes. Results from these four topics are presented in sub-chapters.

### 4.1.1 Customers' requirements

Answering the search question 2 (*How can website requirements be parametrized?)* requires finding out the differences and similarities between the requirements that different customers have for the website. These differences and similarities work as a base for the parametrization. The interview participants were asked about requirements in specific cases, but also in general. Based on these answers, two different types of customers were identified.

The first customer group has very little knowledge about websites and website requirements. They have very limited skills of describing their needs, and only describe very general and small requirements if any. All four participants described this kind of customers. Three out of four interview participants described only this kind of customers with very short requirements list or with no requirements.

*"They didn't really have any requirements for the website because they*

> *didn't have one and didn't really know about things. [...] The customer really didn't know much about anything of these things so didn't really ask for anything. They were ok with everything I suggested.”*
> – Participant B

> “*The good thing is that their requirements are quite simple. Because they're in situation that they don't really know about all the fancy new stuff, so their only requirement is that it needs to be responsive, and easy to use.”*
> – Participant A

> *“They don't have any, because they don't know what they should require.”*
> – Participant D

It is typical, that this kind of customers find something from another website, and list that as their requirement, even if they are not able to describe it properly. They also might already have some solution and want to keep that one but make it better. This kind of requirements are rarely based on technical knowledge but are the only options the customers are aware of.

> *“They've spotted something on another website but they don't know what it actually does or how it works, and then they try to describe it to us in well almost a binary language for us. We of course don't understand, and they don't have a clue what they're talking about so most of the time they try to send us examples from other websites.”*
> – Participant C

> *“They were not IT people so they had very different opinion on some things what is good and they wanted to keep the original CMS.”*
> – Participant A

The second customer group has more knowledge about websites and has a much more detailed requirement list than the first group. These are bigger companies, that already have style guidelines, content for the website and have business in many countries. One interview participant described many this kind of customers.

> *“And the other ones, they're more sophisticated ones. They know what they're buying and they want to have a good product [...] and a multilingual website.”*
> – Participant C

The main requirements all the companies are describing for a website are the that it needs to look good and modern and be simple to use. Some companies describe this further, but it is common that they do not have more specific requirements.

> *“They wanted it to be responsive and easy to use and look good, and also they wanted a good web shop. The site really was like from 70's so it just had to be modern.”*
> – Participant B

> *“The key points that they actually described me were about the layout and how it should feel modern and look good. [...] So basically, what they wanted was a good-looking, simple to use solution.”*
> – Participant A

> *“It was a customer who had a really old website made by himself, but what he really wanted was a modern website and simple solution to update the machines.”*
> – Participant A

The more specific requirements are usually about the web shop, not the website built around it. The most detailed website requirements described were about placing static content or buttons on the website and in what languages the website should be. None of the interview participants described any specific functional requirements in addition to these.

> *“I think those are important where their logo's shown and their contact details.”*
> – Participant C

> *“There's some basic information like one customer had few different brands and wanted to build pages for them”*
> – Participant A

> *“Their current website was lacking multiple languages.”*
> – Participant C

Altogether, based on the interviews, the customers describe very little requirements for the websites. They are able to tell whether they like certain solution or not, but at the beginning of the project, they generally are not able to describe what they want in more detail.

### 4.1.2 Requirements gathering and definition

Answering the research question 3 (*How can a parametric model of the website support finding and defining the customer specific requirements?)* requires studying the current process of gathering and defining customers' requirements in the case company. The interview participants described how they present to the customers what the case company can provide for them, how they find out what the customer needs and how the process continues through the project.

Every one of the interview participants had their own way of finding out the requirements and presenting possible solutions. All the participants showed some kind of examples of previous websites the case company had delivered to other customers, or from other websites. From these examples, the customers could show what kind of things they would like to have on their website.

> *"We try to present like examples of our own website or websites from other customers or potential customers, and from there try to get these components together to clear what they really would like to have."*
> – Participant C

> *"We can show them some example we have made before, and then we show some websites that other our customers have. And then we tell what things we can build there."*
> – Participant B

Two of the participants described, how they ask questions from the customer about their wishes. When they show examples, they are asking the customers' opinion about them, and suggesting possible solutions. None of the participants had any structured way of finding the requirements and presenting and suggesting possible solutions.

> *"We try to ask them a lot of questions about what direction they would want to go. Does the website need to be responsive, yes and no, does it need to look good on mobile phone. Well, nowadays we don't sell anything that is not, but still we ask the questions to look smart. And then*

*from there we try to ask to a direction where we know we have our strength and pondering the answers is giving us a quite clear idea how the website should look like.”*
– Participant C

*“It depends on the customer. [...] We just go through some of the websites we have done previously, or sometimes we just go through the websites of competitors, even if they are not done by us, and ask what kind of things they like there and what kind of things they'd like to have on their own website.”*
– Participant A

Many of the requirements are only found after delivering the first version. When seeing the first version, customers are able to tell what they do not like there and what things should be changed and how. More demanding customers also come up with completely new things when the first version of the website is provided or even later in the process. The requirements need to be reviewed through the website delivery process, and new agreements about the requirements need to be made in some cases.

*“They change quite a bit, so what we try to do at the beginning is that we nail down the things what we are going to do, and then as soon as someone starts asking for other things, then we easily go back and took our information like yeah, this is what we agreed on, we can make it, takes about two hours and then we will take the IT hours out of it.“*
– Participant C

Each of the participants have their own ways of finding and defining the customers’ requirements before and during the project, but everyone’s process is still very simple, as the projects are also simple. No one described any other requirements engineering techniques but showing examples of possible solutions and asking customers what they like in certain solution. In addition to the list of requirements the customers are able to provide, those are the only ways of gathering and defining requirements before building the first version of the website. The requirements are changed and refined after each version based on the customers’ feedback, but the participants do not describe any activity, that would support iterative requirements engineering throughout the project.

### 4.1.3 Delivery process

To support finding answer to the third research question, the interview participants were also asked questions about delivering the websites to the customer. As described in the chapter 2, the success of the requirements engineering process affects the whole delivery process of the product. Also, as the requirements engineering continues also in the later stages of the project, gathering and defining the requirements is closely related to the whole delivery process of the product.

All the website projects described in the interviews, that have resulted in the customer buying the website, have had some changes after the first version provided to the customer. Mostly, the required changes are small style changes, content changes or additions, and site and page structure changes. Overall, the customers were always happy with the final solution provided.

> *"There were some things that had to be changed, like that thing should be here and that there and stuff."*
> – Participant B

> *"Most of the time it was texts and images what had to be changed"*
> – Participant C

> *"Change locations, maybe change colours, replace items or move things around."*
> – Participant C

> *"It wasn't perfect, it needed some finetuning. But the basics were okay."*
> – Participant A

The amount of iterations needed to deliver the final solution that satisfied the customer was related to the amount of content in the website. The simplest websites required only one round of changes, but the ones with a lot of content could require more than ten rounds. The website projects, that were not ordered from a subcontractor, did not have content management system, so the customer needed to send even small changes to text and images for the case company to update to the web site.

> *"If he needs something changed, he can just send us the changes and we can do it for him."*
> – Participant C

*"Well probably at least ten times we needed to change some points. Most of the time it was about little things, about what details should be and where. They provided a huge bunch of different images that needed to be changed, and they came up with like media buttons like Facebook and stuff. And texts and links needed to be changed, but nothing really major."*
– Participant A

### 4.1.4 Improvements

To find the best support for answering the third research question, the interview participants were also asked to describe improvements to the current system. Finding out, how the people, who are responsible of gathering and defining the requirements for the customers' websites, would improve the process, works as a base for the design of the new solution. All four interview participants described a similar improvement to the requirements engineering process: ready templates for different kinds of solutions, that would work both as an example and a base for the requirements.

*"Simple examples, one like really basic, second one a little bit more advanced, and a full customized version, where we could show that these are the websites we do and these are the functions that they have. [...] We would put their content in, their colours and pictures in, and the site would be quite basic."*
– Participant A

*"Well I think more of those templates that I could show the customer how it would look. That would make my job easier."*
– Participant B

*"In the past we thought of having two or three solutions. Simple solution, an advanced solution and professional solution."*
– Participant D

*"And It would be good to have a list of five or six, or maybe even three only, with possible layouts people can look and choose one."*
– Participant C

In addition to the website templates, a list of all possible features and components to be put into the template could be provided to speed up the process. The slowness of delivering the websites for the customers is also the main consideration for two participants, who both suggest that having more people making the websites would be the first improvement for them.

> *"Well we would need a couple of more people who actually know how to do websites."*
> – Participant A
>
> *"More people doing them (websites)."*
> – Participant B

## 4.2 Project documents

After analysing the interviews, material of past website projects was collected. As described in chapter 4.2, nine cases were selected to study. Three of the websites were ordered from the case company's subcontractor, two were own projects that did not lead to a deal, three websites that the case company has delivered, and one project, where the customer decided to use another company for the website, but the case company delivered the web shop for the website.

Two different things were analysed from the case material. First, the customers' requirements at the beginning and their changing during the project were analysed. Then, the characteristics of the final websites were collected and differences and similarities between them were analysed.

### 4.2.1 Requirements

At the beginning of the project, most of the customers provided very minimal requirements. All the companies provided a list of pages on the website, that was also the list of tabs in the navigation bar in header. The companies also had from one to three colours that needed to be used on the website. Only one of the case customers provided full design guidelines, that described fonts, margins and paddings in addition to the colours. One case company delivered all the required content (text and images) to the pages already at the beginning, but most of the companies described, what kind of content they want but delivered the material later in the project.

In the projects, where the case company delivered the whole website, an example website which had the desired layout was usually provided. In case 7, the company had already two other websites, where the layout for the new website had to be copied from. In case 3 and 4, previous websites delivered to other customers were used as models. Case 1 layout was based on the existing website the customer had but was modified to be responsive. Case 2 layout was chosen from ready Voog[3] templates. In the projects, where the website was provided by the subcontractor, customer chose one of the layout templates provided by the subcontractor.

After the first version of the website is delivered, customers usually ask to change and move content on the site. Only one case (case 6) of the six cases, where the customer decided to buy the website from the case company, the customer did not ask any changes, and the site went live with the first version provided. In case 4, the customer asked changes to the website content, but was happy with the layout. It took four rounds of changes to content to get the site ready. In cases 3 and 5, the customer wanted to have small layout changes to the first version of the website, and it took over ten rounds of content changes to finish the website. In case 7, the customer required changes to the layout on many rounds, and content changes for over ten rounds in total before the final version.

### 4.2.2 Characteristics of the websites

In the second phase of the project document analysis, the final websites were analysed. Three different aspects of the websites were studied: structure, layout and functionality, based on Hanus and Klußʼs (2009) definition of user interface elements. The structure is divided to hierarchical structure and layout structure. For clarity, layout is called *styling* in this study.

#### Hierarchical structure

Hierarchical structure describes the relationships between different pages of the website. All the websites in the case study consist of five different kinds of pages. They all have a home page, and four different kinds of sub-pages: web shop page, contact page, content page and news page. Web shop is a dynamic solution, that is usually integrated to customer's existing web site. In this case one web shop can be described

[3] Voog https://www.voog.com/ (Accessed on 25th May, 2018)

as one page, as it is a separate application. Contact page is also a dynamic page with contact form and static content. In some cases, the contact page also has a dynamic map of the company locations. News page is either a static page with news or a dynamic news feed application. Content page is any page with static content, like text and images.

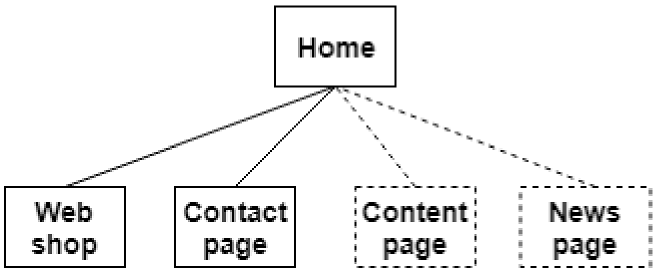

*Figure 4 The basic hierarchy of the websites*

The basic structure of the websites is described in Figure 4. All the websites had at least one web shop and contact page in addition to the home page. The simplest website, Case 1, had only those two. Many of the websites had also either one or more content pages or news page, or sometimes both. Web shop, contact page and news page are always on the first level after home page, but content pages can be also sub-pages down to fourth level. Figure 5 shows the structure of the more complicated pages with up to four levels of sub-pages. One content page can have basically any number of sub-pages. In some cases, also the contact page had content pages as sub-pages.

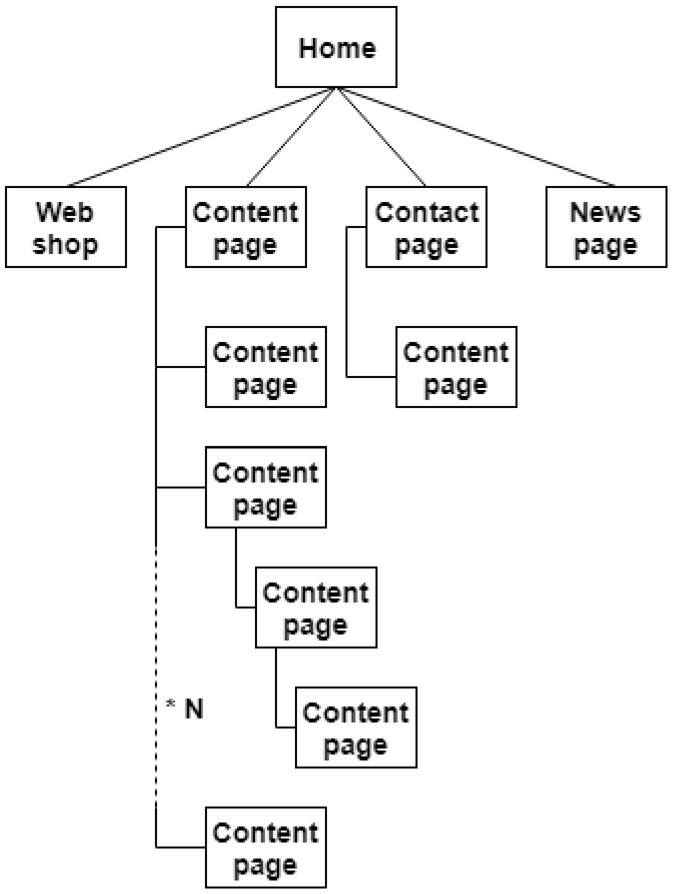

*Figure 5 Hierarchy of the websites with multiple hierarchical layers*

Table 3 describes the amount of first level pages, which is also the number of tabs in the navigation bar. All the websites have one or two different web shops. They all have also one contact page. Cases 1 and 2 do not have any content pages, but all the other cases have from 1 to 5 different first level content pages. Cases 6, 7 and 9 also have a news page.

*Table 3 The amount of first level pages per case*

| | Web shops | Contact page | Content pages | News page |
|---|---|---|---|---|
| **Case 1** | 1 | 1 | 0 | 0 |
| **Case 2** | 2 | 1 | 0 | 0 |
| **Case 3** | 1 | 1 | 1 | 0 |
| **Case 4** | 2 | 1 | 3 | 0 |
| **Case 5** | 1 | 1 | 4 | 0 |
| **Case 6** | 1 | 1 | 1 | 1 |
| **Case 7** | 1 | 1 | 5 | 1 |
| **Case 8** | 1 | 1 | 5 | 0 |
| **Case 9** | 1 | 1 | 5 | 1 |

Table 3 describes the amount of levels in the website hierarchy. As the websites have only content pages below the first level in the hierarchy, only the amount of content pages is needed. Together with table 3, table 4 shows the total amount of different pages in each case. Cases 1-4 have only one level of pages. Cases 7 and 8 have two levels, cases 6 and 9 three levels and case 5 has four levels of pages. In cases 5 and 9, the sub-pages were not visible in the navigation, but were shown as links in the content of the page. The number of these sub-pages is irrelevant in this study, as the amount of sub-pages changes when the content on the page changes.

*Table 4 The number of different level content pages*

| | 1st level content pages | 2nd level content pages | 3rd level content pages | 4th level content pages |
|---|---|---|---|---|
| **Case 1** | 0 | 0 | 0 | 0 |
| **Case 2** | 0 | 0 | 0 | 0 |
| **Case 3** | 1 | 0 | 0 | 0 |
| **Case 4** | 3 | 0 | 0 | 0 |
| **Case 5** | 4 | 17 | N | N |
| **Case 6** | 1 | 5 | 2 | 0 |
| **Case 7** | 5 | 3 | 0 | 0 |
| **Case 8** | 5 | 3 | 0 | 0 |
| **Case 9** | 5 | N | N | 0 |

In the content management system that is used in the new website solution, the same pages in different languages are described as separate pages. As shown in Table 5, the number of languages ranges from one to twelve. This means that the site has from one to twelve different versions, where the hierarchies are identical. The different language versions are copies of each other, and only the language of the content changes.

*Table 5 The number of languages in each case*

| | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| **Languages** | 3 | 3 | 2 | 6 | 1 | 1 | 12 | 1 | 1 |

## Layout structure

Layout structure defines the structure of each page on the website. The layout of the websites consists of three different kinds of pieces: header, footer and content blocks. All the home pages and contact pages have very similar structure to each other, but the content page structures vary a lot.

### *Header*

In the case websites, three different kinds of header structures were found. The third kind of header was only in case 7 – all the other cases had header structures described in figures 6 and 7. The header has usually three different kinds of components: company logo, a horizontal navigation bar with different amounts of menu items, and a language selection button if the site has multiple languages. Figure 6 describes the version where the logo and navigation bar are horizontally on the same line. Figure 7 describes the version where the logo is above the navigation bar. Language selection button is found from the upper right corner in both versions.

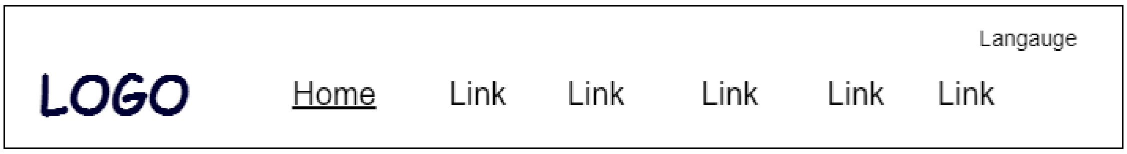

*Figure 6 Header structure with logo and navigation bar on the same row*

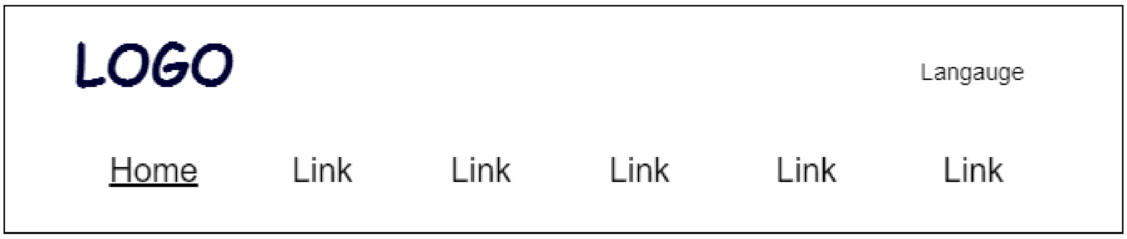

*Figure 7 Header structure with logo above the navigation bar*

In case 7, the header layout was similar as figure 7, but the logo was on the right and language selections on the left. In the mobile versions of the websites, the navigation bar is hidden behind a button. When the button is clicked, a vertical version of the navigation bar appears.

### *Footer*

The footer is a different in each case. Footers consists of different amounts of HTML blocks with different content. What is common between the footers in different cases, is that all the HTML blocks are aligned horizontally. Thus, every footer has from one to N HTML blocks in a row. All the HTML blocks have a little bit different content. The content can be text and images, but also links and buttons.

### *Page contents*

All the pages in the case websites have the same header and footer that were described earlier. The page content between header and footer changes on every page. The home pages in the cases had either one or two columns, which consist of different kinds of content blocks. Figure 8 shows the two different versions. The two-column version has one vertical content block on the left, and varying amount of content blocks on top of each other. The one-column version is similar as the right column of the two-column version. The content blocks have either static content like image and texts, or dynamic web shop content. The number of content blocks varies from three to eight blocks.

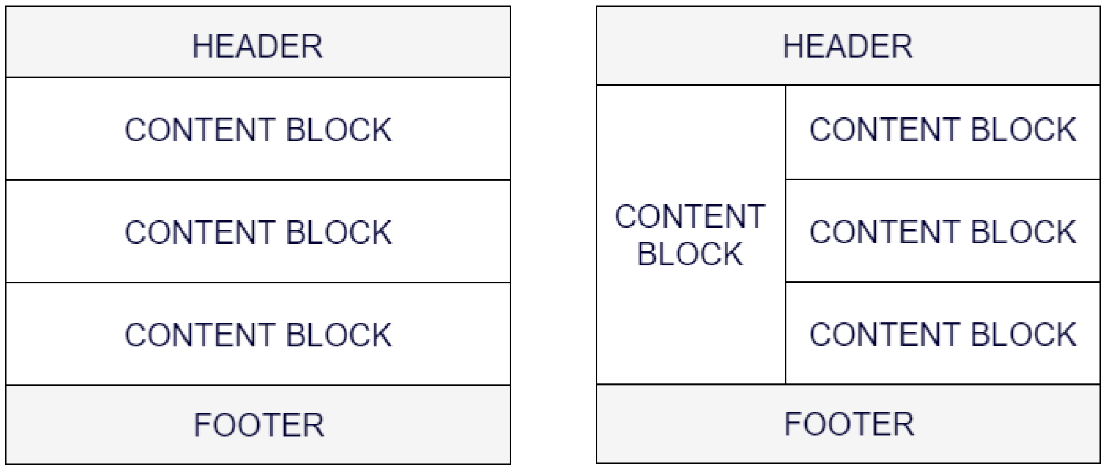

*Figure 8 Home page with one or two columns*

Content pages have similar layout as home page, but the number of columns varies from one to five. Contact pages have one or two columns, with contact form, map, text and images as content.

## Styling

The appearance of the websites is defined with CSS style rules. The different style rules for the case websites can be divided to three different categories: text, colours and frames. Different parts of the page can have different style rules. All these parts are called content blocks in this section, including footer and header.

Text styles affect all the texts on the web site. All the websites had default text styles, from three to five different header text styles and link text styles. For each of these, font, size, font weight and line height were defined. In some cases, also the navigation bar links and footer texts had their own font, size, weight and line height. Links had extra style rules when they were hovered or active: either underlining or colour change.

All the case websites had one main background colour, which was white or light grey. In addition to that, they had from two to five different colours used in styles. All the texts described previously have the default colour defined. As mentioned, link texts can have also a different colour for active and hovered links. Some content blocks have different background colour and border colour defined.

Each piece of content, like text or image, and content blocks have frames, that consist of padding, border and margin. Default text has zero padding and margin and no border by default, but navigation bar links and header texts have their own frame styles. All content blocks and images also have default paddings, margins and borders. Padding and margin only have width defined, but borders have also colour and style defined according to the CSS syntax.

## Functionality

Most of the advanced functionality of the website is in the form of the web shop content, either as a web shop page or other web shop functionality items, like contact form, recent ads or search form. Also, a dynamic news page that some cases had, is a separate news web application integrated to the website. The map on the contact page is implemented with Google Maps[4], except in case 7, where the map is a dynamic vector image. The most important advanced functionality, that affects the whole website, is changing the language. When the user clicks to change the language, the page loads again with the content on the selected language.

Most of the case websites had an automatic image slider on the front page. Also, in half of the cases the site header was dynamic and stuck on the top of the window when scrolled down. The websites made by the subcontractor of the case company had images to slide in smoothly after the page load. Other functionalities of the websites

[4] Google Maps platform https://cloud.google.com/maps-platform/ (Accessed on 27th May, 2018)

are hover and active styles for links, and default HTML content behaviour defined by the browser, like drop down opening.

# 5 Discussion and conclusions

This study covered two fields that have been researched, but not combined in one study: parametric design and website development. Parametric design has been studied in the context of architecture, mechanical engineering and other 3D design (Aish and Woodbury, 2005). ten Teije *et al.* (2004) applied parametric design in web service context but focused on the configuration of the service. Parametric thinking has been a part of user interface construction and web application development in many models and frameworks, some of which are also described in this thesis. This study develops these further and brings parametric modelling into website design. The parametric model presented complements the existing web design tools.

## 5.1 Answers to research questions

In this chapter, the answers to research questions are presented. First, possible ways of describing a website as a parametric model are discussed based on the conceptual background. Second, parametrization of website requirements is discussed based on the empirical research results. Third, the role of a parametric model in requirements engineering is evaluated based on the empirical research results and conceptual background.

The research problem of this study was defined as "*How can parametric design be utilized in website design?*". Three research questions to support solving the research problem were formed:

*RQ1: How can a website be described as a parametric model?*

*RQ2: How can website requirements be parametrized?*

*RQ3: How can a parametric model of the website support finding and defining the customer specific requirements?*

### 5.1.1 Describing a website as a parametric model

As defined in chapter 2.1.1, in parametric design the designed objects have the same overall structure, and variations of objects can be generated by defining values for parameters within the object template (ten Teije *et al.*, 2004). In the conceptual background, two different approaches to defining web applications or user interfaces as general structures with varying details and values were presented: user interface description languages and declarative approach to web application development. These examples work as a base for the parametric model of the website that was developed and will be described in the next chapter. The solution, that will be built based on the model, is going to use XML format for defining the parameter values, so the examples chosen for review in this study are XML-based.

UIML is a user interface description language developed by Helms and Abrams (2008). User interface is constructed of parts, that have presentation properties described as tags in UIML. Characteristics of the user interface are defined as classes and properties for tags. Classes and properties are defined in vocabularies. UIML can be described as a parametric model: tags are different parameters, and classes and properties are parameter values. Vocabularies serve as templates for the model.

Paterno' *et al.* (2009) developed MARIA, that expands the knowledge from UIML and other user interface description languages. Designing user interface items with MARIA is based on design space with four dimensions: abstraction level of the description of the user interface, time when the composition occurs, granularity of the user interface and aspects that are affected by the UI composition. MARIA can also describe a web application as a parametric model: tags similar to the tags in UIML describe parts of the user interface and can work as parameters as the UIML tags. The design space values work as parameters that describe the characteristics of the web application.

Laine *et al.* (2011) used declarative approach to web application development and created XFormsDB, that is an XML-based framework for web applications. With the framework, all parts of the web application are described as XML components. These XML components are tags with attributes, that are defined with vocabularies and libraries. Just as the previous examples, the XML tags defining the components can also be described as parameters with attributes as values.

### 5.1.2 Parametrizing website requirements

Creating a parametric model of a website includes parametrizing the website requirements. The first step is to define the requirements for the websites. Then all the similarities in these requirements are identified. Requirements, that are the same for each website, become fixed part of the layout and are not described as parameters. In this study, the non-functional requirements were the same for each case, and were considered when creating the layouts. Also, all the websites followed the same hierarchical structure and have the same layout structure for the pages: header (with a horizontal navigation bar) – changing content – footer. These are built in the layouts.

Third step of the parametrization of the requirements is to define the aspects of the websites, that have different requirements in each case, and find the range of the possible requirements for the aspect. While the hierarchical structure of the websites remains the same, the number of pages on each level varies from 1 to N pages, and the number of hierarchical levels varies from 1 to 4. Also, the number of content blocks between the header and footer on each page varies from 1 to N. The page content is divided from 1 to 5 vertical columns and have from 1 to N rows of content blocks. The content blocks are either type 1 or type 2.

The last step of parametrizing the requirements is combining the findings to a user interface or web application model. In this case, the requirements differed only in user interface aspects. Thus, the UIML component groups (Helms and Abrams, 2008) were a suitable base for the new model. The outcome of this process is a parametric model of a website based on the defined requirements, that will be presented in chapter 5.2. The parametric model developed is an example of parametrization of website requirements.

### 5.1.3 Parametric model in requirements engineering

The study revealed several shortcomings in the requirements engineering process. The first problem is that the communication about the requirements is not sufficient. Many of the customers have little or no knowledge about websites. Saiedian and Dale (2000) have recognised the same problem: customers are not aware of all the technical constraint, which creates a knowledge gap between the stakeholders. Differences in problem perspective between the stakeholders challenges the communication. Parametric model enables fast creation of high-fidelity prototypes of the websites. Prototypes help to generate knowledge about the system (Saiedian and Dale, 2000), and narrowing the knowledge gap to ease the communication.

Prototypes also work as structuring resources in communication between stakeholders (Johansson and Arvola, 2007). Saiedian and Dale (2000) point that using visualizations of the product can help clarify and detail the understanding of the requirements. They point that ensuring that the customer understands what kind of product is possible to be delivered with certain resources is key to defining such requirements that can be met. The case study also shows that many of the requirements are defined only after the customers see the first version of the solution. With a prototype created with the parametric model, these requirements could be defined already at the beginning.

In general, that prototyping ensures improved communication between stakeholders (Käpyaho and Kauppinen, 2015). According to Saiedian and Dale (2000), the success of requirements engineering process depends heavily on the success of the communication between application developers and other stakeholders. The salespeople act as intermediaries between the developers and customers in the case company. Intermediaries often lack in understanding of the needs and constraints of the sides involved in communication (Saiedian and Dale, 2000), which highlights the need for successful communication.

The high number of iterations needed in requirements engineering to achieve the final solution in some of the cases suggests that the requirements engineering process is not efficient. The interview participants describe different ways of gathering the requirements and presenting possible solutions, but none of them had any structured process for that. They also did not describe any activity, that would support iterative requirements engineering throughout the project. The prototypes generated with the parametric model can structure the process. The model is also designed to be used iteratively, which supports iterative requirements engineering process.

## 5.2 Parametric model of a website

Based on the findings described in chapter 4, a parametric model of a website was developed to support the requirements gathering process. Three different boilerplate websites have been developed earlier to work as templates for the websites. For the first prototype version of the website, one of the ready website layouts is chosen. It can be either the customer or the salesperson who chooses the layout that they think would best suit the customer's needs.

The division of the website into parts is based on UIML (Helms and Abrams, 2008) described in chapter 3.2, the interface layer from figure 2:

*Structure* (What parts comprise the website?)

*Style* (How each part is presented?)

*Content* (What content each part has?)

*Behaviour* (How does each part behave?)

*Structure* and *style* are their own parameter groups in the model. *Content* is added to the website in the content managements system to the content block areas that are defined in the structure. *Behaviour* is default for each layout and content type, as customers rarely require any certain behaviour. Only the header behaviour (whether sticks to the top when scrolling down) is included in the model as a part of the style.

To create a new website, three parameters described in figure 9 need to be defined: siteName, layoutID and logoURL. SiteName is the name of the website, which is usually name of the company. LayoutID is the id of the chosen boilerplate layout. LogoURL defines the location of the image file of the logo. LogoURL is not a mandatory field but having the company's logo in the header is one of the basic requirements in all the websites in this study, so it can be defined already at this point. With these parameters defined, a new website with the layouts default colours and default sub-pages is created.

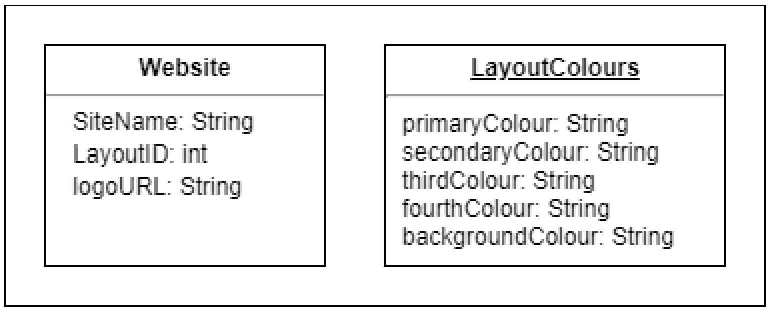

*Figure 9 Parameters in the first stage of website development*

As described in the previous section, the customers provide from one to three different colours to be used in the website at the beginning of the project. The layout templates have five different default colours described in the figure 9 defined:

primaryColour, secondaryColour, thirdColour, fourthColour and backgroundColour. Each piece of the layout has some of these colours by default. To customize the website to the customer’s liking, these colours should be replaced with the ones the customer wants to have on the website.

The structure of the website consists of four different groups of parameters described in figure 10: FirstPage, SubPage, Languages and PageContents. FirstPage defines the parameters for a first-level page that is shown in the navigation bar in header. PageName defines the translation code for the name of the page, that is shown on the navigation bar and in the URL. PageType is the number of the desired page type: content page (1), web shop (2), contact page (3) or news page (4), which are based on the page types found in the case websites. SubPage defines the parameters to a page, that is a sub-page to any other page. As the SubPages are always content pages as noted in the previous section, PageType does not need to be defined. SubPage always has one direct parent page, which is defined with parentPage parameter. The value of parentPage is the pageName of the parent page.

Languages define the default language for the website, that is used when the user visits the website for the first time, and other languages that the user can choose to view the website on. PageContents define the content structure for the page. PageName defines on which page the content is. Columns defines the number of vertical columns on the page. ContentSlots defines the total amount of content slots. ContentSlotTypes lists the types of the content slots in the same order as they appear on the page. Contents can be either web shop content (e.g. latest products window, contact form or search form), that is built in the case company’s system, or other content (e.g. text, images or map) that is added to the page in the content management system.

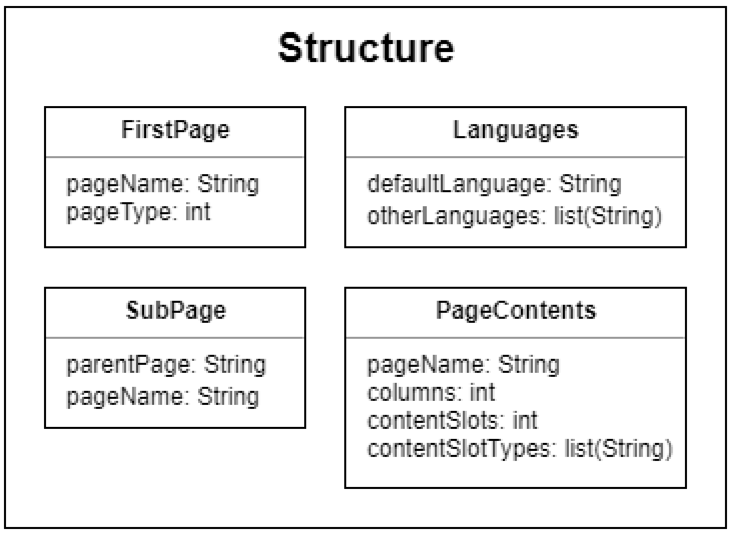

*Figure 10 Parametric model of the structure of the website*

First level pages, some sub-pages and languages are something that the customers usually know already at the beginning of the project, as was found out in the research. The first version of the website is usually made with them and the layout colours defined. After the first version, the customer has more resources to define requirements for the content. Then, the missing content slots can be defined. After the first version, many customers also want some small changes to the layout styles. To make that as easy as possible, style parameters for different parts of the layout have been defined. They are shown in figure 11. These style parameters override the equivalent style rule in the layout.

Most of the style parameters are based on CSS style rules. Header has two style parameters, that make more advanced changes: sticky and showFlags. Sticky defines, weather the header stays on top of the window when scrolled down, and show flags defines, whether the items shown in the language menu are country flags or language names. Other texts (link and headers H1-H5) inherit the font and colour from Text, unless they have own values defined.

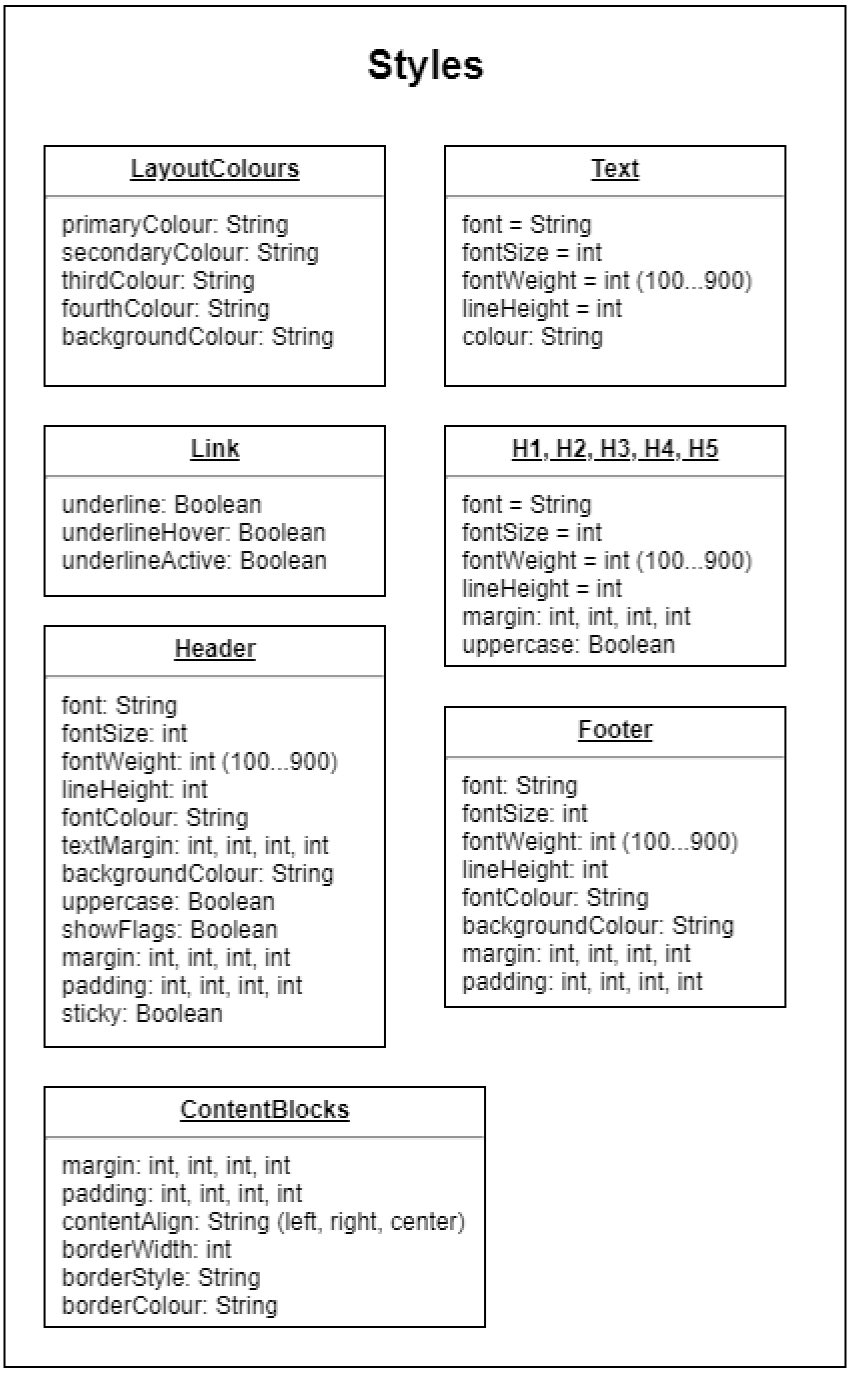

*Figure 11 Parametric model of the styles of the website*

## 5.3 Implications

The parametric model developed in this study has several implications in the case company. The description of the parametric model works as guideline for the new website development solution. The solution will be implemented purely based on this study. The goal of the solution is to make the website development process faster and more efficient. The lack of competent developers that could develop the websites was stated as a shortcoming of the current solution. By speeding up the development process, one developer will be able to develop more websites in certain timeframe. Also, as the website can be generated only by modifying the XML based model, the required skill-level for the website developer will be lower. With the parametric model, only unified language programming level (level 7 in Vuorimaa *et al.'s* (2016) definition) is required, whereas the current required skill level is multiple language programming (level 8). This means that more of the existing developers in the company will be able to develop websites for customers.

The parametric model supports the requirements engineering process in many ways that are described in chapter 5.2.3. In practice, the developed model creates guidelines to support the requirements engineering in the customer projects in the case company. It also supports the whole sales process, not just the requirements engineering and development. High-fidelity prototypes created with the parametric model provide a good basis for thorough evaluation of the product for the customers and can be used effectively to encourage customers to buy the product (Rudd *et al.*, 1996).

## 5.4 Limitations and validity

The research methods used create some limitations for the study. The study was conducted in a one company with a very specified field. Also, the interview participants were only from three countries: Finland, Netherlands and Germany, but the case company has customers and salespeople also in many other countries. The interviews were limited to the case company employees, and the customer point of view was left out from the study. Also, the number of case projects was limited, and several website projects from the previous three years were excluded from the study. The amount of project documentation available varied a lot between the cases, so some of the cases were studied more thoroughly than others.

The validity of this study is evaluated based on three validity criteria for qualitative research presented by Whittemore *et al.* (2001). The first criterion *credibility* evaluates whether the research results reflect the opinions and experience of participants, and project documents truthfully. In this study, the interview results are based on direct quotes of the participants. Still, the analysis of the results required making interpretations. The author of the study is an employee of the case company and has participated in some of the case projects. Completely objective interpretation of the results was not possible, as the author had knowledge about the cases that did not directly arise from the interviews and project documents.

The second evaluation criterion for this study is criticality, which addresses the ability to critically reflect the results and theoretical background (Whittemore *et al.*, 2001). The limitations of the study described earlier were considered when analysing the results and have been taken into account in the presentation of the model. The theoretical background was selected critically, and the trustworthiness of the resources was evaluated.

The third evaluation criterion used is integrity, that refers to recursive validation of the research and humble presentation of the findings (Whittemore *et al.*, 2001). The first analysis of results was conducted after three interviews. Based on the findings, the validity of the method for the research was questioned, and a decision of using project document analysis as an additional research method was made. The validity of research methods and results was considered throughout the study. The parametric model developed is described as one possible solution in certain conditions. The model presented is not considered as a final solution, but as a guideline for the new solution, that evolves as time passes.

## 5.5 Future work and conclusions

In this study, parametric design and website design were combined in a new way. A parametric model of a website was developed to make the design and development of websites for customers in the case company faster and easier. More work in the future is needed to overcome the limitations of the study. In this study, the requirements that the customers have for the websites were only gathered from the project documents and by interviewing the salespeople. In the future, also the customers point of view to the requirements should be considered.

This study focused on developing the possible model. Reviewing of the success of the model in use at the company was out of scope in this research. In the future, the effects of the model in requirements engineering, sales process and development process should be studied to evaluate the functionality of the model in practice. To take the simplifying of the development process even further, a graphical user interface (GUI) for defining the parameter values could be developed. With a GUI, no development skills would be required to create prototypes of the websites and even simple, ready websites.

This study was conducted as a case study in one company in a specialised field. In the future, the model should be validated also in other contexts. The model could be developed further to be more general, so that different kinds of websites could be made with it. This would require researching the subject in different kinds of companies with different kinds of customers. Having different kinds of customers means that also the requirements they have for the website are different.

The goal of this thesis was to produce a proposal of the new solution for developing full websites for the case company's customers, that would also help the salespeople in their work. The parametric model of the websites was developed, and its effects on the requirements engineering, sales and development processes were evaluated. Based on the results and literature, using the parametric model would have positive effects on these processes. Thus, the research has met its practical goal, and also provided knew knowledge about applying parametric design to requirements engineering and web design.

# References

Abdelsalam, M. (2009) 'The Use of the Smart Geometry Through Various Design Processes: Using the programming platform (parametric features) and generative components', *Ascaad*, pp. 297–304.

Aish, R. and Woodbury, R. (2005) 'Multi-Level Interaction in Parametric Design', *Lecture Notes in Computer Science*, 5531(August). doi: 10.1007/978-3-642-02115-2.

Andriole, S. J. (1994) 'Fast, cheap requirements prototype, or else!', *IEEE Software*, 11(2), pp. 85–87. doi: 10.1109/52.268964.

Bolchini, D. and Paolini, P. (2002) 'Capturing Web Application Requirements through Goal-Oriented Analysis', *WER*, pp. 16–28.

Bowen, G. A. (2009) 'Document Analysis as a Qualitative Research Method', *Qualitative Research Journal*, 9(2), pp. 27–40. doi: 10.3316/QRJ0902027.

Brinck, T., Gergle, D. and Wood, S. D. (2002) *Usability for the Web: designing Web sites that work*. San Francisco: Morgan Kaufmann Publisher, An Imprint of Elsevier.

Escalona, M. J. and Koch, N. (2004) 'Requirements engineering for web applications – a comparative study', *J. Web Eng.*, 2(3), pp. 193–212.

Escalona, M., Mejías, M. and Torres, J. (2002) 'Methodologies to develop web information systems and comparative analysis', *Upgrade: the European Online Magazine for the Information Technology Professional*, III(3), pp. 25–36.

Eskola, J. (1998) *Johdatus laadulliseen tutkimukseen*. Tampere: Vastapaino.

Ferreira, M. J. and Loucopoulos, P. (2001) 'Organization of analysis patterns for effective reuse', *ICEIS 2001 - Proceedings of the 3rd International Conference on Enterprise Information Systems*, 2(January 2001), pp. 766–773.

Gustavo, A. *et al.* (2004) 'Web services: concepts, architectures and applications'.

Springer Berlin.

Hanus, M. and Kluß, C. (2009) 'Declarative programming of user interfaces', *International Symposium on Practical Aspects of Declarative Languages*, pp. 16–30. doi: 10.1007/978-3-540-92995-6_2.

Helms, J. and Abrams, M. (2008) 'Retrospective on UI description languages, based on eight years' experience with the User Interface Markup Language (UIML)', *International Journal of Web Engineering and Technology*, 4(2), pp. 138–162.

Hernandez, C. R. B. (2006) 'Thinking parametric design: Introducing parametric Gaudi', *Design Studies*, 27(3), pp. 309–324. doi: 10.1016/j.destud.2005.11.006.

Hoffmann, C. M. and Kim, K.-J. (2001) 'Towards valid parametric CAD models', *Computer-Aided Design*. Elsevier, 33(1), pp. 81–90. doi: 10.1016/S0010-4485(00)00073-7.

'IEEE Standard Glossary of Software Engineering Terminology' (1990) *IEEE Std 610.12-1990*, pp. 1–84. doi: 10.1109/IEEESTD.1990.101064.

Johansson, M. and Arvola, M. (2007) 'A case study of how user interface sketches, scenarios and computer prototypes structure stakeholder meetings', *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it*, 1(XXI), pp. 177–184.

Karten, N. (1994) *Managing Expectations*. Dorset House. Available at: https://books.google.fi/books?id=4vMJAQAAMAAJ.

Käpyaho, M. and Kauppinen, M. (2015) 'Agile requirements engineering with prototyping: A case study', *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, pp. 334–343. doi: 10.1109/RE.2015.7320450.

Laine, M. *et al.* (2011) 'Toward unified web application development', *IT Professional*, 13(5), pp. 30–36. doi: 10.1109/MITP.2011.55.

Laine, M., Shestakov, D. and Vuorimaa, P. (2012) 'XFormsDB: An Extensible Web Application Framework Built upon Declarative W3C Standards', *Applied Computing Review*, 12(3), pp. 37–50. doi: 10.1145/2387358.2387361.

Laperdrix, P., Rudametkin, W. and Baudry, B. (2016) 'Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints', *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 878–894. doi:

10.1109/SP.2016.57.

Lowe, D. and Hall, W. (1999) 'Hypermedia and the Web: an Engineering Approach', *John Wiley & Sons Ltd*, p. 599. Available at: http://eprints.ecs.soton.ac.uk/2927/.

Oxman, R. and Gu, N. (2015) 'Theories and Models of Parametric Design Thinking', *Real Time - Proceedings of the 33rd eCAADe Conference Vienna, Austria, 16-18 September*, 2(September), pp. 477–482.

Paterno', F., Santoro, C. and Spano, L. D. (2009) 'MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments', *ACM Transactions on Computer-Human Interaction*, 16(4), pp. 1–30. doi: 10.1145/1614390.1614394.

Rudd, J., Stern, K. and Isensee, S. (1996) 'Low vs. high-fidelity prototyping debate', *Interactions*, 3(1), pp. 76–85. doi: 10.1145/223500.223514.

Römer, A. *et al.* (2001) 'Effort-saving product representations in design—results of a questionnaire survey', *Design Studies*, 22(6), pp. 473–491. doi: https://doi.org/10.1016/S0142-694X(01)00003-5.

Saiedian, H. and Dale, R. (2000) 'Requirements engineering: Making the connection between the software developer and customer', *Information and Software Technology*, 42(6), pp. 419–428. doi: 10.1016/S0950-5849(99)00101-9.

Sawyer, P. and Kontonya, G. (2001) *Software Requirements*, *Guide to the Software Engineering Body of Knowledge - SWEBOK.*

Schmitz, P. (2001) 'The SMIL 2.0 timing and synchronization model: Using time in documents', *Using Time in Documents. Technical Report MSR-TR-2001-01 January*, 2, p. 2001.

Schrage, M. (2004) 'Never go to a client meeting without a prototype [software prototyping]', *IEEE Software*, 21(2), pp. 42–45. doi: 10.1109/MS.2004.1270760.

Schumacher, P. (2008) 'Design Research within the Parametric Paradigme', *Publicado como 'Smart Work–Patrik Schumacher on the growing importance of parametrics' In: RIBA Journal Setembro.*

Sommerville, I. (2010) *Software Engineering*. 9th edn. USA: Addison-Wesley Publishing Company.

Taylor-Powell, E.; Renner, M. (2003) ‘Analyzing Qualitative Data’, *Revista de Administração de Empresas*, 45(2), pp. 34–45. doi: 10.1259/bjr/16316438.

ten Teije, A., van Harmelen, F. and Wielinga, B. (2004) ‘Configuration of Web services as parametric design’, *Engineering Knowledge in the Age of the Semantic Web, Proceedings*, 3257, pp. 321–336.

Toffetti, G. *et al.* (2011) ‘State-of-the-Art and Trends in the Systematic Developmet of Rich Internet Applications’, *Journal of Web Engineering*, 10(1), pp. 70–86.

Vuorimaa, P. *et al.* (2016) ‘Leveraging declarative languages in web application development’, *World Wide Web*, 19(4), pp. 519–543. doi: 10.1007/s11280-015-0339-z.

Walker, M., Takayama, L. and Landay, J. A. (2002) ‘High-Fidelity or Low-Fidelity, Paper or Computer? Choosing Attributes when Testing Web Prototypes’, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 46(5), pp. 661–665. doi: 10.1177/154193120204600513.

Whittemore, R., Chase, S. K. and Mandle, C. L. (2001) ‘Validity in qualitative research’, *Qualitative Health Research*, 11(4), pp. 522–537. doi: 10.1177/104973201129119299.

Woodbury, R. (2010) ‘Elements of parametric design’. Taylor & Francis Group.

Yin, R. K. (1994) ‘Discovering the future of the case study. Method in evaluation research’, *Evaluation practice*. Sage Publications Sage CA: Thousand Oaks, CA, 15(3), pp. 283–290.

# Appendix

## Interviews with sales department

**General questions**

Describe typical customers that we build website for.

How do you find potential customers?

How do you present for the customer what kind of website we can build for them?

**Project specific questions – no deal cases**

What kind of requirements did the customer have at the beginning? How did they describe them?

Was the customer happy with the first version provided? If not, what should have been changed?

Did the customer ask for something we cannot provide? What?

Why did not the customer want to make a deal?

**Project specific questions – deal made**

What kind of requirements did the customer have at the beginning? How did they describe them?

Was the customer happy with the first version provided? If not, what had to be changed?

How did the requirements change during the project?

How many times things needed to be changed to meet the customer requirements? What had to be changed?

Was the customer happy with the final solution? What feedback did they give?

How long did it take to provide the final solution?

Did the customer ask for something we cannot provide? What?

**General questions**

What kind of other requirements the customers have for new website at the beginning? How do they describe them?

How do those requirements change after they receive the first version of the website?

What kind of things customers ask but cannot be delivered by us?

How long does it usually take to deliver the final version that the customer is satisfied with?
How many times does something need to be changed before the customer is happy?

What would make the way we deliver websites better?

What kind of solution would you like to have for delivering full websites to customers?