Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

A TWO-WAY METAMODELING APPROACH IN WEB ENGINEERING

XHEVI QAFMOLLA

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Information Science and Computer Engineering

Supervisor : doc. Ing. Karel Richta, CSc.
Co-supervisor : doc. Ing. Vojtěch Merunka, Ph.D.

Prague, February 2015

Thesis Supervisor:

DOC. ING. KAREL RICHTA, CSC.
Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Thesis Co-supervisor:

DOC. ING. VOJTĚCH MERUNKA, PH.D.
Faculty of Economics and Management
Czech University of Life Sciences Prague
Kamýcká 129
165 21 Prague 6
Czech Republic

Abstract

This dissertation thesis introduces a new model-driven approach in Web engineering that aims to decrease the communication gap between stakeholders during the early phases of the project and reduce the development effort required for the production of the final Web application. The approach has been applied in the domain of content-based Web applications and is supported by related methodology and proof-of-concept tools.

While several model-driven approaches specialized in the discipline of Web engineering have been developed in the last decade, the majority of them is outdated, or is applied under specific conditions in practice, usually involving a high cost of adoption. The main reason is that most of these approaches apply the modeling part in a top-down mode, i.e. they focus on a high-level conceptual model of the Web application upon which later are applied transformation techniques to generate the final components. The drawback of such approach is two-fold. Firstly, communication between stakeholders on a high-level at the early phases of the project is ambiguous, leading to problems and increased overhead effort at later phases. Secondly, transformation of an abstract model is complex and it involves several errors and manual operations. In contrast, our proposal consists of a two-way approach of model creation and transformation. Both technical and non-technical stakeholders use a common low-level conceptual model to fine-tune requirements, such as a rapid prototype of the Web application in HTML or other form (bottom-up phase). After several iterations, once the model is enough mature, it is used as an input for the (semi-)automated creation of the final application's components and its initial content (top-down phase).

The proposed approach has been evaluated qualitatively in real-world case studies and quantitatively through a large number of simulated projects using open-source Web application templates. The achieved results are promising and they show potential improvement in the practical application of model-driven techniques in the Web engineering discipline.

Keywords:
Web Engineering, Content-based Web Applications, Model-driven Development, Code Generation.

Acknowledgements

The completion of this thesis would not have been possible without the help and support of many important people.

First and foremost, the biggest appreciation goes to my supervisor, Doc. Karel Richta and my co-supervisor Doc. Vojtěch Merunka, for their invaluable help, directions and advice in critical moments during the research and for keeping me motivated throughout the years. Thank you to both of you!

Further I would like to thank the opponents of the thesis, for the time they have devoted in the review of my work and helped me finalize this research with success.

To my colleagues and friends from the department of Computer Science and Engineering in the Faculty of Electrical Engineering, Czech Technical University in Prague, who always were there for me, giving objective and constructive feedback for my work – thank you to all of you! Namely to Ing. Nguyen Viet Cuong and Ing. Tomáš Procházka M.A. for your help and inspiration through the years of study – thank you!

Many thanks to the colleagues of O2 Czech Republic a.s., Etnetera a.s., Tinit s.r.o and many other friends and colleagues from the Web community, for helping me review, evaluate and improve my work by providing real-world data and experience.

This work has also been supported by the Department of Computer Science and Engineering, Faculty of Electrical Engineering and by grants of Czech Technical University in Prague.

Dedication

Dedicated to my wife for her endless patience and support of my work . . .

Contents

List of Figures **ix**

List of Tables **xi**

Abbreviations **xii**

1 Introduction **1**
1.1 Web Engineering Perspective 1
1.2 Model-driven Perspective 2
1.3 Web Modeling Discipline 3
1.4 Rationale and Research Questions 3
1.5 Thesis Outcome 4
1.6 Problem Definition 6
1.7 Research Approach 7
1.8 Thesis Outline 8
1.9 Textual Notations 10

2 Background **13**
2.1 Web Engineering Framework 13
2.2 Web Application Characteristics 14
2.3 Web Application Types 16
2.4 Web Application Development Process 17
2.5 Requirements 17
2.6 Modeling in Web Engineering 18
2.7 Usability of Web Applications 18

3 State of the Art and Related Work **21**
3.1 Web Modeling Language 21
3.2 Unified Modeling Language 22
3.3 UML-based Web Engineering 24
3.4 CMS Modeling Language 25
3.5 Web Engineering Method 25
3.6 Other Model-driven Approaches in Web Engineering 28

3.7 Survey on Practical Application of Related Works 28

4 A Two-way Metamodeling Approach 31
4.1 Prototyping the Presentation as a Conceptual Model 31
4.2 A Generic Metamodel for Content-based Web Applications 34
4.3 Semi-automated Generation of the Web Application 36
4.4 Administration of the Content 41
4.5 Limitations 45

5 Case Study 47
5.1 Requirements and Prototype 47
5.2 Generating the CMS from the HTML prototype 51
5.3 Administration of Content 54

6 Evaluation and Results 55
6.1 Qualitative Results 55
6.2 Quantitative Results 59
6.3 Comparative Evaluation 62

7 Conclusions and Discussion 67
7.1 Conclusions 67
7.2 Discussion 68
7.3 Summary 70

Bibliography 73

Publications of the Author 79

A Anotace 83

List of Figures

1.1 Context of the domain of research 5
1.2 Overview of the thesis outcome 6
1.3 Action research method 8

2.1 Highlighted areas of Web engineering framework, by Kappel et al., affected by this research 15
2.2 A typical architecture for Web applications 15
2.3 Categorization of Web applications 16

3.1 Examples of WebML: a) data model, b) hypertext model, c) process model 22
3.2 UML examples of: a) class diagram, b) communication diagram, c) use-case diagram 23
3.3 Overview of UWE models 24
3.4 Abstraction levels and involved user roles in CMS-ML 26
3.5 Approach overview and abstraction levels of WEM 27
3.6 Chronology and maturity of related approaches 28

4.1 The classical process of Web application development (above) vs. our proposed one (below). 33
4.2 Simplified view of main components of the proposed metamodel 35
4.3 Scheme of semi-automated generation of CMS 37
4.4 Algorithm for parsing the input HTML prototype 39
4.5 Parsing demonstration from part of a page 40
4.6 An excerpt of a typical CMS administration interface 41
4.7 Use-case diagram of the administration operations 44

5.1 Structure and content of the initial HTML prototype 49
5.2 Preview and content blocks of the homepage in the HTML prototype . . . 50
5.3 An excerpt of the homepage representation in HTML and “readable” JSON format 52
5.4 Recursive parsing and storing of the file content and regions 52
5.5 Part of a page (above) and its database representation in the CMS (below) 53
5.6 Administration options based on the content type 54

6.1 Graph showing percentual split of templates by page types 60
6.2 Graph of templates and number of manual operations 61
6.3 Total number of manual operations by number of page types and its trendline 62

List of Tables

3.1 Survey results about some MDD-approaches 29

6.1 Results of qualitative evaluation 58
6.2 Comparative evaluation criteria and their support by our and other related approaches and languages 65

Abbreviations

AJAX	Asynchronous JavaScript and XML
CBWA	Content-based Web application
CGA	Completeness of Generated Application *(comparative evaluation criteria)*
CMS	Content Management System
CMS-IL	CMS Intermediate Language
CMS-ML	CMS Modeling Language
CSS	Cascading Style Sheets
DOM	Document Object Model
DM	Support for Domain Modeling *(comparative evaluation criteria)*
HTML	HyperText Markup Language
IT	Information Technology
JSON	JavaScript Object Notation
MAB	Ability to Model Applications Behavior *(comparative evaluation criteria)*
MDD	Model-Driven Development
MOF	Meta-Object Facility
MUI	Modeling of User Interfaces *(comparative evaluation criteria)*
MVC	Model-View-Controller
NPF	Support for Navigation and Page Flow *(comparative evaluation criteria)*
OMG	Object Management Group
OOHDM	Object Oriented Hypermedia Design Method
OOWS	Object Oriented Web Solution
PI	Platform Independency *(comparative evaluation criteria)*
RDBMS	Relational Database Management System
SaaS	Software as a Service
SDLC	Software Development Life-cycle
SQL	Structured Query Language
TM	Transformation of Models *(comparative evaluation criteria)*
UML	Unified Modeling Language
URL	Uniform Resource Locator
UWE	UML-based Web Engineering
WebDSL	Web Domain Specific Language
WebML	Web Modeling Language
WEM	Web Engineering Method
WYSIWYG	What You See Is What You Get
WWW	World Wide Web
XML	Extensible Markup Language

Chapter 1

Introduction

Complexity of software systems has become *de facto* an indispensable issue that engineers have to deal with from the early stages of the software development life cycle (SDLC). The reasons are many: from the demand of higher levels in performance, security, reliability and interoperability to new technologies arising at a fast rate.

With the growth of Internet, Web-based solutions to software problems have become massively popular. Web engineering as a discipline encourages the application of systematic, controlled and quantifiable approaches for the development of high-quality, ubiquitously usable Web-based systems and applications. Therefore an academic approach is at hand when trying to improve the situation. On the other side, Web applications are now already integrated within the commercial environment of small and large scales. Understanding the needs of the users is crucial for an academic approach to be useful in the real-world projects. As such, it is necessary within a study to balance and combine both scientific and practical perspectives. Throughout the thesis we highlight this fact, giving emphasis to the benefits of being driven from an academic perspective, while delivering also practical benefits side to side.

1.1 Web Engineering Perspective

Since the new millennium, designing and maintaining high quality Web applications has been one of the major challenges of the community from different industries. Little emphasis has been put to the *end-to-end* process of what it means to create and maintain high quality Web applications. Instead, solutions are typically built by simply applying requirements at-hand and in an ad-hoc manner. This has lead to frequent problems in usability, maintainability, quality and reliability [16].

Web engineering is a multidisciplinary domain, involving knowledge from many areas, such as system analysis and design, requirements engineering, human-computer interaction and user experience, testing, modeling etc. Touching such a complex set of concepts, makes Web engineering as a discipline almost unique in the perspective of applying common software engineering techniques. From a practical viewpoint, typically, the development of a Web application requires people from these different domains of knowledge, who need to work together with the same artefacts. It is crucial that at any phase of the process all the stakeholders understand properly the requirements and work with the same notations. That's why it is important that everybody “speaks the same language”.

In practice, however, due to different factors, many Web engineering projects fail to meet the business needs, involving delays, exceeded budgets and/or failure to deliver the expected functionalities. It is a similar pattern to the problem of the software crisis that rose in the seventies due to a lack of suitable process models and application architecture constraints [20]. So, addressing these issues with the aim of providing a better *approach* to the discipline in general and to bring improvements in any of its specific areas is a task that we consider and solve within this study.

1.2 Model-driven Perspective

Standalone applications that do not need adaptation, integration and future maintenance is already considered an outdated concept. The Object Management Group (OMG) has helped to reduce the complexity of software applications, lower their costs and speed up their implementation [37]. OMG has founded and defined standards that stress out the importance of a design-first approach in development, separating business and application logics from the underlying platform technology. Such approach is very important for increasing efficiency and quality of software applications and it is today a well established concept, supported by the research community.

OMG highly recommends usage of models as “the core” of design and development process of software applications. This approach is denoted as model-driven development (MDD). Models represent, in this case, a part of the system's functionality, structure, content and/or behavior. Following MDD principles has many tangible benefits in practice, for many reasons. For example, such approach makes it easier to reproduce implementations of the same model and deploy it for different platforms, when they follow the same initial structure.

OMG has defined through the years several standards, languages and technologies suitable for the software modeling domain in general (not only for Web engineering). Namely, the most well-known outcome of these initiatives are the Meta-Object Facility (MOF) standard [33] and its underlying mainstream modeling language called Unified Modeling Language (UML) [51], which is a general-purpose modeling language. Models, however, can be defined in any language as long as they are enough expressive for the given domain.

OMG follows a layered architecture of modeling. This means that depending on the phase of the project or other affecting circumstances, one can use different modeling concepts that may be generic, or specific, depending on the level of knowledge at the given moment. Following OMG and MDD principles is crucial in order to provide a long-lasting methodology, language or approach, regardless of the domain. Our work is heavily based and inspired by the MDD technologies and principles.

1.3 Web Modeling Discipline

Web modeling, as a branch of Web engineering, putting models in the core of systematic Web engineering, has been a topic of research for several years. Existing approaches aim to reduce ad-hoc development of Web-based applications and systems by introducing technology-driven methods, as described previously. By definition, the **Web modeling discipline** focuses on the design notations and visual languages that can be used for the realization of robust, well-structured, usable and maintainable Web applications. Designing and implementing a large-scale Web application usually upholds the approach of using models for various abstractions of the involved aspects. Typically these concepts include information structure, content, navigation and presentation.

However, while MDD has received considerable attention and is well on its way to becoming a promising paradigm in solving difficult problems of software engineering in practice, current Web modeling approaches are still “one step” behind. Some of them have deployed tools and other facilities that assist their concepts. However, they are affected by issues and limitations that are common up to a certain extent, such as supporting specific platforms, only model-to-code transformation capabilities etc. In order to become fully aligned with the MDD paradigm and to be embraced by practical applications, these methodologies require more adaptation and transition support. Some relevant approaches in Web modeling discipline are described in chapter 3 of this thesis (state-of-the-art).

We could generally say that MDD principles and existing Web modeling approaches stand on two planes that are converging slowly. The reason behind this issue is that aligning two different concepts at a high level, even though very similar, is a very complex task. The track of Web modeling is missing the required degree of generality, by focusing mostly on notional aspects and domain-specific concerns for the Web. In practice there is still a large “gap” to be filled and this is one of the contribution that this research has aimed to deliver.

1.4 Rationale and Research Questions

With the rapid expansion of the Internet over the last two decades, the World Wide Web (WWW) and its usage have massively changed. So have changed the needs and requirements for Web applications, which have become a solution for different projects in many industries. Like never before, in the last years there has been such a fast pace in bringing to the community new technologies supporting the development of Web projects [17].

To keep up with these inevitable technological changes several activities have emerged that deal with concerns from the user's and developer's perspectives. Systematic ways and approaches have been introduced as an effective way of creating and managing Web applications. However, as discussed previously, application of these methodologies is not a mainstream in practice. On a large scale, most of the projects are executed in a “quick-and-dirty” way, which leads to a huge fragmentation of solutions and platforms. This happens because application of current methodological approaches is not seamless and it requires specific setup of the team and the development environment. Therefore, studies show that in practice the quality of the projects outcome is usually poor and the lifespan of the products is short and cumbersome [18].

Throughout our research we have explored several areas in the Web engineering domain and analyzed end-to-end the whole SDLC of Web applications, including design, implementation and usage [A.7, A.8]. During the exploratory phase of our research, our work was involved in finding answers to the following research questions:

> **Research Question Q1**
> “What are the present difficulties in the domain of Web engineering today? What are the current obstacles, limitations, drawbacks?”
>
> **Research Question Q2**
> “What is the state of the art in this field? What is the situation in practice? Are current approaches applied properly? Are they effective?”
>
> **Research Question Q3**
> “Can MDD be applied in practice in the domain of Web engineering, and up to what extent? What are the benefits, opportunities, strengths, future advances and promises of such approaches?”

Finding answers to these research questions is a complex task, but it has been a useful prerequisite for the proper definition of the thesis outcome and the benefits that it should bring to the community. An answer to the research questions is presented in chapter 3 (state of the art and related work) and in the discussion section of chapter 7 (conclusion and discussion).

1.5 Thesis Outcome

The motivation of our work is to design MDD-based techniques that, when applied in practice, will improve the processes and outputs in the areas of Web application creation, usage and administration. In previous works we have analyzed relevant MDD approaches and methodologies in Web engineering to understand their strengths and weaknesses and we have built upon lessons learned, as well as practical real-world projects outcome, in which we have been involved [A.7, A.4, A.8].

We have looked at the domain of research from a broader perspective, see figure 1.1. Through this work we have achieved a more specific definition of the problems, which is described in the next section. Our proposed approach, based on a so-called *two-way* metamodeling principle has been applied in the domain of content-based Web applications (CBWA), one of the most important types of Web applications with a wide range of application in the private and corporate environments [A.2]. There is currently to date,

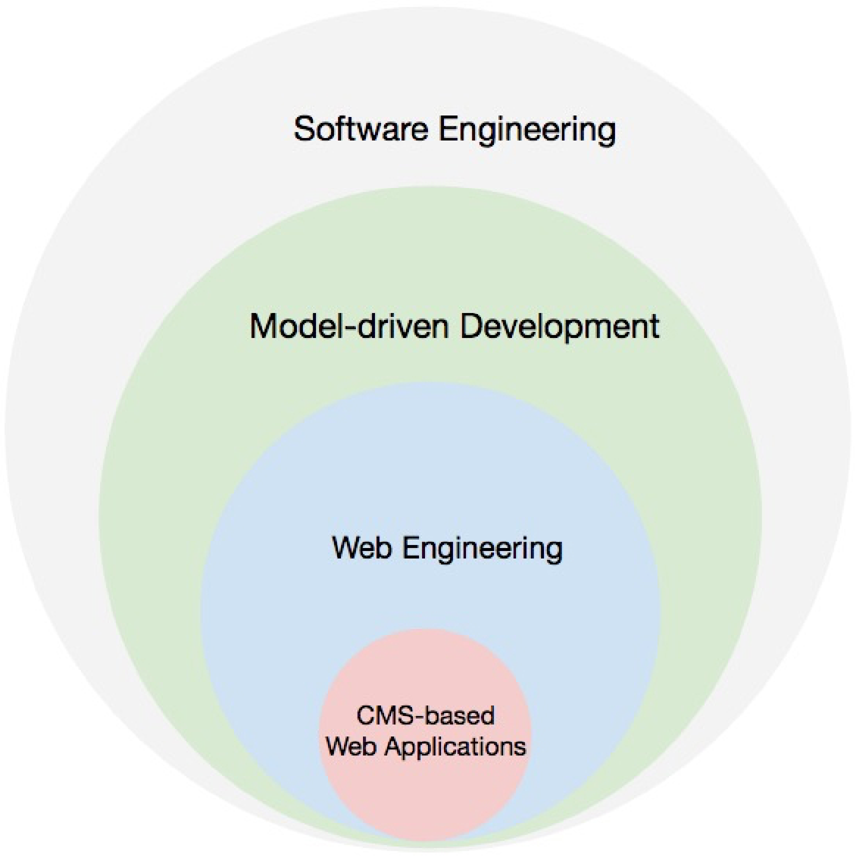


Figure 1.1: Context of the domain of research

approximately 67 millions of CBWA-s active, which accounts for approximately 6% of overall online Web-based solutions in the world [6, 23, 55].

Specifically, the outcome of this work is a methodology for the (semi-)automated generation of a CBWA based on a corresponding metamodel and some tools to support the methodology. Our work is largely based on natural processes, lessons learned, best-practices and patterns from practice. The aim is to make the methodology automated as much as possible (sometimes manual operations are required, therefore it is *semi-*automated), general-purpose, flexible and easy to use, see figure 1.2.

The results achieved in experimental case studies and data from quantitative and qualitative evaluation phases show that application of the approach in practice is beneficial and a further development of the tools in order to allow full integration in the commercial environment can make the proposed approach a mainstream and standard solution, when applicable to the given project.

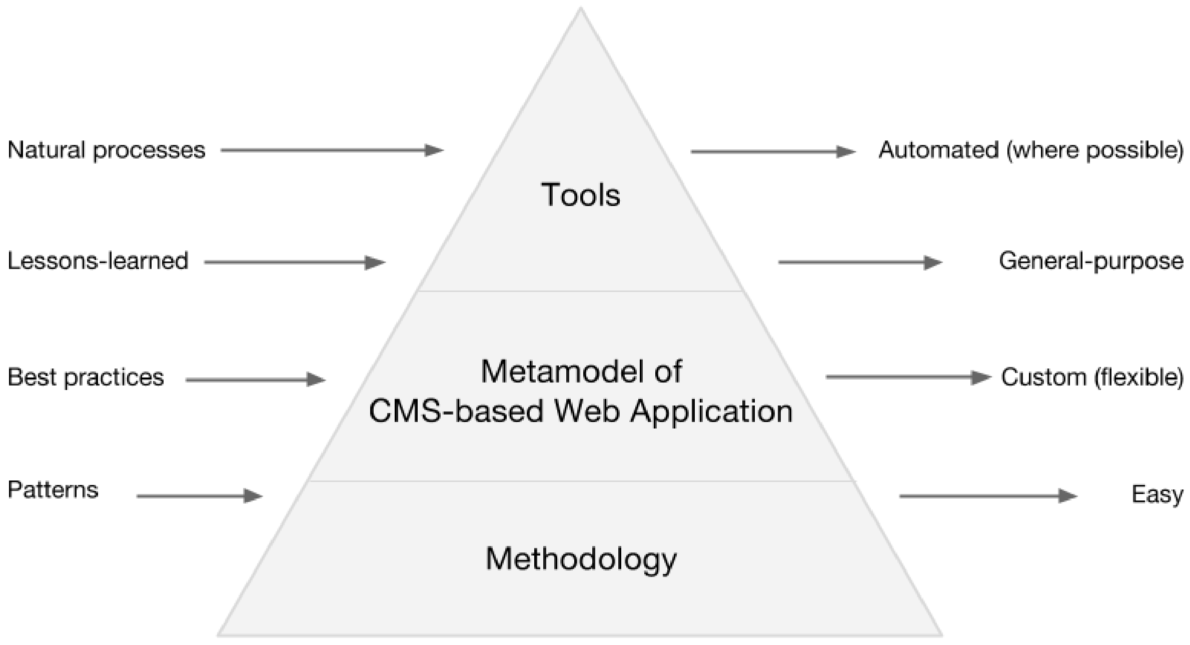


Figure 1.2: Overview of the thesis outcome

1.6 Problem Definition

Currently, there are hundreds of Content Management Systems (CMS) and CMS-based platforms known *publicly* and used for the creation and operation of CBWA [11]. Many others are *custom-developed* and probably little, or not at all known to the community.

The fact that so many CMS platforms exist, indicates that there is a large diversity in the methodologies behind their creation, which leads to fragmentation of solutions. Although the main principle of putting models in the core of the development process is sometimes applied, fragmentation still exists and new CMS-es are still being developed in an ad-hoc way. Moreover, with the introduction of software-as-a-service (SaaS) architectures, CMS became an online tool. This has made the situation even more complicated by introducing a whole new level of concepts and options for Web applications content creation. Fragmentation of platforms is a problem when it comes to making a choice of the system in the real world. Several studies and comparisons have been made between features that these systems provide to the end user [2, 5, 12]. Analyzing these features and making a decision is a hard task that requires a lot of time and resources.

Behind all of this complexity there is a confused user, who has his needs and must make a choice. The choice of the CMS is a decision that will affect them for a long period of time and therefore it is very important. The project owners, who usually have little or no technical experience, are not able to use out-of-the-box solutions without the involvement of a technical person, therefore communication between them is crucial. Another important fact to consider in practice is that even for simple Web applications the corresponding overhead related with the initial platform setup can be huge. These facts lead to increased costs and resource overhead in the project. Typically, the complexity for the development of a Web project involving creation of a content-based Web application varies between a few man-days and several hundreds, depending on its size [32, A.7], while naturally, it is in the benefit of everyone involved to keep these costs down.

Considering these challenges of the Web engineering community, we have defined the following problems around which our research work was carried out and evolved towards the delivery of a solution:

Problem P1
"To define model(s) of the Web application that will allow the users to communicate conceptually on the same level, while also allow automation of some parts of the application development."

Problem P2
"To design a methodology for the creation of a content-based Web applications that will be enough simple for practical usage by the non-technical user, while still enable the technical user to adapt and extend it in a natural way."

The results that we have achieved in our work are promising and they may provide radical improvements to the state of art in the domain, once the supporting tools will be enough mature and integrated in the business environment.

1.7 Research Approach

Given the fast-evolving, utmost nature of Web engineering discipline in general and Web-related technologies in particular, it was important to embrace such an approach for the research, that would allow gradual modification and improvement of the produced outcomes. To find an appropriate approach for this task we have been inspired by the *action research* model and method defined by O'Brien R. [38]. This model allows for a progressive problem solving and reflective feedback process that can be applied in research work, where individuals and/or teams can cooperate effectively. It is a reasonable way of solving a complex and evolving problem through time. This model involves planning, actions- and results-taking in an iterative and feedback-loop way, see figure 1.3 [1].

Such a choice of the research approach has been very important. Throughout the research's life-span, there were several pivoting points, when we had to change the course and look at the presented problems under different decisive criteria, or from another perspective. This research project began with the aim to define a metamodel for the ultimate definition of Web projects (of any kind). Soon we realized that such a scope is very large and complicated and that we needed to focus on a smaller domain. Therefore we decided

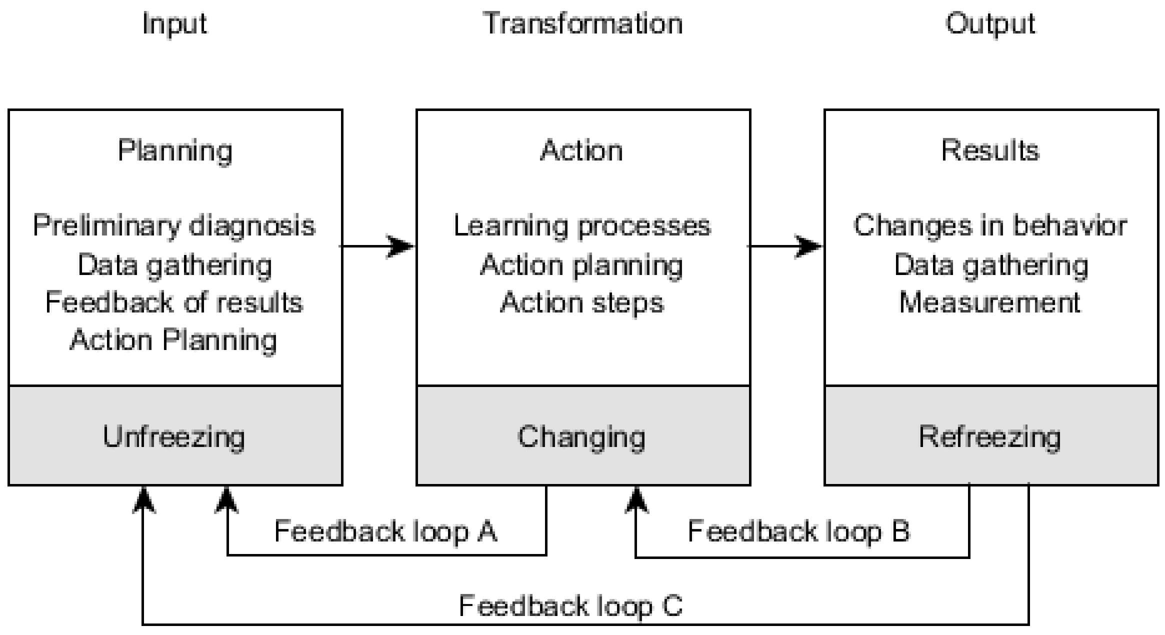


Figure 1.3: Action research method

to focus on CBWA as one of the most widely-spread types of application in general practice. However, we tried to keep the invented principles applicable not only in the CBWA domain, but more widely, whenever possible.

A second pivoting moment during the research was the input from practical, real-world projects, in which we were involved. This gave us some substantial research data to focus on for the analysis of potential pros and cons of existing methodologies, as well as our own.

Finally, we understood that a metamodel for the Web engineering domain is not enough descriptive to be advanced and embraced from other peers in the research community. So we bundled the metamodel within a methodology and developed some proof-of-concept tools to support it. Reaching towards the end of the research period, we evaluate the research approach as crucial for the successful completion of work and achievement of positive results.

1.8 Thesis Outline

In this section we give an outline of the chapters contained in the thesis, for easier understanding of its structure and for a better experience when referencing to the required content.

1. *Introduction*: In the introduction chapter we cover the basic rationales of why this research has started and what it tries to solve. We look at current situation and

needs from the perspectives of Web engineering discipline, the OMG and model-driven development point of view and from an intersection of the two, i.e. the Web modeling domain. Further we outline the initial research questions and the outcome of the thesis, describing also the problem definitions. We also discuss the methodology and phases under which this research was carried out. Finally we give some textual notations that are used throughout the thesis content.

2. *Background*: In the background chapter we describe some fundamental concepts around the Web engineering discipline that are relevant for this research. We highlight characteristics of the Web applications that are important for a further understanding of the proposed approach. We also describe the areas of Web engineering that are impacted by this research, namely the Web development processes, the modeling paradigm in Web engineering, the requirements of Web applications and their usability principles.

3. *State of the Art and Related Work*: In this chapter we present the state of the art in the domain of Web modeling, covering selected approaches and languages that are related with our research work. Namely, we discuss these approaches by giving a definition of their fundamental concepts and examples of their corresponding notions. We also discuss the most important strengths and weaknesses for each approach. A more in-depth evaluation, including a comparison between these approaches and our proposal, is given later in chapter 6 (evaluation and results).

4. *A Two-way Metamodeling Approach*: Chapter 4 covers the main results of our research work and its outcome. We outline the results from a high-level perspective and then go in-depth through the definition and explanation of each part of the approach. We depict in each section the necessary specifications, however for an easier understanding of the approach in this chapter, we only provide brief specification and cover application details and information in the chapter 5 (case study), through an example. In the last section of the chapter we point out also known limitations of our proposed approach and also discuss them later in chapter 7 (conclusions and discussion).

5. *Case Study*: In this chapter we demonstrate the most important concepts of the proposed approach through an example. This aims to give the audience a more concrete understanding of the approach and its characteristics that were described in the previous chapter. In the given example we omit some technical and implementation details in order to better focus on the fundamental aspects of the approach, therefore keeping the scope of this chapter as clear as possible for audiences with a different level of technical understanding and backgrounds.

6. *Evaluation and Results*: In chapter 6 we present the evaluation process and the results achieved by our work. The evaluation process covers three phases. The first one is the qualitative evaluation phase, in which the aim was to gain general feedback

and conclusions about overall performance criteria for our proposed approach. The second phase is the qualitative evaluation. This was done with the goal of assessing the applicability of the (semi-)automated generation of the CMS based on ready-made templates. The focus of this phase was the evaluation of the concept and the related tools. Finally, an assessment of the overall approach according to evaluation criteria from related studies and works is presented. This is done for the purpose of comparing our approach with similar other approaches in the same domain (where applicable). The focus of this part is to distinguish on a theoretical level the cases, *where* the methodology would be beneficial and *under which circumstances*, regarding the stakeholder's needs and requirements.

7. *Conclusions and Discussion*: In the last chapter we conclude with a summarization of the work, its outcome and the results. We highlight once again the definition of the problems that this thesis has dealt with and recapitulate the main achieved principles and solutions. We briefly describe the most important conclusions achieved through this work. Further, we discuss some aspects of the proposed approach with regards to their strengths and weaknesses and give some rationale behind these facts. Finally, we discuss here some proposal and inspirative questions for future works in the area of Web engineering and related model-driven methodologies development.

1.9 Textual Notations

For the purpose of distinguishing regular text from important notations and definitions and in order to give a clear context to the meaning of some sentences and terminology, we describe below some typographical conventions that are used throughout the text of this thesis:

- To distinguish the definition of an important term from its normal usage in text, the term is written using **bold** fonts.

  Example:

  > As defined by Kappel et al. a **Web application** is a software system that provides Web specific resources (such as content, or other services) through a user interface, typically as a Web browser.

- To highlight the focus to a term within a sentence we have used *italic* fonts. Italic fonts are also used to distinguish *foreign* terminology.

  Examples:

  > Little emphasis has been put to the *end-to-end process* of what it means to

create such types of Web applications.

Complexity of software systems has become *de facto* an indispensable issue that engineers have to deal with from the early stages of the software development life cycle.

- Code snippets (including pseudo-code), are written using the special `Consolas` type font.

  Example:

  ```
  "UL"
       "class" => "custom-menu"
       "id" => "cm"
       "content" => "DuplicateRemove"
       "LI"
            "data-action" => "rem"
            "content" => "Remove"
  ```

- Examples, *exact* mentions from a referenced source and *non-literal* meaning of notations are quoted inside double quotation marks (“”).

  Examples:

  Although WebML has been relatively long in the scene of research and development of modeling Web applications it still lacks support for some important concepts, for example “user interfaces”.

  Our motto is: “Deliver an unforgettable event of your life.”

  It is important that everybody “speaks the same language”.

Chapter 2

Background

In this chapter we outline some basic theoretical concepts and areas from the Web engineering domain that are necessary for a proper understanding of this research. We explore these background areas from several perspectives, in order to give a broader overview of the relevant concepts and the related literature.

2.1 Web Engineering Framework

As it was discussed in the introduction chapter, the complexity of Web applications in the commercial field has grown dramatically through time. In the beginning of the new millenium, Web was still mainly in the form of static documents that were created and maintained manually. Through time it became more interactive. Later, workflow-based, collaborative, and transactional Web applications were introduced and in the last few years, social, semantic and device-specific characteristics of Web started to grow. Each of these properties and characteristics has basically impacted the way Web applications are created and used. In order to put such a large area of research under a formal perspective, we have chosen the systematic framework of Web engineering projects development, defined by Kappel et al. [26]. This framework is one of the most comprehensive and widely accepted from the Web engineering community. According to this study the Web engineering (as a discipline) can be organized into these three main pillars: approach, product development and quality, each of them containing several sub-areas, see figure 2.1 (page 15) for a full picture.

Our study has focused on improving the state-of-the-art situation in some aspects of the following areas of the Web engineering discipline: processes, requirements, modeling and usability. For the scope of this research these areas have been chosen, because our main aim is to propose a methodology for general usage *in practice* and these areas are those with the highest potential benefits in the successful adoption of the methodology in real-world projects. The other areas, while also important and fundamental for the proper systematic development of a project, are not of a direct concern in our research and the proposed approach.

2.2 Web Application Characteristics

By definition a **Web application** is a software system that provides Web specific resources (such as content and other services) through a user interface, typically a Web browser [26]. The reference to the term Web application throughout this thesis corresponds to this definition as well.

Web applications have several characteristics and perspectives from which one can look into them. Categorization of these characteristics is done into the following four groups, following the ISO/IEC 9126-1 standard for the evaluation of software quality [24]:

Product-related characteristics: these are considered the main building blocks of every Web application. Namely, the product-related characteristics include: content, navigation (also denoted in literature as hypertext), structure and presentation. Each of these blocks is considered from a static as well as behavioral (dynamic) perspective.

Usage-related characteristics: these characteristics cover the social (user-centric), technical and physical context of the Web application. Under the social aspect we consider the cultural background of the end users of the application. The technical aspect deals with the technological properties (software and hardware) of the devices involved in providing of the access to the Web application. The physical context concerns time and location parameters of the application.

Development-related characteristics: from a development perspective Web applications involve several characteristics related to the *resources* that are necessary for its creation and operation. These are factors involving people (development), infrastructure, process and any possible integration with existing system.

Evolution-related characteristics: evolution is a dimension that goes across all three above-mentioned groups of characteristics. This is an attribute that comes due to conditions and requirements of Web applications that involve continuous change of requirements, a fast pace of development and a high level of pressure in terms of time-to-market necessity.

From an architectural standpoint a Web application can be composed in various ways. Usually, the way the architecture is built, is affected by the purpose of the application and its performance parameters, regarding its reliability and scale. A comprehensive study of the applicable architectures in a Web application is discussed by Eichinger in the “Web Application Architectures” chapter in Kappel et al. [26]. However, for simplification purposes we have defined in figure 2.2 a typical architecture, which is commonly applicable for most types of Web applications in general and content-based Web applications in particular. The methodology and tools discussed later in this thesis will also conform to this architecture.

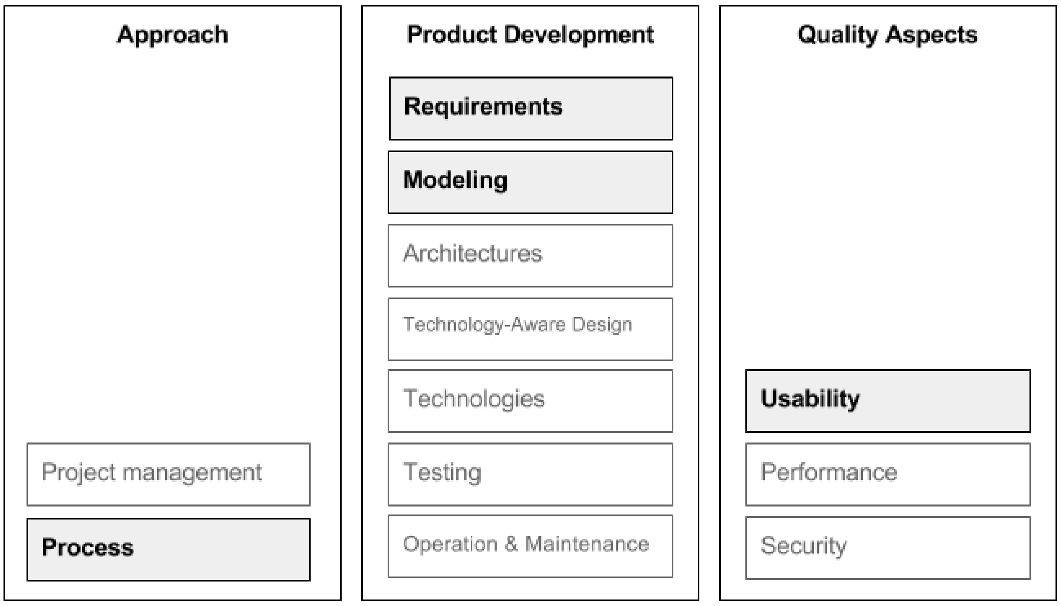


Figure 2.1: Highlighted areas of Web engineering framework, by Kappel et al., affected by this research

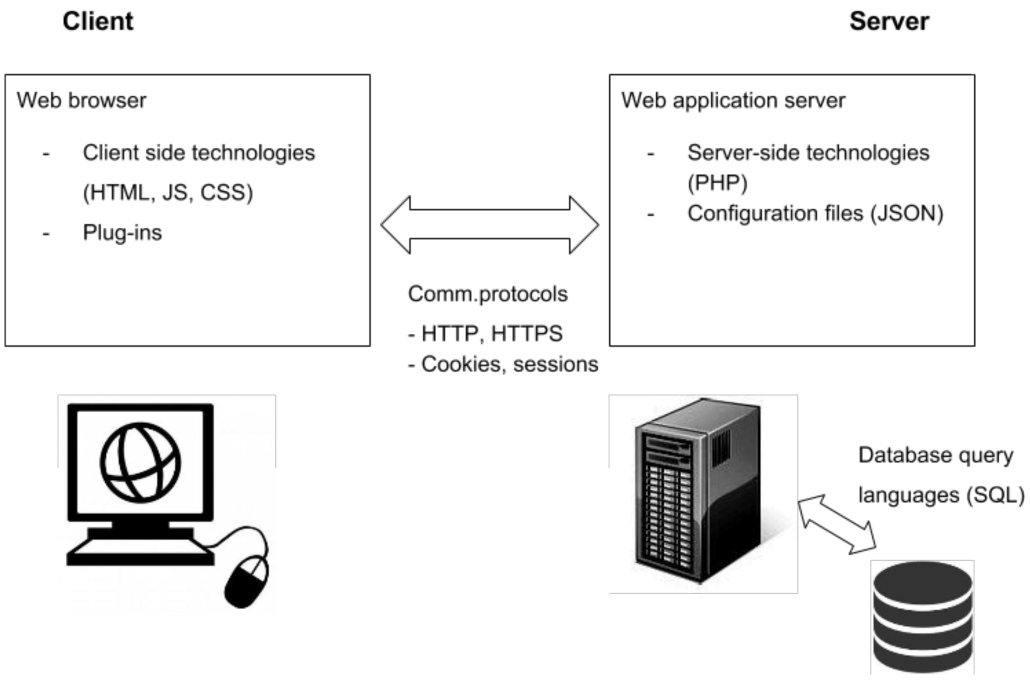

Client
Server
Web browser
- Client side technologies (HTML, JS, CSS)
- Plug-ins
Web application server
- Server-side technologies (PHP)
- Configuration files (JSON)
Comm.protocols
- HTTP, HTTPS
- Cookies, sessions
Database query languages (SQL)

Figure 2.2: A typical architecture for Web applications

2.3 Web Application Types

Based on their attributes and previously mentioned characteristics, Web applications can be categorized into several groups. This categorization has been a subject of several studies and surveys [13, 40], which have achieved quite similar conclusions. The categorization looks at the properties of the Web applications from a time versus complexity perspective, see figure 2.3 [26].

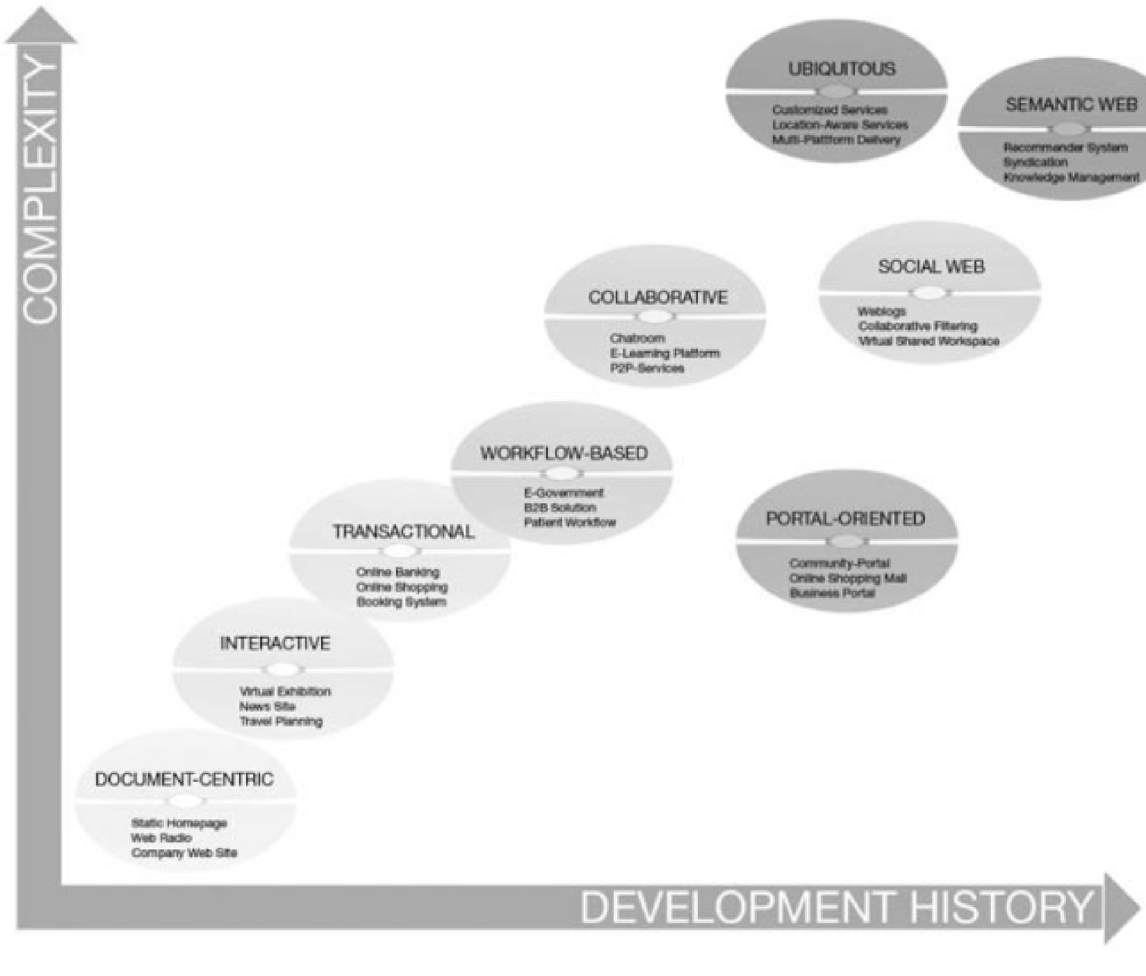


Figure 2.3: Categorization of Web applications

As it was described in the introduction chapter, this thesis focuses on the domain of Content-based Web applications. A content-based Web application is an online application that enables creation, publication and administration of Web pages through a central interface, usually labeled as a content management system (CMS). By this definition and taking in consideration the product-related characteristics described previously, it is clear that such type of application is an intersection of more Web application groups. This is because any type of Web application has a presentation part that shows some content. The differences consist in the degree of complexity of this content and the resources that are used to generate and maintain this content. In the research phases we have given particularly focus to the fact that more types of applications are covered in the application of methodology and tools involved, hence making the results and conclusions as much general as possible. Assumption and limitations are also pointed out accordingly.

2.4 Web Application Development Process

The process under which the Web application is developed is of extreme importance. From a development process perspective, the Web engineering discipline has learned from classical software development. General best practices in applied processes to the classical software development have been adapted for the specific needs of Web applications, since several ago [56]. Processes are usually formalized and they have a specific set of steps that needs to be followed in order to produce the outcome. However, empirical studies show that development time of a Web application is usually short and in cycles [31]. For this reason, heavily formalized processes are usually not appropriate in practice as they increase the overhead and required resources for completing a project successfully. In the modern era of Web development, processes tend to be iterative and agile, in order to support adaptivity and changes. This has proven to deliver good results in practice [41].

Therefore our proposed approach, methodology and its supporting tools have been developed to allow and support agile techniques and quick iterative development, as one of the key performance factors for a successful execution of the Web project.

2.5 Requirements

Requirements are also a critical activity for the creation of a Web application, mainly because they define the final outcome and also set up the criteria for its success. Requirements definition is a complex activity, because it involves *human* input. This leaves room for ambiguity, incompleteness and errors. There is a large “gap” in communication between stakeholders with a different background, typically technical versus non-technical. Usually non-technical stakeholders (business people) are the project owners and sponsors, while technical stakeholders (IT people) are responsible for the project's execution and realization. Even though several methodologies exist for requirement gathering and engineering, practice shows that the current processes in the Web engineering domain are insufficient [4, 36]. Bad requirement definition leads to increased costs and overall excessive overhead in the later phases of the project.

From a classification point of view requirements can relate to any of the areas of the Web application, for example they can involve “technical aspects”, “architecture”, “security” etc. Regardless of the type, however, the process behind requirement definition, evaluation and implementation is crucial. Our research focuses on improving the definition process of explicit requirements related to the Web application. The most important factor for the successful management of requirements is an effective communication and understanding by all the involved parties. It is important to minimize the “gap” that exists between two or more parties that deal with different aspects of the same thing [54]. The two main principles in achieving this are: 1) communication on the same level of abstraction and sharing of the same context between the involved parties, and 2) iterative approach of requirement definition and execution. Our approach is based on these two principles as it is highlighted in chapter 4 (a two-way metamodeling approach).

2.6 Modeling in Web Engineering

The history of the classical software engineering discipline has shown that ad-hoc development of applications is bad in practice [8]. Web engineering is a newer discipline than that of developing classical software; therefore the application of modeling principles in Web projects is still not as wide-spread. Capturing concepts by means of models, however, is something that should be fundamental for the achievement of successful and long-lasting Web applications.

Several initiatives have emerged in the past years that focus on modeling specific concepts of Web applications [A.8, 34]. They define methodologies and tools usually based on a domain-specific approach for the Web as opposed to using standards, such as UML. This, however, leads to a high cost and effort of adoption in practice. On the other hand general-purpose approaches are easier to adopt but they lack the strength and expressiveness of a domain-specific one. Therefore, typically some superstructural concepts are added on top of the basic ones, leading again to increased costs and effort for practical applications. Several of the model-driven Web engineering methodologies have been discontinued. Some serve only as a technical reference. Just a few have been used in practice, mostly in the academic area or in large-scale companies that can afford to support experimental research and development. We outline some relevant modeling approaches and languages in chapter 3 (state of the art and related work).

Our approach is fundamentally based on the principle of modeling, which has been applied since the early stages of the research. We aim to use seamless transformation between models following a natural process, based on practical experience. Artefacts that are integral part of a Web project, namely interactive prototypes and a meta-model, based on the most common content-based Web platforms, are the two basic parts of the proposed methodology as will be described in the main part of the thesis.

2.7 Usability of Web Applications

Considering the *quality* aspect of the Web engineering discipline, usability is a crucial parameter that impacts the ability of a user to complete his goals. Usability is defined as the ease of use and learnability of a product, in our case, the Web application. The three main criteria of the usability are: efficiency, effectiveness and satisfaction. These criteria have been analyzed and specified in more detail in several studies [35, 47]. Several studies have shown that a big amount of productive time a user spends with the Web application is lost due to usability reasons [30].

One important factor that distinguishes Web applications from the classical software is that you cannot train the users of the Web application. Web applications are delivered and used remotely, they are not sold in boxes that come with a manual for the reader. Moreover, learning a new system is an effort that nowadays everybody tries to reduce or overcome, so even if some instructions are given, they need to be simple, brief and easy to remember. The right principle to approach this thought is to build applications that require as little as possible learning of a new system, i.e. they are heavily based on the users intuition and expectations. It is best practice to build applications in the way that users think and react and to allow them to achieve the tasks they are trying to complete with a minimal effort.

Usability is usually considered a metric of the final product. But usability is something that affects also the process behind Web application development and maintenance. In our study we consider usability as one of the important factors determining the application of the methodology in practice therefore the methodology has been designed to achieve a high degree of usability in practice. As is explained later and was backed up by the quantitative evaluation data, the proposed methodology follows a natural processes for the administration of the Web application and it is comprehensible and easy to apply in practice without the need of a long learning curve, or past experience with the involved technical concepts.

Chapter 3

State of the Art and Related Work

Many researchers are working on the Web engineering discipline by approaching Web application development from a model-driven perspective. We outline here the most relevant ones for our work. This chapter gives also comprehensive answers to the research questions **Q1** and **Q2** and a partial answer to question **Q3** (which is addressed again later in chapter 7, conclusions and discussion).

3.1 Web Modeling Language

Ceri et al. describe the Web Modeling Language (WebML) in [10]. It is a notation language for dealing with the definition and specification of Web applications on a conceptual level. WebML consists of seven phases: requirements specification, data design, hypertext design, architecture design, implementation, testing and evaluation, maintenance and evolution. These phases conform to the Web engineering discipline principles discussed in the previous chapter. The method is supported by WebRatio [21], which is a tool that provides modeling and generation of Web applications by means of the WebML. WebRatio is still actively being developed, but it is commercial, so it disables the possibility for its extension and usage from the open-source community of developers and a more transparent understanding of the involved concepts.

One of the main differences between the WebML and our methodology is that WebML uses separate models for different aspect and phases of the Web project. Specifically it uses models for the data, hypertext, and presentation aspects, see figure 3.1 [49]. Moreover for the presentation aspect it does not introduce any specific notation for its expression in the conceptual level, which would be specific for the methodology.

Although WebML has been relatively long in the scene of research and development of modeling Web applications [9] it still lacks support for some important concepts, for example "user interfaces" [58].

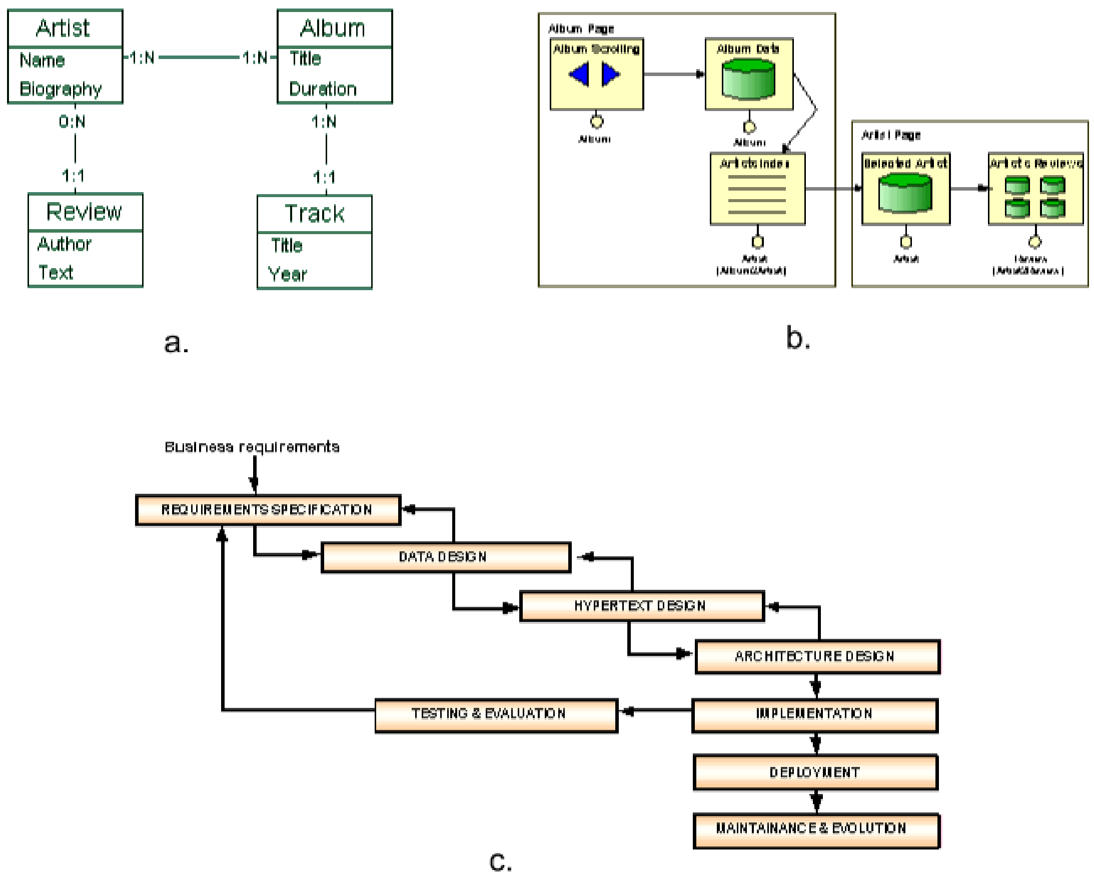
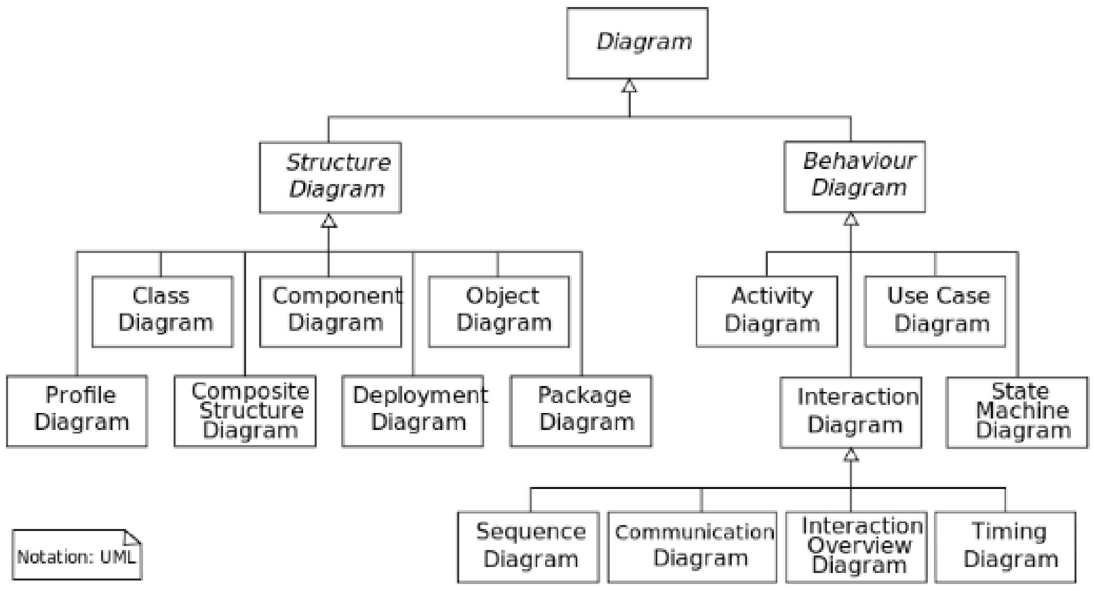


Figure 3.1: Examples of WebML: a) data model, b) hypertext model, c) process model

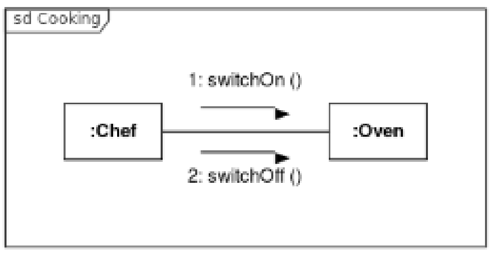
3.2 Unified Modeling Language

The Unified Modeling Language (UML) is a general-purpose language that is commonly used for analysing, modeling and implementing different systems and processes in software engineering [51]. UML is quite fully-fledged and flexible, however this same fact impacts also its applicability in the Web engineering discipline. Models in UML are typically informal and they require the user to have deeper understanding of the domain. Moreover, UML lacks any formality in its semantics and implementation rules, which makes it harder to transform UML-based instance models into functional assets.

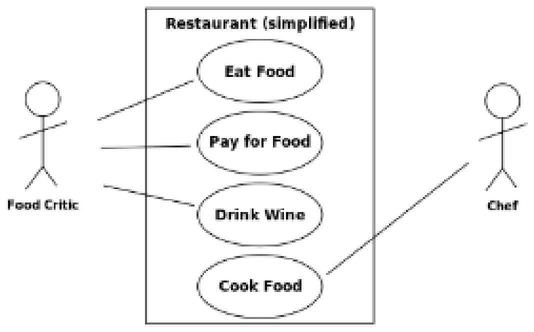
Models in UML are represented by one, or more diagrams. These diagrams can represent two types of views or representation of the system, namely structural or behavioral, see figure 3.2 [52]. Structural diagrams focus on describing a static structure of the system using objects, attributes, operations and relationships. Behavioral diagrams show a dynamic representation of that system in terms of collaboration and changes based on time and other impacting factors.



a.



b.



c.

Figure 3.2: UML examples of: a) class diagram, b) communication diagram, c) use-case diagram

Typically, domain-specific concepts are embodied into UML using some extension mechanisms. These mechanisms allow additional terminology to be provided on top of *core* UML models. Grouping and implementation of a specific set of these extensions is represented by a UML profile. The drawback of such approach is that the profiles are only additional and they cannot modify existing UML core notations, without creating any their derivatives first. This adds to the complexity and overhead of the approach, when applied in practice.

3.3 UML-based Web Engineering

Koch et al. describe a UML-based approach of modeling Web applications called UML-based Web Engineering (UWE) [27]. UWE has been developed with the aim to provide a modeling language, specific for the Web domain, based on UML. It consists of the notations as well as a process to support the development of Web applications in general using a model-driven approach, see figure 3.3 [50]. The most practical benefit of UWE, is that it provides tools (MagicUWE, MagicDraw) that assist the stakeholders during the process of model creation and modification. Some of these tools are automatic. MagicDraw [29] is a commercial-licensed software used for modeling in UML-based notation different aspects of software. MagicUWE [53] is a plugin for MagicDraw. Models developed in UWE are usually simple and easy to understand but less efficient, when it comes to transformation techniques, although automated generation of Web applications is also one of the goals of UWE approach.

UWE follows an iterative and incremental approach for the development of Web applications. This approach, however, focuses mainly on customized and adaptive systems, without dealing much with the content management activity therefore it is only related with our research remotely, because it is also a model-driven approach. However, UWE is open-source and it is highly driven by OMG standards [28]. This makes it a promising candidate for a wider practical application in the future, once it achieves a certain maturity.

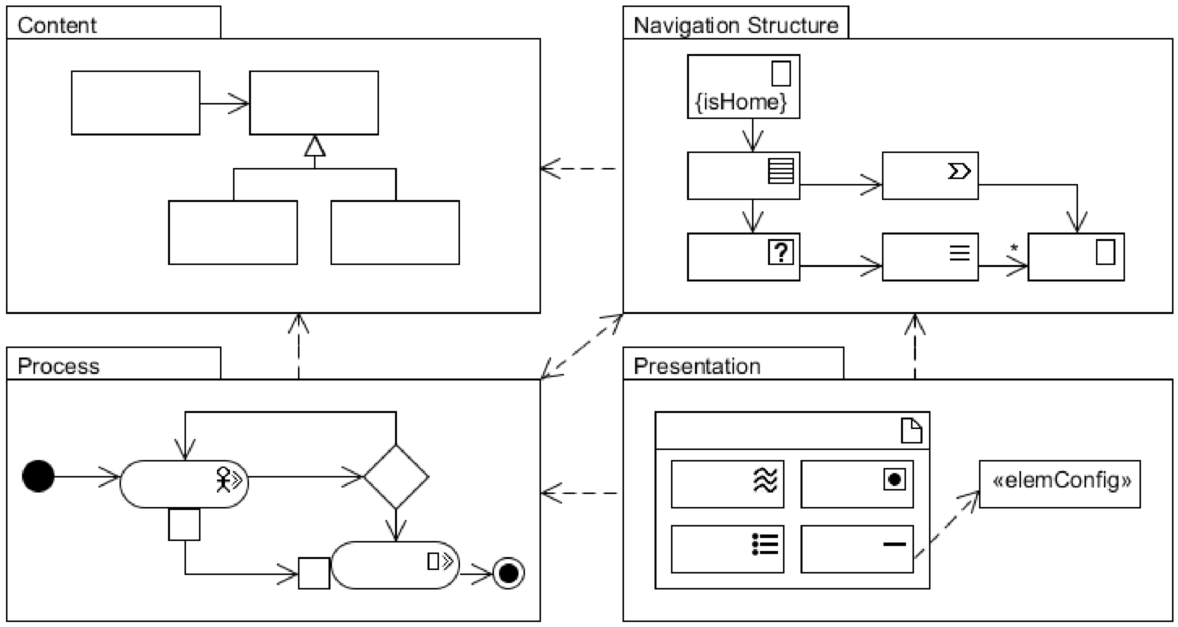


Figure 3.3: Overview of UWE models

3.4 CMS Modeling Language

Saraiva et al. propose a *graphical* language for rapidly modeling Web sites. It is called CMS Modeling Language [44]. The CMS Modeling Language (CMS-ML) is a graphical-oriented language based on models for the high-level definition of CMS-based Web applications. It aims to facilitate the view of the non-technical stakeholders to the Web application's model as well as allow him to perform simple modifications of the model on a conceptual level.

The language consists of two levels: The first level is for the System Designer (technical users), i.e. the CMS Intermediate Language (CMS-IL), which is a set of common low-level concepts for the CMS platform. The second level is for the Business Designer (non-technical user), i.e. the CMS Modeling Language and it provides a set of elements used to quickly model a typical Web application. The model provided by the Business Designer is transformed to the CMS-IL model and then refined and deployed by the System Designer.

CMS Modeling Language introduces several elements necessary for the creation of a Web application. These elements are generic and are modified at the later phases of the project in the form of a Web site template. Models of the language are based on different conceptual models, which makes the communication between stakholders harder, as compared to our proposed approach. Figure 3.4 [44] shows the levels under which the language and its elements operate (see page 26).

3.5 Web Engineering Method

Souer et al. present the Web Engineering Method (WEM), which is a method covering specification, design, implementation and maintenance of CBWA [48]. The method focuses on reducing the “gap” between the business requirements and the resulting Web application's artefacts by reducing the complexity of the application's model. Similar to other related methodologies WEM enables the non-technical stakeholder to make configuration of the application with little or no need of intervention from the technical stakeholder using both abstract and concrete (specific) syntax notations, depending on the stage of the project. The concrete syntax is used for a representation model on the instance level, while the abstract one corresponds to the implementation model and its formalization as a domain model level, see figure 3.5 [48] (page 27).

WEM consists of a modeling tool for the automated configuration of the CMS. The CMS itself is not produced, however. The approach combines principles of some existing methods and enriches them by an abstract and concrete syntax based on a domain model and end-user analysis. As a result it promises to reduce the complexity of implementation of the Web application. The structure and elements of WEM are rather complicated, leading to a higher effort for its adoption in practice. Also its interoperability with general purpose Web-based languages is limited making the transformation part harder. However, some evaluation projects have been carried on using WEM and the results presented are also promising.

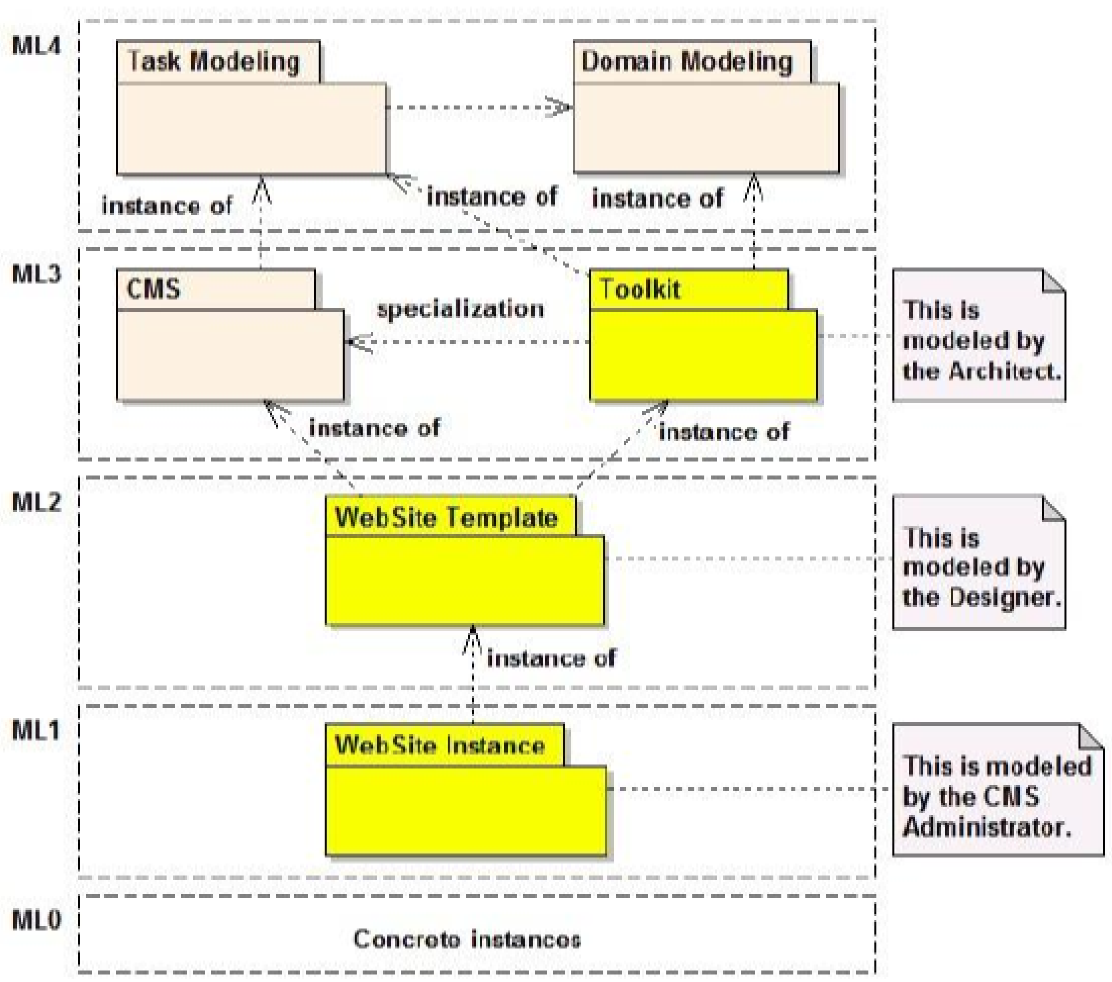


Figure 3.4: Abstraction levels and involved user roles in CMS-ML

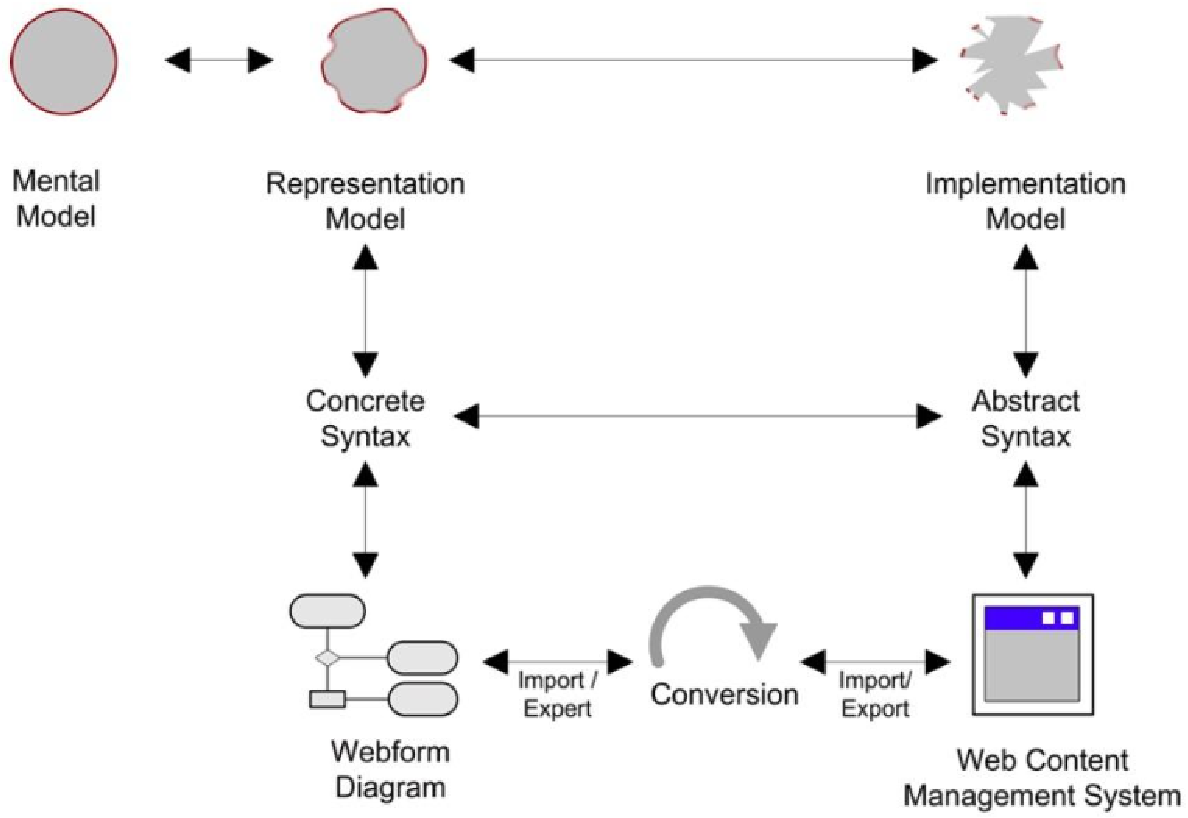


Figure 3.5: Approach overview and abstraction levels of WEM

3.6 Other Model-driven Approaches in Web Engineering

Other less well-known, model-driven approaches in Web engineering include OOHDM [45], OOWS [39], WebDSL [14] and a few others [A.8]. Most of these approaches are discontinued for several years already, or they are not directly related with work from our research, therefore we have only included them in the references for the audience to study, where appropriate. Below we show a diagrammatic representation of chronology in which the mentioned approaches and languages have been created, see figure 3.6. The size of the boxes in the chart represents the approach's "reputation" in the Web engineering community.

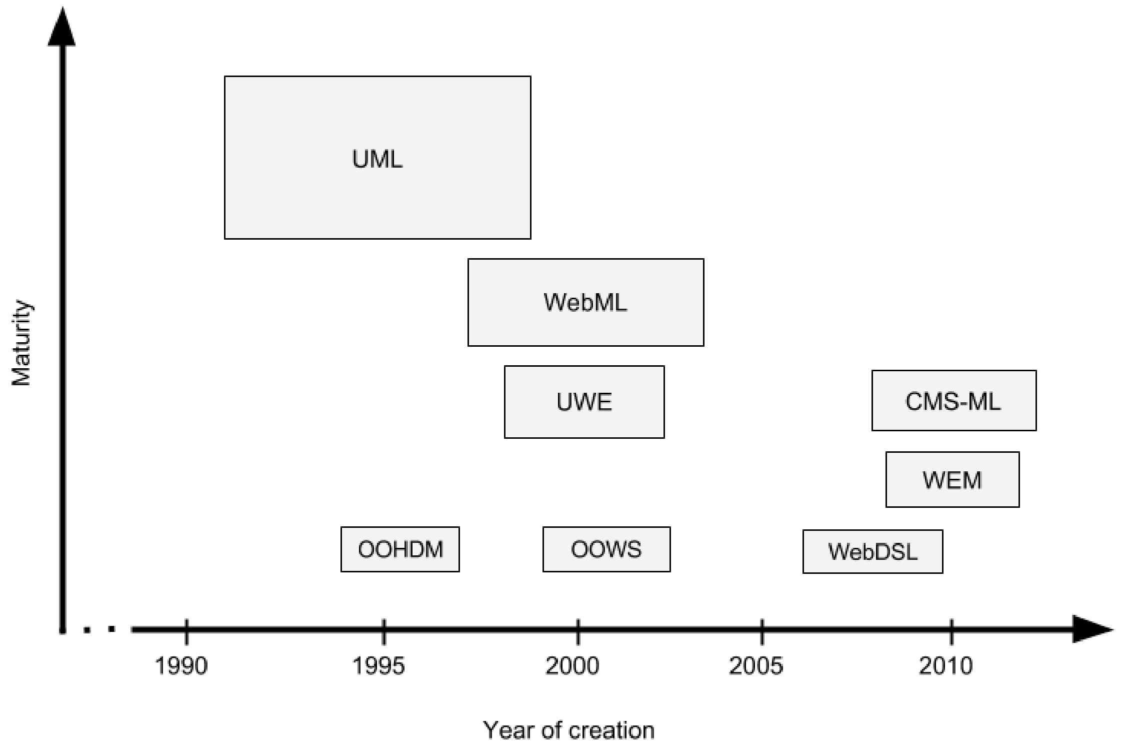


Figure 3.6: Chronology and maturity of related approaches

3.7 Survey on Practical Application of Related Works

The previously described approaches have been analyzed and evaluated in several studies [A.8, 58]. The main drawback is in the limited support of already standardized Web applications concepts that are commonly used in Web projects and that are developed without following any formal methodology or process. This leads to a considerable "gap" between the modeling concepts and the technologies required to support their implementation in practice. Following up on this fact we conducted an informal survey with peers outside the academic area to evaluate the real-world situation. A total of twenty respondents from Web development community in the Czech republic (involved in real-world projects with several agencies) were asked to answer three questions for each chosen modeling approach described above. The questions investigated, whether the subjects knew the approach and, whether they had applied or used them before in some projects.

Table 3.1: Survey results about some MDD-approaches

Approach	**Knowledge**	**Application**
Web Modeling Language	5	0
Unified Modeling Language	18	4
UML-based Web Engineering	3	0
CMS Modeling Language	1	0
Web Engineering Method	0	0
Others (OOHDM, OOWS, WebDSL)	2	0

The results confirm that the previously discussed approaches are rather academic and they have not yet widely been applied in the commercial field, see table 3.1. With regards to application in practice only UML has had some real-world application with 20% of respondents giving a positive answer to this question. The other approaches had no practical involvement whatsoever. Although, with regards to knowledge about these methodologies and their tools, the situation is better. UML is the most well-known language in Web engineering related projects with 90% of the respondents giving a positive answer to that question. Other languages and approaches are considerably less well-known. However, the purpose of this survey is *not* to give a statistical conclusion of the popularity of the related works, but rather a general idea about their application in real-world projects. Therefore we do not argue about the numbers being different in another country, or environment, especially in the countries from which the approaches have originated and where the development community of these approaches is more active.

Chapter 4

A Two-way Metamodeling Approach

The main result of this research is the proposal of a two-way metamodel-based approach in Web engineering for the (semi-)automated creation of the content-based Web applications. The approach consists of three parts:

The first part concerns the requirements-gathering phase. We propose the usage of a presentation prototype as the conceptual model of the CBWA in order to facilitate effective communication between technical and non-technical stakeholders during the preparation and finalization of functional requirements (bottom-up phase). In most cases this presentation prototype is represented in Hypertext Markup Language (HTML), but it can be practically any arbitrary interactive prototype representing the final Web application (for example wireframes exported in the XML format, or any other parsable language).

The second part deals with the application logic, more specifically in the case of a CBWA with the CMS. We propose a simplified metamodel and a technique for generating the CMS automatically by parsing the HTML prototype, which was the output of the first phase (requirement gathering). We achieve this by using some predefined heuristic rules and reverse engineering principles coming as a result of analyzing common CMS solutions and real-world projects (top-down phase).

The third part deals with the administration of content in the future, more specifically with the effectivity and usability of this process. We propose the usage of the Web application's presentation as the CMS interface for the most common operations of content administration such as duplication, removal and editing of content assets.

4.1 Prototyping the Presentation as a Conceptual Model

The requirements-gathering and specification phase involves communication between technical and non-technical users. Practice shows that non-technical users relate more to the Web application's visual presentation, while the technical users think more in terms of functionalities and their programmatic representation. This fact has inspired us to find a solution that would bring these two concepts as close together as possible. In figure 4.1

(page 33) we show in a schematic way the principle of our proposed approach, as opposed to the typical one in Web engineering.

It is standard in Web engineering projects to prepare a prototype of the final Web application at a certain point in time. The abstraction level of this prototype is relatively low as it resembles maximally the final output and it decreases through time, with later iterations. The fidelity (level of detail) of the initial prototype in practice is arbitrary and it ranges from very low, when prototyping tools such as wireframes are used, to very high, when pixel-perfect graphics are embedded and the prototype is in the form of HTML templates. We propose to use the HTML prototype as the conceptual model for the requirements-gathering and specification phase. More precisely, in terms of MDD, this HTML prototype serves as the hypertext and presentation model of the Web application. The *hypertext* model describes the structure and the *presentation* model describes the interfaces and layout of the Web application. The *content* model representing the data is automatically generated and modified through time, as we will describe in the latter chapters of this thesis.

The HTML prototype is usually developed by the technical user. It represents the final product but does not contain yet real data (content) and it is not yet complete. However, it is not a temporary product, as would be a conceptual model prepared in a domain-specific language, or UML just for the need of communication between the non-technical and technical user. In practice, non-technical users that deal with Web engineering projects have a considerable understanding of HTML language and other related technologies behind the creation of the Web application's prototype. This maximally reduces the “gap” between the involved parties and in certain cases enables the non-technical user to directly modify the prototype to fit his conceptual model.

The HTML prototype is easy to create and cheap to modify, therefore it supports agile and iterative development methodologies. The iterative approach allows a gradual clarification of stakeholder's needs. Through iterations the stakeholders are able to specify and finalize functional requirements. Non-functional requirements can be covered by other means that are appropriate for the given project. Usually, however, because Web-related requirements are not very formal they can be also represented through a prototype (at least partially). The cost of adoption and application of this method in practice is minimal, because it does not involve introduction of a new language, framework or other concepts for the purpose of the requirement gathering and finalization phase.

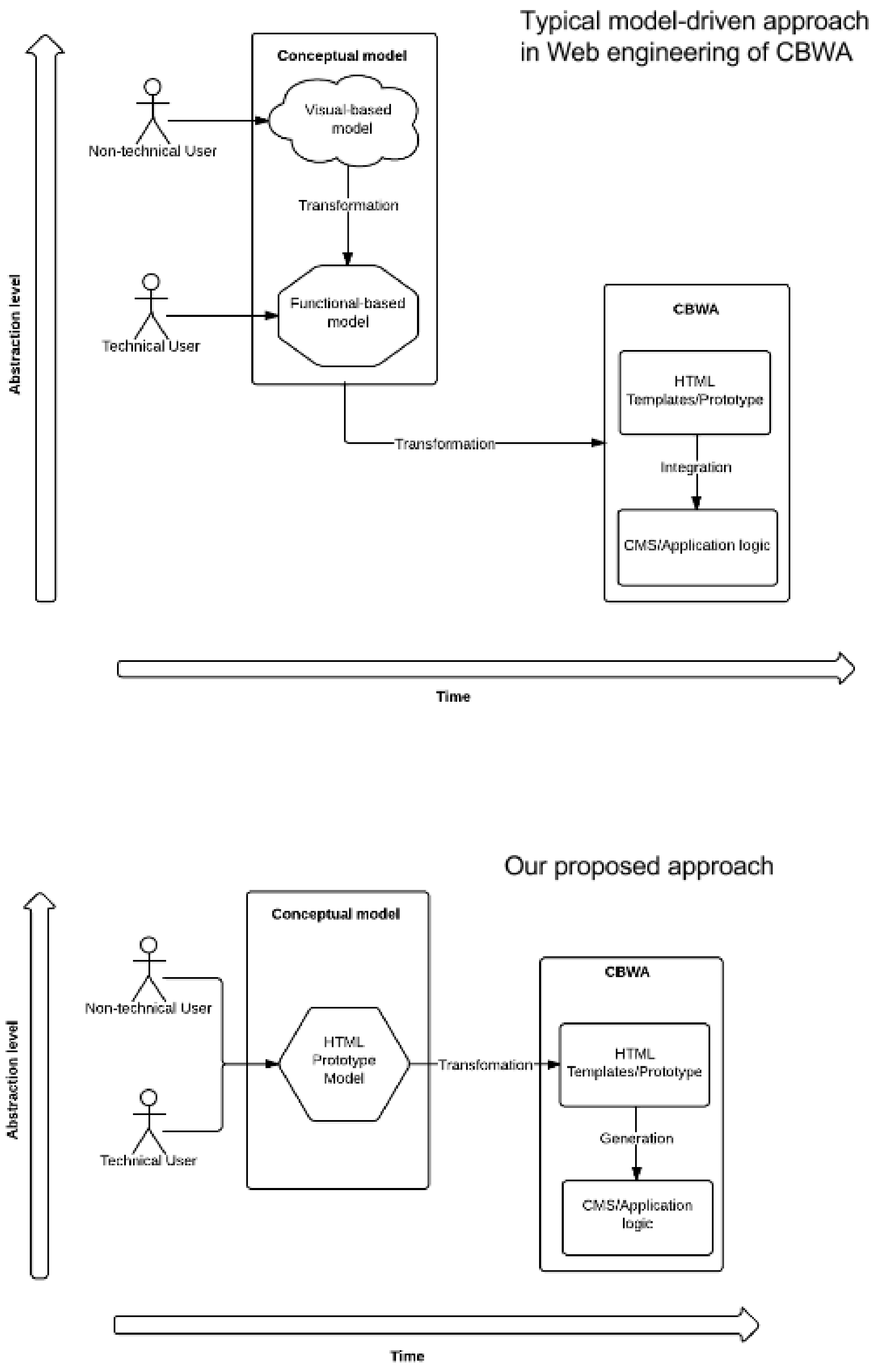


Figure 4.1: The classical process of Web application development (above) vs. our proposed one (below).

4.2 A Generic Metamodel for Content-based Web Applications

As the second part of the approach, to support the above-mentioned methodology, we have designed a metamodel of the CMS that enables transformation of the presentation prototype into the resulting Web application. It is important to clarify that the Web application is represented by the underlying CMS including the initial content that is part of the input conceptual model, although incomplete. The main advantage of this is that the resulting CBWA representation is custom-made for the current project and it contains only concepts and assets that are relevant for that project. Moreover, time and overhead required for creating the CMS from ready-made solutions such as for example Wordpress [57], Drupal [7, 15], Joomla [25, 46] etc. and its modification for the project-related assets is minimized. Another benefit also comes from the fact that the conceptual model in the previous phase is a common one for both technical and non-technical users and the level of its abstraction is more near to the actual CBWA representation, see figure 4.1 (page 33).

For the modeling part of the CMS we have based our work on the CMS common metamodel, proposed by Trias et al. [20]. The CMS common metamodel has been defined after analyzing three of the most popular open-source CMS platforms available in the market, namely: Drupal, Wordpress and Joomla. The resulting metamodel is a union of the metamodels of each of the previously mentioned CMS platforms. The approach chosen by the authors leads to a generic view of the CMS-based Web applications domain.

For the purpose of our research, which focuses in the application of the methodology for content-based Web applications, we have proposed and implemented several modifications leading to a simplified variation of this metamodel. See figure 4.2 for a simplified, high-level presentation of the metamodel's packages and main components. Our proposal is a result of a three-phase process. In the first phase we have analyzed the CMS common metamodel and its characteristics. The aim of this phase was to understand the holistic view of the CMS-based Web applications modeling in general. In the second phase we analyzed some selected real-world projects that use one of the CMS platforms mentioned above to compare the theoretical and practical usage of the model's components. We focused on both the methodology and the technical aspects of these projects. In the third phase we applied some reverse engineering principles to design the simplified variant of the CMS metamodel and fine-tuned it through several iterations during the application of the model concepts into the corresponding tools, discussed in the latter chapters of this thesis.

The metamodel consists of three packages: navigation, user and content. These packages are based on the categorization of Web-related characteristics and their modeling, which was discussed in the previous chapter. The components of our proposed metamodel are: pages, templates, regions, content, user and function packages. This is the minimum set of components that we need to capture the system's structure and behaviour.

The core component of the model is the *page*, which represents the actual pages of the Web application, be they of any type. A page is identified by a unique string, representing the URL address, from which it can be retrieved. The page is always constructed by a

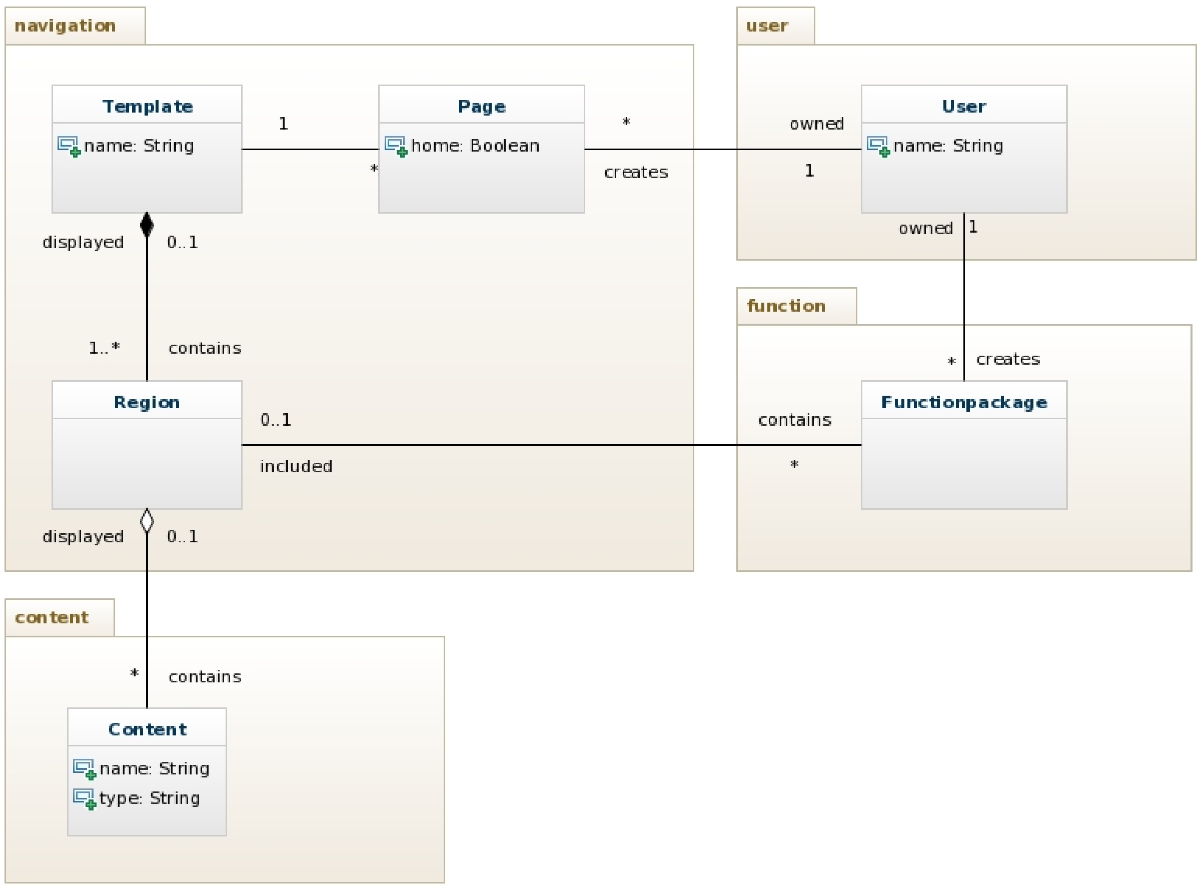


Figure 4.2: Simplified view of main components of the proposed metamodel

template, which represents the page's layout, i.e. the hierarchy and arrangement of the elements from which the page consists. Elements are considered logical page blocks, for example a page's menu, carousel of images etc. These elements are denoted as the *regions* of that page. Regions are distinct areas that do not overlap with each other. They contain selected pieces of the page's *content*, but they can also contain other smaller regions. Content is the "atomic" part of a page that consists of one or more HTML tags, which are later created and maintained in the CMS. The *user* is another central component of the system and it represents any users of the Web application. The user is the only actor of the system (from the UML context), i.e. in the model we do not distinguish the users by their roles. Users can create *functional packages*, which are regions that contain certain application logic, for example dynamic forms. Functional packages are predefined pieces of application logic. They are not automatically generated, although their content can also be modified.

This view of the metamodel is enough expressive but also generic to cover most projects in the domain of CBWA, as it was later proven during the evaluation phase. It is important to notice here the distinction between a model-level and a metamodel-level based approach. A metamodel is a higher level model of the system's components, which is the approach that we are using. Typically, models of a CMS deal with some semantics of that system, involving concepts such as images, comments, articles etc. On a higher level though, all these concepts can be generalized and they are covered in our case by concepts such as content, regions etc. Reaching this level of abstraction requires a more generic definition of the related metaclasses and attributes but also gives wider application possibilities. Therefore it is crucial to keep these conclusions in mind when approaching the latter top-down phases.

As an output, the concepts involved in our model are represented by UML metaclasses. UML is a MOF-compliant language, which ensures for the models establishment under the OMG's approach and their standards. The metaclasses describe the underlying concepts without any context-related information about their way of presentation, which leads to an enough robust definition.

4.3 Semi-automated Generation of the Web Application

At a certain point of time the HTML prototype is considered as final by the involved stakeholders. We propose a technique and a supporting tool (parser) that enable the generation of the underlying CMS for the Web application (semi-)automatically from this final HTML prototype. (Semi-)automatic means that partially the output is generated automatically, but in some specific cases some manual intervention is required.

The interactive prototype is a folder containing static files in HTML, JavaScript, Cascading Style Sheets (CSS) files, images and eventually subfolders, in which those files are located. This folder is processed by the parser in several steps based on an algorithm, see figure 4.4 (page 39).

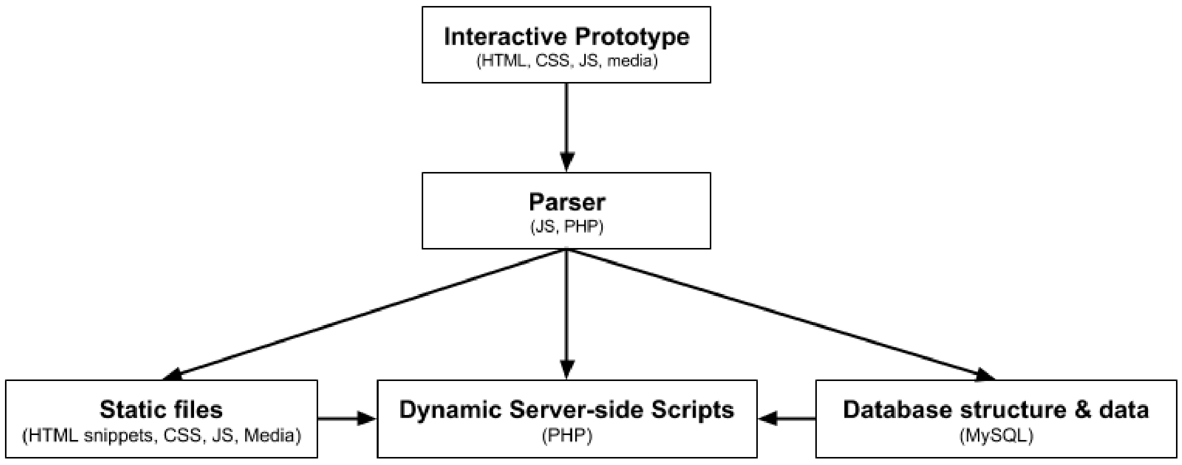

Interactive Prototype
(HTML, CSS, JS, media)
Parser
(JS, PHP)
Static files
(HTML snippets, CSS, JS, Media)
Dynamic Server-side Scripts
(PHP)
Database structure & data
(MySQL)

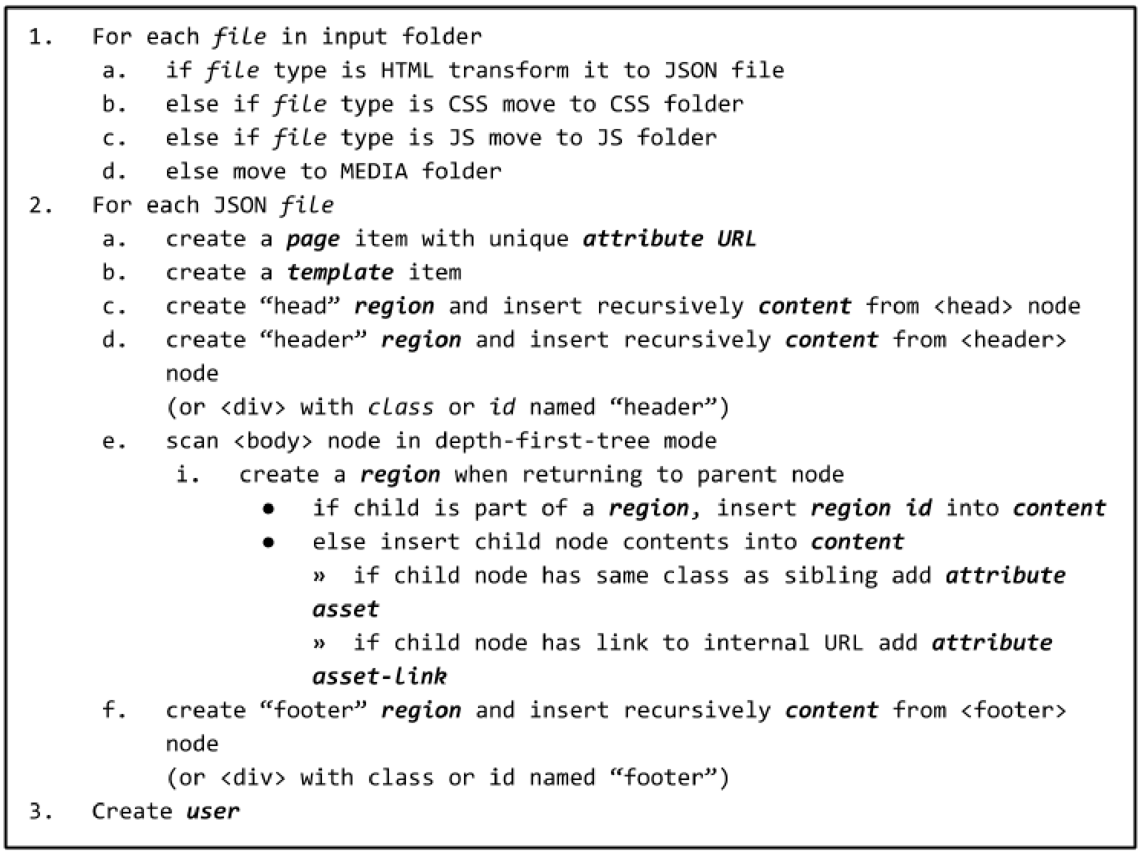
Figure 4.3: Scheme of semi-automated generation of CMS

Each of the HTML input files represents a page-type of the Web application, for example “homepage”, “categories”, “product page” etc. This is a necessary convention for the application of the proposed technique, however, this is not very limiting, because it is quite a standard procedure in real-world projects. Once the input prototype has been processed, the parser generates a temporary file for each page-type in the JavaScript Object Notation (JSON) format. The JSON file contains a presentation of the Document Object Model (DOM) tree of the input HTML files and it facilitates the parsing and application of predefined heuristic rules for the generation of the CMS. For each JSON file a template and a page is generated. If the input prototype contains several pages of the same type, they are also generated and applied the same template. Templates are composed by several logical content blocks, as described previously, i.e. regions. Technically, the regions are identified by the type of content (HTML tags) of which they consist. For example a region of text is distinguished (and separated) from a contact form region, by identifying the closing `<div>` tag of the text region from the opening `<form>` or opening heading `<h>` tag of the contact form. Regions and contents are assigned a unique identifier through which they are later recognized and accessed for modifications, see figure 4.5 (page 40). Identical regions in the files are stored only once. These are repetitive HTML parts of all pages, such as typically the header and footer regions. This is where we apply the heuristic rules, while comparing two pieces of the JSON file, because not always these parts are identical, for example an active element of the menu is highlighted in one page while another one in another page, although the menu is the same element in both pages.

Content blocks, or regions, within one page are considered as assets for the CMS, i.e. the user can create another identical sibling of that asset with the same properties and attributes. He can remove or modify existing ones. Modification options vary based on the type of content (see the next section for the use-case diagrams).

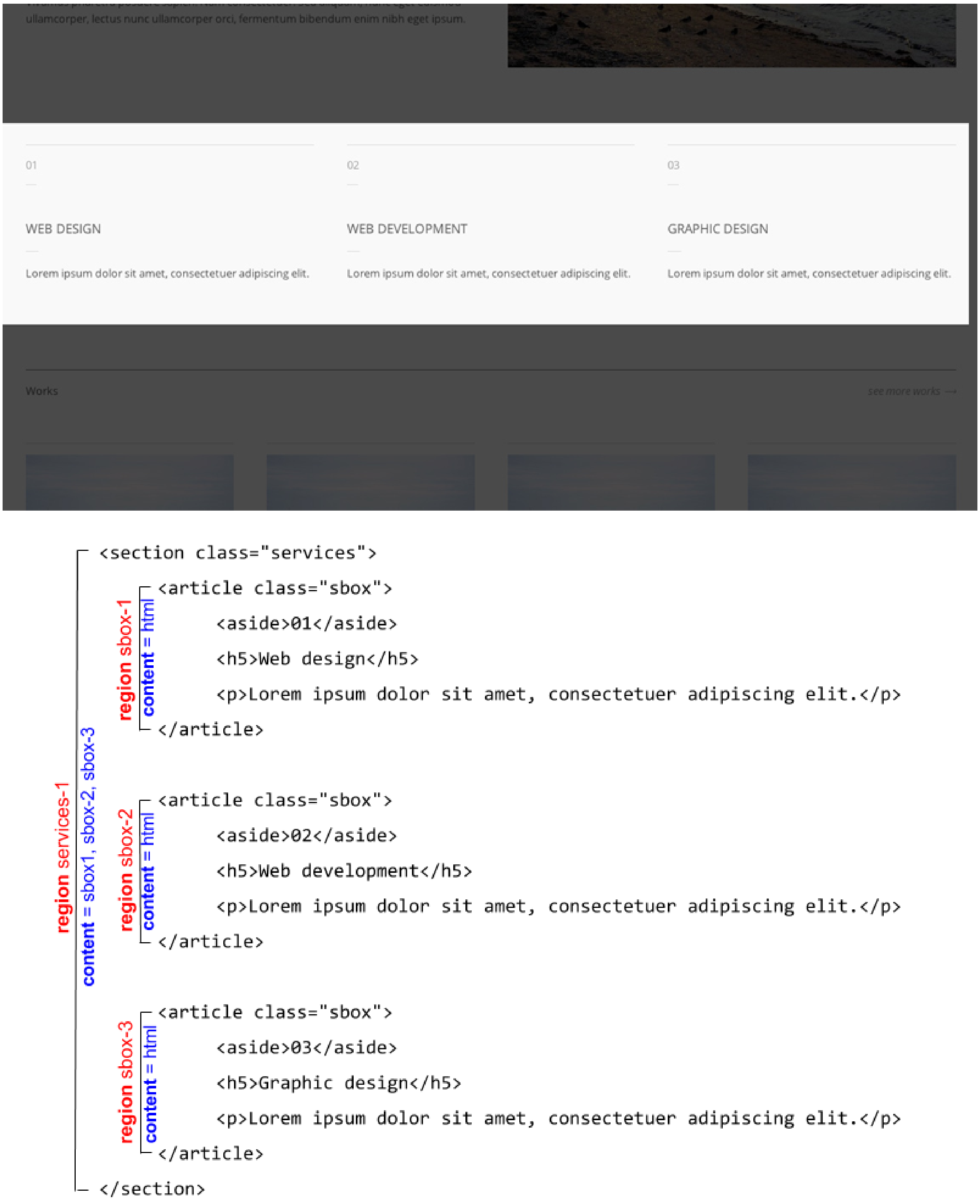
The result of the parser is a representation of the Web application consisting of three types of output. The first are static files such as HTML snippets, CSS, JavaScript and media (typically imagery). They are placed in the server and referenced from the corresponding content elements of the pages. The second type of output files are server-side scripts representing the application logic needed to view and manipulate with the pages, both from an interaction perspective and also from an administration perspective (see next section). It is important to understand that the programmatic logic of the server-side script is also static and not derived from the input prototype. However, parts of it are modified based on the prototype contents. The third kind of output is a script with the database structure and data of the CBWA, with which the server-side scripts will operate. In our case for the proof of concept, we have used a Relational Database Management System (RDBMS), namely MySQL, however the technology does not have any impact on the principles that are applied and theoretically any other technology can be used.

At this moment, the outputs are only produced but they need to be processed manually in order to be integrated into the server, because the development is for the purpose of proof-of-concept scope. An integration with the server would be highly encouraged as it would increase effectivity but it is quite a complex problem itself due to several architectural possibilities of the server.



```
1.   For each file in input folder
     a.   if file type is HTML transform it to JSON file
     b.   else if file type is CSS move to CSS folder
     c.   else if file type is JS move to JS folder
     d.   else move to MEDIA folder
2.   For each JSON file
     a.   create a page item with unique attribute URL
     b.   create a template item
     c.   create "head" region and insert recursively content from <head> node
     d.   create "header" region and insert recursively content from <header>
          node
          (or <div> with class or id named "header")
     e.   scan <body> node in depth-first-tree mode
           i.   create a region when returning to parent node
                  •   if child is part of a region, insert region id into content
                  •   else insert child node contents into content
                      »  if child node has same class as sibling add attribute
                      asset
                      »  if child node has link to internal URL add attribute
                      asset-link
     f.   create "footer" region and insert recursively content from <footer>
          node
          (or <div> with class or id named "footer")
3.   Create user
```

Figure 4.4: Algorithm for parsing the input HTML prototype


ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum.

01

WEB DESIGN

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

02

WEB DEVELOPMENT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

03

GRAPHIC DESIGN

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

Works

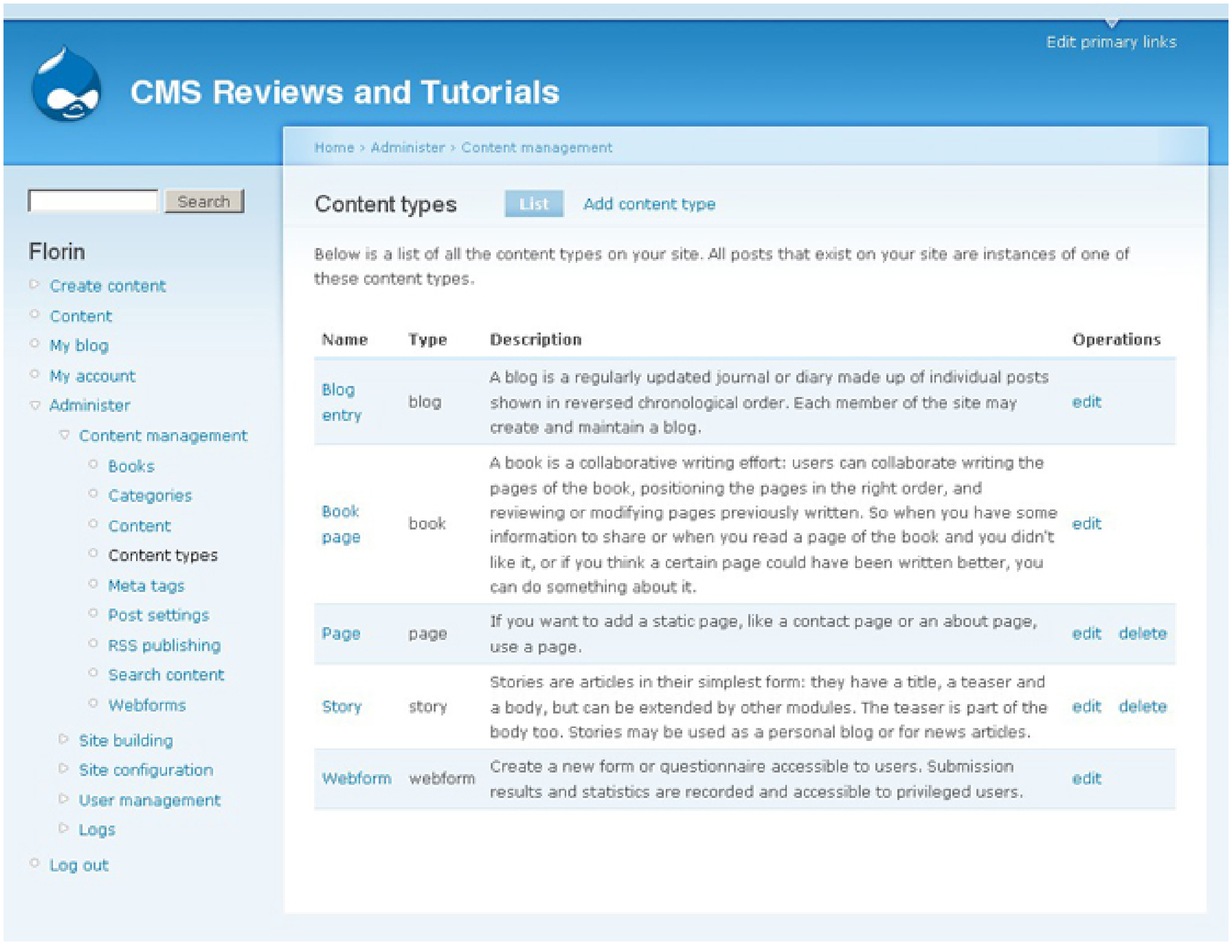
see more works →

```
<section class="services">
  <article class="sbox">
      <aside>01</aside>
      <h5>Web design</h5>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
  </article>

  <article class="sbox">
      <aside>02</aside>
      <h5>Web development</h5>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
  </article>

  <article class="sbox">
      <aside>03</aside>
      <h5>Graphic design</h5>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
  </article>
</section>
```

Figure 4.5: Parsing demonstration from part of a page

4.4 Administration of the Content

Typically a separate interface for the creation and modification of assets, is commonly used in practice. This is partially due to conventions and partially due to the usually top-down modeling approach discussed earlier. Both these reasons are inter-related. Conventionally, when building a CMS, the developer creates a presentation part and an administration part of the Web application. The presentation part is what the end-users see and corresponds to the conceptual model of the non-technical stakeholder, specifically to what he wants that application to present to the users. The administration part is what the users with some "super-powers" see and corresponds more to the technical representation of the Web application, involving concepts such as articles, themes, comments etc. All these concepts are separated and distinct and their relation is usually shown through tables, see an example in figure 4.6. Such presentation corresponds to the database structure of the system and hence it is closer to the technical stakeholder's conceptual level.



Below is a list of all the content types on your site. All posts that exist on your site are instances of one of these content types.

Name	Type	Description	Operations	
Blog entry	blog	A blog is a regularly updated journal or diary made up of individual posts shown in reversed chronological order. Each member of the site may create and maintain a blog.	edit	
Book page	book	A book is a collaborative writing effort: users can collaborate writing the pages of the book, positioning the pages in the right order, and reviewing or modifying pages previously written. So when you have some information to share or when you read a page of the book and you didn't like it, or if you think a certain page could have been written better, you can do something about it.	edit	
Page	page	If you want to add a static page, like a contact page or an about page, use a page.	edit	delete
Story	story	Stories are articles in their simplest form: they have a title, a teaser and a body, but can be extended by other modules. The teaser is part of the body too. Stories may be used as a personal blog or for news articles.	edit	delete
Webform	webform	Create a new form or questionnaire accessible to users. Submission results and statistics are recorded and accessible to privileged users.	edit	

Figure 4.6: An excerpt of a typical CMS administration interface

Based on our methodology and the way which the data assets of the CMS are created, stored and maintained, we propose the usage of the Web application presentation layer (its Web pages) as an administration interface for the content. Using the presentation pages as an administration interface brings several benefits. The rationale behind this is that the presentation has the exact same structure and form as the mental model of the user, as we described previously. Given that the presentation prototype was the initial model for the requirements and the CMS assets are generated based on it and therefore conform to its structure, it is much more effective to use this presentation as the administration interface.

As a proof of concept we have developed a tool to support this technique. It is an extension for the Google Chrome Web browser. When the extension is enabled, the actual presentation part of the Web application turns into administration mode and the Web application assets can be modified by pointing the mouse to the content and right-clicking for a contextual menu that offers a list of possible operations that can be performed with that element. Of course, this can be limited to users with "super-powers" and be accessible after login. Technically, the modifications are asynchronously propagated from the client's browser to the CMS and stored in the server via Asynchronous JavaScript and XML technique (AJAX). Below we will describe some more details about the specific options and operations that this tool allows.

Once the tool is enabled for administration mode, the presentation will turn into the CMS interface. The only difference is that according to the placement of the mouse cursor, an element of the DOM will be highlighted for the user to distinguish that he can interact with that element. An excerpt of the tool is shown in figure 5.6, in chapter 5 (page 54). When the user chooses an element he wants to interact with, he can right-click on it and a context-menu is generated based on that element's type. There are several use-cases from which a user can choose. After choosing an item of the menu, the user can change that element, i.e. modify its existence, content and/or appearance. These modifications are temporarily stored locally in the Web browser first (i.e. in the rendered HTML and CSS level). Then, when the user is ready to deploy the related content changes a JavaScript file will match the modified elements unique identifiers and create accordingly Structured Query Language (SQL) statements in order to persist the changes on the database level.

In figure 4.7 (page 44) we show the possible administration operations in the form of a UML use-case diagram. These operations are available only for the user, who has been identified as an administrator, therefore the login and browsing-related use-cases are not included in this list. By right-clicking anywhere in the page, the user can select, whether he wants to interact with the whole page, with the highlighted HTML element under cursor, or a new HTML element. Below we are going to describe briefly the meaning for each operation.

1. *Interaction with the page.* If the user chooses to interact with the page, he can delete that page, or duplicate it (create an exact same clone). Deleting the page, removes it from totally from the CMS, including its related assets, namely the regions and content. Duplication creates an identical sibling of that page, with same asset and content data (i.e. page structure) but with a different URL address, depending on the page's title.

2. *Interaction with an existing element.* Interaction with an existing HTML element is the most common operation that users will make. Similarly as with the whole page, he can delete it, i.e. remove it from the page's structure, or clone it, i.e. create a clone of that element. In case the element is duplicated, it is assigned the same content value and other attributes as the original. Also it is assigned under the same parenting element and container. Typically the user will duplicate an existing element and then modify its values accordingly. Modification of the element is possible explicitly, i.e. by directly editing its HTML, JS or CSS content attributes and data, which is more appropriate for a technical user. But it is also possible to modify most commonly used attributes, for example for a `<div>` block element that contains just text, it is possible to modify just its text directly.

3. *Interaction with a new element* . Similarly as in duplication of an existing element, the user can also append to the page a new empty element. This is possible by writing custom code, or by choosing one of the most predefined elements such as headlines, text blocks, or images.

As was proven later in the evaluation phase these use-cases are just enough to provide the most common operation for the administration of content via the CMS. They basically allow the user to access any element of the page regarding its context from the presentation view, which makes it much easier for the orientation. This technique is very beneficial to the users, because it reduces the learning curve of operating with a new system, increasing the efficiency of work and most importantly improving the overall usability and user perception of the Web application [A.6, A.2].

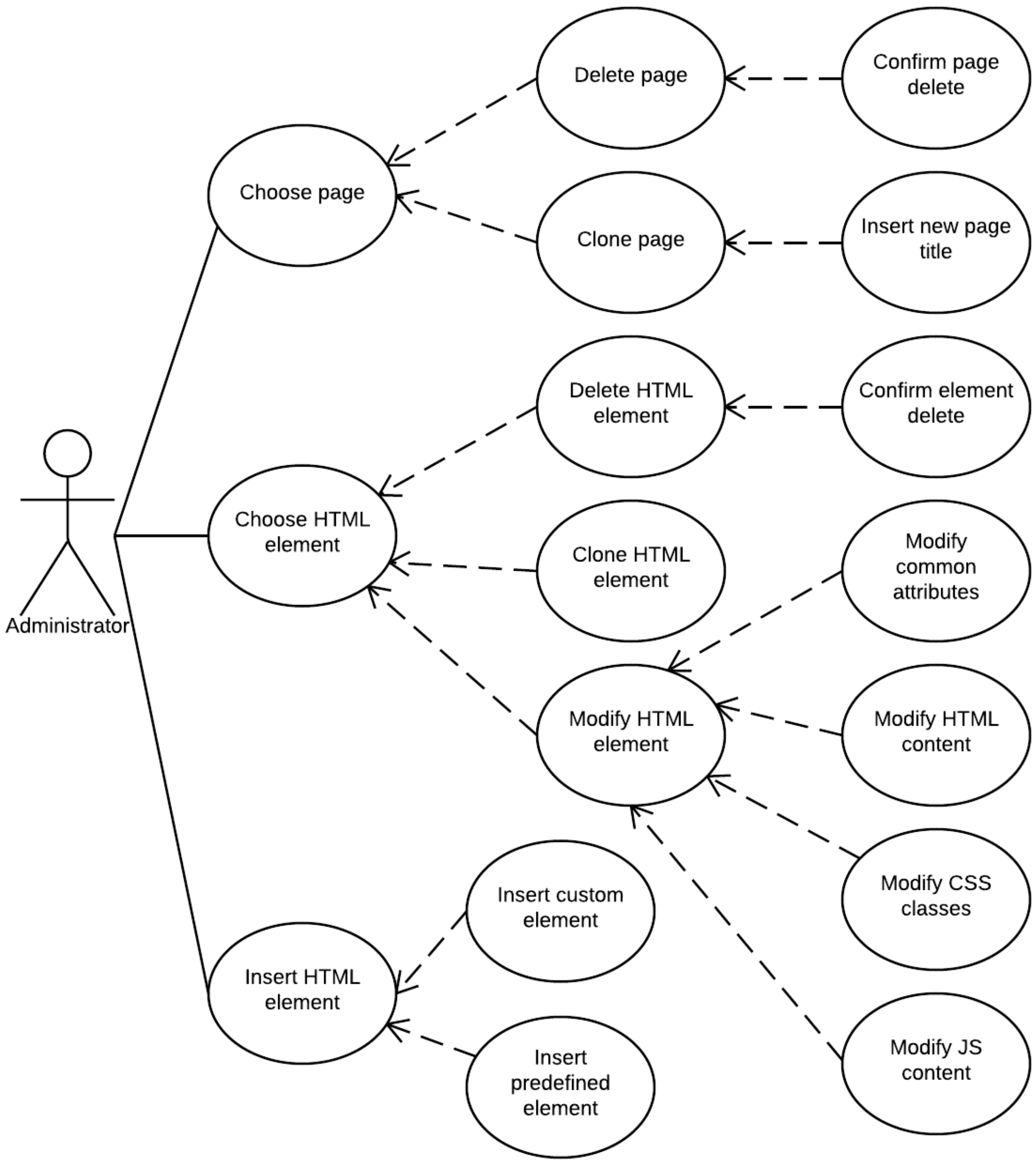


Figure 4.7: Use-case diagram of the administration operations

4.5 Limitations

Our proposed approach, the related methodology and tools have certain limitations. In this section we are going to mention these limitations and give some rationale about their impact in practice.

The first limitation comes from the fact that the approach is specialized and fine-tuned for the family of content-based Web applications. This is a hard limitation that cannot be overcome through iterations or improvements of the methodology or tools. Reasoning from this fact, we could say that the approach is not directly applicable for Web applications that consist of heavy business logic, such as portal-based Web applications, or social-based applications etc. However, the majority of Web applications have at least a small presentation part that consists of user generated content, therefore the approach can be partially applied to this part. Of course, the larger the part, the more beneficial is the application of our proposed principles. On the other side, Web applications that heavily rely on content, such as Wiki pages, presentation websites etc. can gain a much higher level of benefit from this approach.

The second limitation relates with the first one and it is regarding the integration of application logic to a content-based Web application, for example a contact form, or newsletter registration etc. These are common operations that require creation of some dynamic assets. To overcome this limitation, the proposed metamodel contains the Function package. Function package allows integration of programmatically generated content into the system. It requires of course some technical knowledge (database versus scripts integration etc.) and is a task that must be performed by the technical stakeholder. In practice, common CMS solutions allow such functionality to be integrated also automatically from the non-technical user through the installation of a predefined package via the administration interface. Of course, such approach is possible in our case as well. Several packages from the repository of common CMS solutions can be installed and integrated easily into the database and they require little modification of the scripting language. A future optimization of this process by creation of a repository custom-made for the proposed methodology would be beneficial.

The last limitation that we are aware of, is the need for manual intervention for some parts of the methodology. This limitation is two-fold. Firstly is comes due to the very complex and fully-fledged nature of Web applications nowadays, which makes it almost impossible to prepare a “silver-bullet” solution that would fit for all applications. Secondly, the prepared supporting tools are for the proof-of-concept scope. Their further development can improve the automatization process, which would increase more practical application and feedback from the community. This would make the methodology and tools more robust and applicable to a wider range of situations and projects.

Chapter 5

Case Study

For a better understanding of how the proposed approach can be applied in a Web project we describe in this chapter a case study to demonstrate the most important concepts. This is a small-size project, which is based on a real-world scenario and conclusions. For simplicity reasons and better demonstration of the most important factors, some execution details have been omitted and simplified in order to highlight only the most important aspects of the approach.

We consider a project of an event planning company, which wants to create a new website for their services. There is one representative of the company, with limited technical knowledge, who is responsible for the project. He has hired one technical person to carry on the project with him. The aim is to keep project costs as low ass possible, because the company is just starting and their funds are limited. However, the company's representatives are aware that an online presence is very important to their business and therefore they want to keep the website neat, modern and flexible for future development.

5.1 Requirements and Prototype

The company's representative has made a research of the current situation in the field of event planning, in order to best understand the competition and the market. As a part of this research, he has also analyzed the way that similar companies present their services and products in the Web. Below is the initial list of his requirements:

1. Our company's website must be beautiful, modern and neat. It must communicate to the visitor the core values that we follow in our company: creativity, professionalism, friendliness.
2. Our motto is: "Deliver an unforgettable event of your life."
3. Our customers are individuals, families and small businesses with up to 20-30 employees.
4. Our services include event-related planning and event organization end-to-end.

5. We communicate with the customer from the beginning to understand their expectations. Together we choose a theme for the event. Afterwards we provide creative ideas based on that theme and arrange all related details for a perfect organization of the event.

6. The website will consist of two parts, one with static information including description of our core values, our services and our address and other contact details. The second part will be dynamically updated by us and it will include changing information, such as up-to-date, recent events and recommendations from our past customers.

7. We also need a contact form, through which our potential customers can contact us directly, or they can leave us their email address/telephone number and a note and we will contact them back.

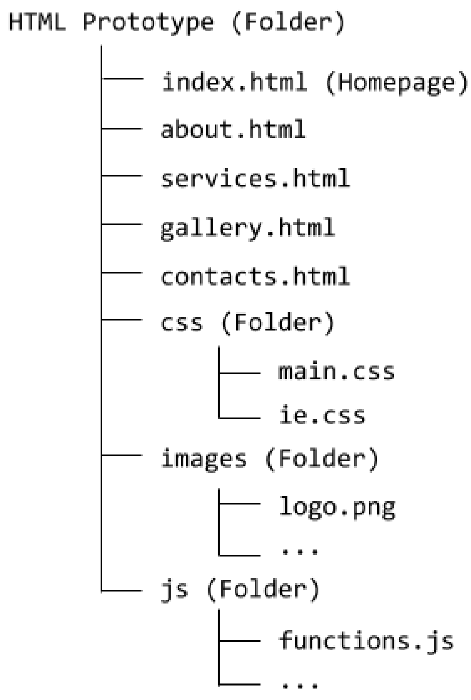
8. A lot of our customers are busy professionals, who are always on the go, therefore our website must be user-friendly and optimized also for mobile devices.

9. The key to marketing of our company will be our social media presence, so the website must support the idea of our visitors and potential customers to give recommendations via our social channels.

As is seen in this case study, the set of requirements is a mixture of functional and nonfunctional requirements. Requirements 1–5 contain high-level information that does not directly impact the way that the prototype will be built. However they give a general idea about what kind of website the owners are looking for. The rest of the requirements are more concrete and they give enough understanding of the final deliverable expectations. Overall this list of initial requirements is just enough specific to give a general idea of the website's content.

Due to the generic overview of these requirements and the fact that the stakeholders want to keep the cost of the project as low as possible, the technical person has looked for and found a free website template that could be appropriate for the initial version of the Web applications presentation, see figures 5.1 and 5.2 (page 50). Of course, it requires several modifications, but it is usable, for an initial version of the website.

In the requirements gathering phase there is still room for changes in the prototype. One of the questions that usually arises is, whether to make this changes within the prototype or whether to make them after the CMS is generated and use the CMS to carry out these changes. This is a decision that is usually done by the technical person. As a rule of thumb, layout and structural changes, especially those requiring changes of the HTML code are more convenient to be executed in the HTML prototype, in order to avoid later modification manual intervention in the data model. A layout change may be considered adding, removing, swapping the positions of certain elements, for example putting the testimonials content block in the left and the recent events block to the right side, see figure 5.2. Structural changes include taking of content blocks and copying or moving them to another page, for example copying the content block with the three event

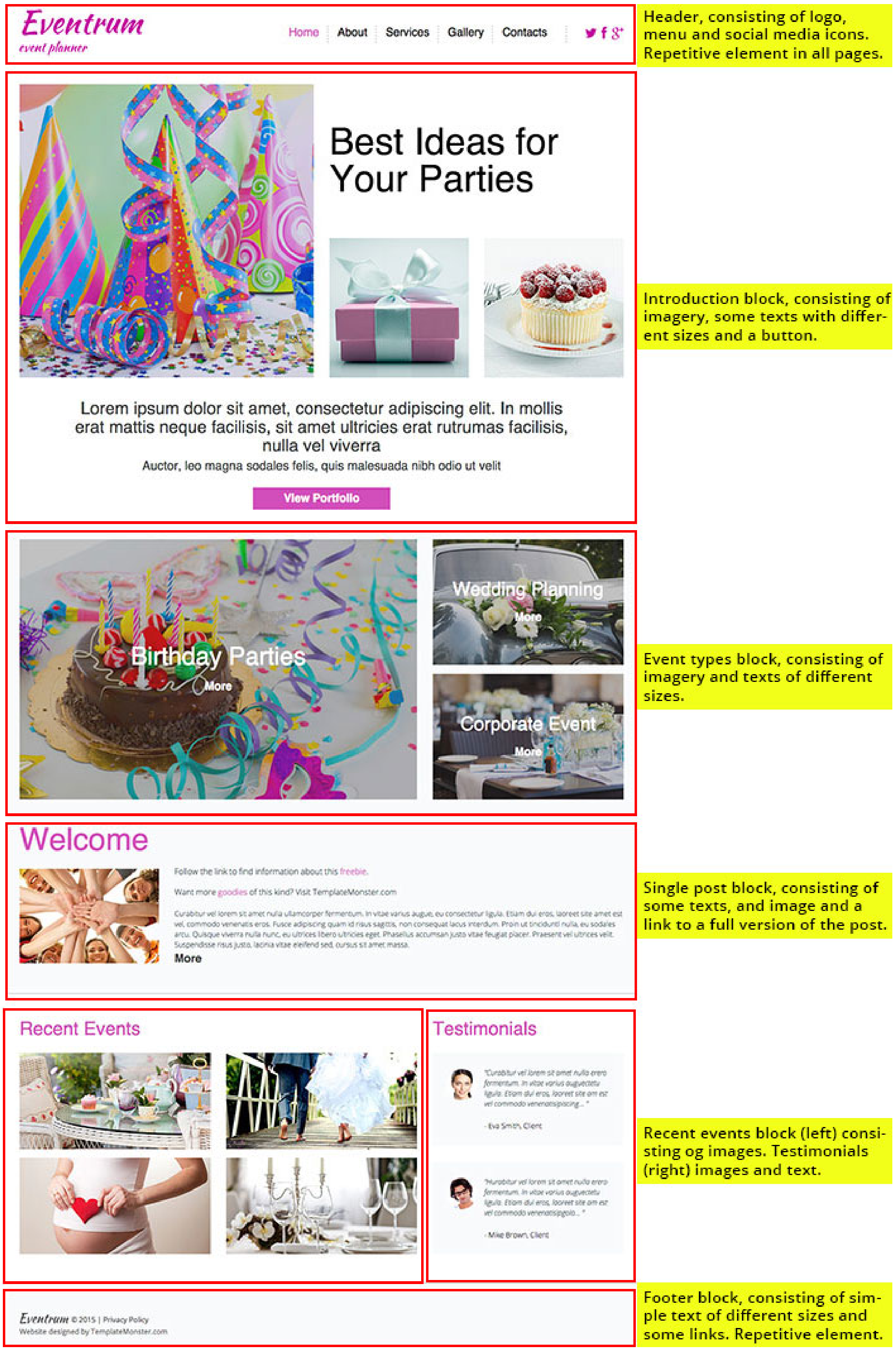


```
HTML Prototype (Folder)
    │
    ├── index.html (Homepage)
    ├── about.html
    ├── services.html
    ├── gallery.html
    ├── contacts.html
    ├── css (Folder)
    │       ├── main.css
    │       └── ie.css
    ├── images (Folder)
    │       ├── logo.png
    │       └── ...
    └── js (Folder)
            ├── functions.js
            └── ...
```

Figure 5.1: Structure and content of the initial HTML prototype

types (birthday, wedding and corporate events) into the page with the presentation of the services. Changes of content (including imagery) and styles can be easily performed later from the CMS without any problematic issues.

It is important to note here, that no temporary models (such as schematic representations of the websites structure, data models etc.) are created and the project members can already communicate with each other very specifically about their concerns using the HTML prototype. For a larger project, this technique is also applicable, except that the resulting HTML prototype would be larger, consisting of more pages and components. Logically the duration of this phase would also be longer, but no theoretical limitations would be encountered, because any requirement at this phase would be in a certain form represented in the actual prototype and fine-tuned later in several iterations. A second important thing to highlight is that picking a ready-made HTML template for the initial prototype is quite a common solution nowadays, especially in small projects with limited budget. However, if no such limitations were present, it is still valid that in the early phases of the project, a technical person would take care of the design of the Web application, therefore delivering a custom-made template for the prototype. The designs are sometimes made in the form of sketches and wireframes, which are also exportable into HTML documents, therefore the technique can also be applied in such case. Graphics can be embedded later, either through later design iterations (which is more common in practice), or via the CMS.



Header, consisting of logo, menu and social media icons. Repetitive element in all pages.

Best Ideas for Your Parties

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In mollis erat mattis neque facilisis, sit amet ultricies erat rutrumas facilisis, nulla vel viverra

Auctor, leo magna sodales felis, quis malesuada nibh odio ut velit

View Portfolio

Introduction block, consisting of imagery, some texts with different sizes and a button.

Event types block, consisting of imagery and texts of different sizes.

Follow the link to find information about this freebie.

Want more goodies of this kind? Visit TemplateMonster.com

Curabitur vel lorem sit amet nulla ullamcorper fermentum. In vitae varius augue, eu consectetur ligula. Etiam dui eros, laoreet site amet est vel, commodo venenatis eros. Fusce adipiscing quam id risus sagittis, non consequat lacus interdum. Proin ut tincidunt nulla, eu sodales arcu. Quisque viverra nulla nunc, eu ultrices libero ultricies eget. Phasellus accumsan justo vitae feugiat placer. Praesent vel ultrices velit. Suspendisse risus justo, lacinia vitae eleifend sed, cursus sit amet massa.

More

Single post block, consisting of some texts, and image and a link to a full version of the post.

"Curabitur vel lorem sit amet nulla erero fermentum. In vitae varius auguecotu ligula. Etiam dui eros, laoreet site am est vel commodo venenatispiscing... "

- Eva Smith, Client

"Hurabitur vel lorem sit amet nulla erero fermentum. In vitae varius auguecotu ligula. Etiam dui eros, laoreet site am est vel commodo venenatispgolo... "

- Mike Brown, Client

Recent events block (left) consisting og images. Testimonials (right) images and text.

Eventrum © 2015 | Privacy Policy
Website designed by TemplateMonster.com

Footer block, consisting of simple text of different sizes and some links. Repetitive element.

Figure 5.2: Preview and content blocks of the homepage in the HTML prototype

In figure 5.2 is shown a preview of one of the pages of the prototype, specifically the homepage. The rest of the pages have a similar look and feel but consisting of different content blocks.

5.2 Generating the CMS from the HTML prototype

After several iterations and modifications, the stakeholders have reached a final version of the prototype and are ready to start preparing the real content. Given the visual-oriented nature of the prototype, most concerns at this phase were related to the Web application's look and feel. Such modifications can be handled also directly from the CMS, as described previously.

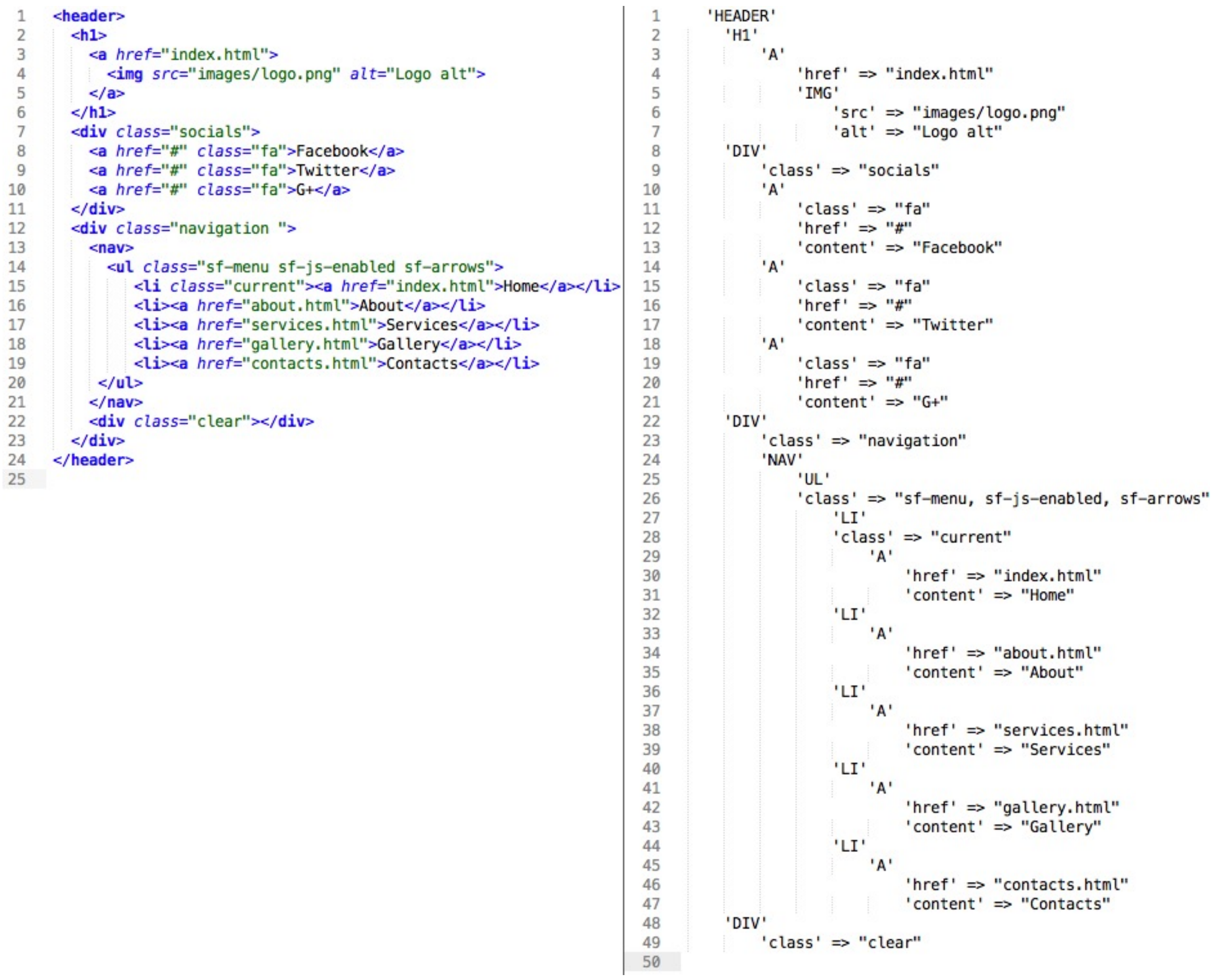
Once the HTML prototype is ready, it can be used as an input for the (semi-)automated generation of the corresponding CMS. This input is a folder with the structure and file content as shown in figure 5.1. The folder consists of static files in HTML, CSS, Javascript and some imagery that is part of the initial pages presentation. No dynamic (script) files are included in the folder. This form of the HTML prototype is a typical artefact for any Web project.

At this moment the folder is parsed by the scripts that were discussed in the previous chapter. These scripts consist of dynamic server-side files in PHP. The script can be run in a local or remote server. For this project the scripts were run locally.

The script will first arrange the input files depending on their type, namely grouping the markup, stylesheets, JS and media files together in separate folders. At this moment in the markup the URL addresses to those files are also updated accordingly. Final HTML markup files content is transformed into JSON format for easier post-processing, see figure 5.3. The JSON files are the core of the new CMS and its initial content. Its nodes are further analyzed by the parser and processed in order to deliver the Web application's structure and initial contents. First the `<header>` and `<footer>` nodes are stored separately. Then the scanning is done recursively for the whole `<body>` node and using the depth-first algorithm, each node content is stored into a database structure for persistence. Nodes that contain no children nodes are stored directly, while those that do are stored as regions, indicating the pages content structure, see figure 5.4.

Another set of pre-processed (static) server-side files are merged to the final output and modified according to the input prototype content. These files contain the classes and functions for a generic Web application business logic, in a model-view-controller (MVC) and user architectural pattern, naturally according to the Web application's metamodel described in the previous chapter. Also they contain the Web application's content structure in the form of SQL database files exports, see demonstration in figure 5.5.

After these files are generated they are ready to move to the server as they are. This operation must be manually handled, because of the many possible server configurations that must be taken into account, which account for a very complex implementation logic of an automated installation file and it is not within the scope of this work.



```html
<header>
  <h1>
    <a href="index.html">
      <img src="images/logo.png" alt="Logo alt">
    </a>
  </h1>
  <div class="socials">
    <a href="#" class="fa">Facebook</a>
    <a href="#" class="fa">Twitter</a>
    <a href="#" class="fa">G+</a>
  </div>
  <div class="navigation ">
    <nav>
      <ul class="sf-menu sf-js-enabled sf-arrows">
         <li class="current"><a href="index.html">Home</a></li>
         <li><a href="about.html">About</a></li>
         <li><a href="services.html">Services</a></li>
         <li><a href="gallery.html">Gallery</a></li>
         <li><a href="contacts.html">Contacts</a></li>
      </ul>
    </nav>
    <div class="clear"></div>
  </div>
</header>
```

```
'HEADER'
  'H1'
    'A'
      'href' => "index.html"
      'IMG'
        'src' => "images/logo.png"
        'alt' => "Logo alt"
  'DIV'
    'class' => "socials"
    'A'
      'class' => "fa"
      'href' => "#"
      'content' => "Facebook"
    'A'
      'class' => "fa"
      'href' => "#"
      'content' => "Twitter"
    'A'
      'class' => "fa"
      'href' => "#"
      'content' => "G+"
  'DIV'
    'class' => "navigation"
    'NAV'
      'UL'
      'class' => "sf-menu, sf-js-enabled, sf-arrows"
        'LI'
        'class' => "current"
          'A'
            'href' => "index.html"
            'content' => "Home"
        'LI'
          'A'
            'href' => "about.html"
            'content' => "About"
        'LI'
          'A'
            'href' => "services.html"
            'content' => "Services"
        'LI'
          'A'
            'href' => "gallery.html"
            'content' => "Gallery"
        'LI'
          'A'
            'href' => "contacts.html"
            'content' => "Contacts"
  'DIV'
    'class' => "clear"
```

Figure 5.3: An excerpt of the homepage representation in HTML and “readable” JSON format

```php
$jsonStruct = new RecursiveParser(
      new RecursiveArrayIterator(json_decode($json, TRUE)), RecursiveParser::SELF_FIRST
);

foreach ($jsonStruct as $key => $val) {
      if(is_array($val)) {
            storeRegion($key);
      } else {
            storeContent($key, $val);
      }
}
```

Figure 5.4: Recursive parsing and storing of the file content and regions

```
<div class="grid_6" id="products">
  <h2>Best Ideas for Your Parties</h2>
  <div class="row" id="r1">
     <img src="images/p1_img1.jpg" alt="">
  </div>
  <div class="row" id="r2">
     <img src="images/p2_img1.jpg" alt="">
     <img src="images/p2_img2.jpg" alt="">
  </div>
</div>


CREATE TABLE `page` (
...
)

INSERT INTO `page` (`id`, `home`, `title`, `template`) VALUES
(1, 1, 'Best Event Planning Company', 'homepage'),
...
;


CREATE TABLE `content` (
...
)

INSERT INTO `content` (`id`, `type`, `inner_content`) VALUES
...
(22, 'simple', '<img src="images/p1_img1.jpg" alt="">\r\n'),
(23, 'simple', '<img src="images/p2_img1.jpg" alt="">\r\n'),
(24, 'simple', '<img src="images/p2_img2.jpg" alt="">\r\n'),
(25, 'simple', '<h2>Best Ideas for Your Parties</h2>\r\n'),
...
;


CREATE TABLE `region` (
...
)

INSERT INTO `region` (`id`, `name`, `content`, `regions`) VALUES
...
(11, 'r11', '22', ''),
(12, 'r12', '22, 24', ''),
(13, 'products', '25', 'r11, r12'),
...
;
```

Figure 5.5: Part of a page (above) and its database representation in the CMS (below)

5.3 Administration of Content

Once the configuration files are moved into the corresponding server (or put into operation in a local server) it is possible to start administration of the content from the presentation. We will demonstrate here how the administration of the pieces of content and regions shown above would occur, for example during the activity of translating the content texts into Czech language, since the original template was in English and the change of some images.

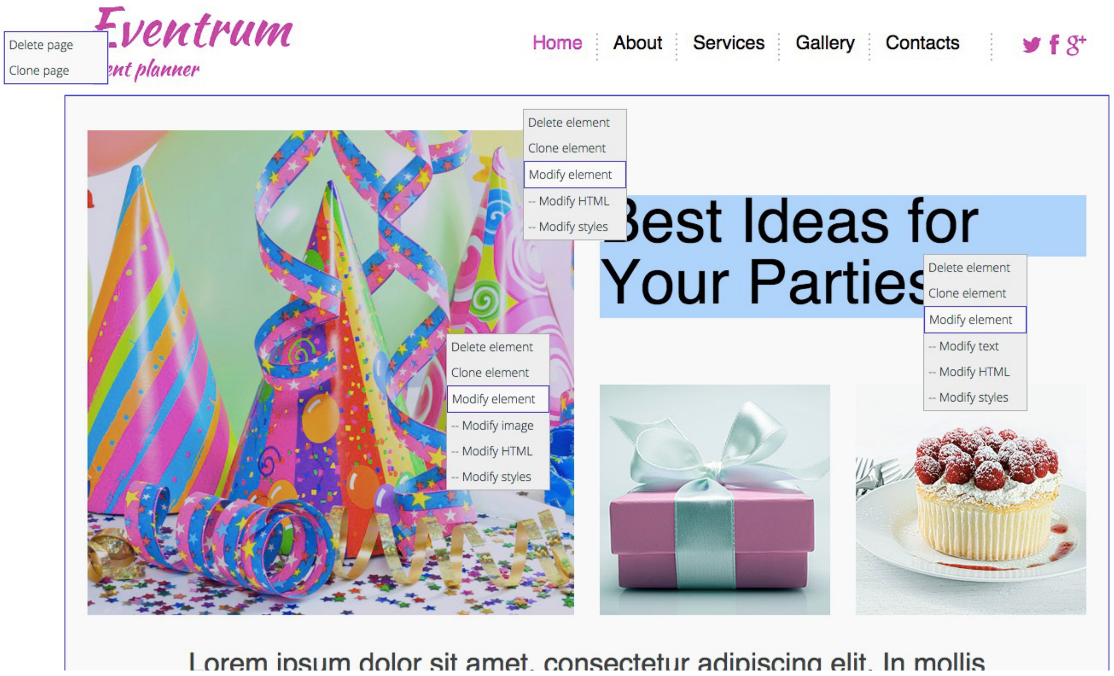


Figure 5.6: Administration options based on the content type

As it is depicted in figure 5.6, the user has several options for modification of the presentation. Hovering on the top level elements of the page will give him the options to delete or clone the whole page. Focusing on the content, namely the introduction block, which was discussed in the previous section, depending on which element the user will chose, he is given the appropriate modification options. Highlighting the `<h2>` node he can easily change the text of the content element with the id 25, highlighting the image on the left, will give the option to modify the content element with the id 22, or highlighting the whole block he can modify the HTML content or manipulate with the whole region with id 13.

Using the presentation as the administration interface is very efficient and the modification of the initial template towards the final user expectations is fast and easy. So the further requirements of the stakeholders can be quickly executed once the CMS is in place as shown above.

Chapter 6

Evaluation and Results

In this chapter we present the evaluation of the approach and the achieved results. The evaluation was carried out during the first half of the year 2014. Before this time the approach was applied and tested several times with real and simulated Web engineering projects with different companies and individuals. The purpose of this preliminary testing was to get initial feedback from the involved people as well as to fine-tune the methodology and tools to fit as best as possible in the different scenarios. From the beginning of the evaluation period, for a period of six months no further modifications and development was made, in order to keep the result data consistent and comparable.

Evaluation was done two-fold, qualitatively and quantitatively [A.6, A.2]. The purpose of the qualitative evaluation phase was to gain general feedback and conclusions about overall performance criteria for our proposed approach. The focus of the qualitative evaluation was the process of applying the methodology in Web engineering projects, i.e. the human factor. The qualitative evaluation was done with the goal of assessing the applicability of the semi-automated generation of the CMS based on ready-made templates. The focus of this phase was the evaluation of the concept and the related tools.

Finally an assessment of the overall approach according to evaluation criteria from related studies and works is presented. This is done for the purpose of comparing our approach with similar other approaches in the same domain, where applicable. The focus of this part is to distinguish on a theoretical level the cases, where the methodology would be beneficial and under which circumstances regarding the stakeholder's needs and requirements.

6.1 Qualitative Results

The qualitative evaluation was done via selected real-world Web engineering project involving the creation of a content-based Web application. We executed three projects following our methodology with selected companies in the Czech republic. The projects were chosen of different scales, complexity and purpose in order to rule out the specific conclusions that may be related to the individual projects.

The first project delivered a relatively simple Web application for the presentation of a musician's curriculum and his concerts program. The second project was with a graphic studio that required a portfolio presentation of their design works. The third one was an online catalogue presentation for a medium-sized company selling food products.

To evaluate the approach and its application process in the projects we conducted survey interviews with project team members after the completion of the project. We interviewed a total of fifteen people using the same survey interview questions. The interviewed members were directly involved in the execution of the projects. They were chosen from both technical and non-technical domain in order to get the feedback from people having different background and skills. Each of the interviewed persons was given a list of statements and they were required to rate the statements in a scale from zero to four; zero denoting full disagreement and four denoting full agreement with that statement. The statements of the survey aimed to rate the approach based on the following areas:

1. Using of the presentation prototype as a conceptual model
2. Generation of CMS based on the HTML prototype
3. Usage of the presentation for the administration of content
4. Overall perception of the methodology

At first an introduction statement was given to explain the meaning of each of these areas. Then one question was given to be rated as described previously with an agreement level of zero to four. For each area the same set of criteria were chosen for the evaluation, namely:

1. How well the individual understands this part of the methodology
2. How often he thinks that this part is applicable in practice
3. How easy would he say was its application
4. How well does he think the application of the methodology has fulfilled the goals

In table 6.1 (page 58) we present a summarization of the scores and results. Under each line of the table we show the number of answers that were given for the given score for each particular area and evaluation criteria. We have highlighted the cells containing the highest number, for easier orientation through the table's data. Further we are going to asses the evaluation statements for each area.

In the following paragraphs we make a summary of each area with regards to the qualitative evaluation results and conclusions.

Presentation prototype as a conceptual model: Usage of the presentation prototype as a conceptual model for the communication between stakeholders during the requirements gathering phase rated very well, as is shown by the evaluation result. 93% of the respondents stated that they understand well this part of the approach and that they find it applicable in practice. 66% of the respondents rated its application as easily feasible, while 60% said that it fulfilled their expected goals.

The reasons that were given behind the lower scores were related to the expectation of communicating and covering non-functional requirements. This is clearly a limitation as was discussed in the previous chapters, however it is possible to keep the presentation prototype as a conceptual model only for some part of the application and combine it with other models or tools that are more appropriate for other application-specific requirements.

Semi-automated creation of the CMS: This part of the approach rated the lowest among all the interview areas. Only 33% of the respondents stated that they understand well the principle of the creation of the CMS based on the HTML prototype, while another 33% were neutral on this statement. 47% thought that this is applicable in practice. Only 20% of the respondents found its application easily feasible, while another 47% were neutral on this statement. The majority of the respondents, over 53% were skeptic about the fulfillment of the expected goals from this part of the approach.

The reasons given behind the lower scores were related with a common concern about the applicability of the approach in non content-based Web application. One of the project contained a large part of application logic related with the sales and administration of products and their attributes, not just for the presentation part. Our assessment regarding this part is that the methodology has been fine tuned for content-based Web applications. So have been the used tools, which make the application quite limited at this phase. Another reason of the lower scores were bugs that were discovered during the application. Given the fact that we did not want to modify the approach during the evaluation phase these concerns have lead to lower scores in all the interviewed members. These concerns are a matter of further development and modification of the methodology but it is not within the scope of this work. It should serve as inspiration for other research groups, who would like to investigate other types of Web applications and want to apply a similar methodology.

Content administration via the presentation: The principle of administration of content from the presentation interface rated best in the interviews. Over 93% of the respondents understood well this part of the methodology and also found it well applicable in practice. Over 70% of them thought that the application is easily

Table 6.1: Results of qualitative evaluation

Presentation prototype as a conceptual model	**0 - fully disagree**	**1**	**2**	**3**	**4 - fully agree**
Well understanding	0	0	1	8	6
Applicable in practice	0	0	1	7	7
Ease of application	0	2	3	9	1
Fulfills expected goals	0	3	3	9	0
Semi-automated creation of the CMS	**0**	**1**	**2**	**3**	**4**
Well understanding	1	4	5	4	1
Applicable in practice	1	3	4	6	1
Ease of application	1	4	7	1	2
Fulfills expected goals	1	3	8	3	0
Content administration via presentation	**0**	**1**	**2**	**3**	**4**
Well understanding	0	0	1	8	6
Applicable in practice	0	0	1	7	7
Ease of application	0	1	3	9	2
Fulfills expected goals	0	1	4	9	1
Overall perception of the approach	**0**	**1**	**2**	**3**	**4**
Well understanding	0	1	2	6	6
Applicable in practice	0	0	2	8	5
Ease of application	0	2	3	9	1
Fulfills expected goals	0	2	3	8	2

feasible and that it fulfills the expected goals, while less 26% were neutral on these statements.

The reasoning behind these scores were related to the usability and efficiency of the proposed process and its short learning curve. Also the method is easy and very straight-forward, because it is according to the well-known WYSIWYG (what you see is what you get) principle, which is widely used in the Web engineering community projects, although only on the level of simple HTML editing tools.

Overall perception of the approach: The overall perception of the approach was also very positive. Over 80% of the respondents stated that they have a general good understanding of the approach and more than 86% said that they find it applicable in practice, which is a very good overall performance result, given the fact that some of the respondents had problems with the part of the semi-automated generation of the corresponding CMS. About 67% of the interviewers found that the approach is easily applicable and that it fulfilled their expectations, while less than 20% of them were neutral on these statements.

Generally the reasons given on the overall evaluation part were positive and they found the approach and the related methodology and tools as promising and optimistic for the future.

6.2 Quantitative Results

The quantitative evaluation phase was done for the validation of the second part of the approach, i.e. the semi-automated generation of the CMS from the HTML prototype, where some concerns were raised during the qualitative phase of the evaluation. The evaluation was done by processing several ready-made hi-def HTML prototypes based on freely available Web templates online [3, 19, 22]. These templates were chosen by hand and they were picked to simulate by a maximum extent modern real-world projects.

A total of sixty templates were downloaded and used for the experiments. The templates had a maximum of three page types. As a page type we consider a unique layout of the page elements different from the other pages, for example homepage, contacts page, portfolio page etc. For the purpose of conducting the experiment and analyzing the relations between the data we decided to choose the same amount of templates for each number of page types, i.e. twenty for each, see figure 6.1.

After downloading the template to the local drive, they were parsed one by one. After the parsing was completed the resulting CMS was analyzed and evaluated in-depth for each case. We focused on measuring the amount of manual operations needed after parsing of the prototypes. By manual operation we consider the need of modification of the input prototypes or the output files in order to correspond to the context of the represented Web application by that template, respectively the corresponding CMS. As we described previously, no changes of the tools were made throughout the evaluation phase, only the input and output artifacts were modified.

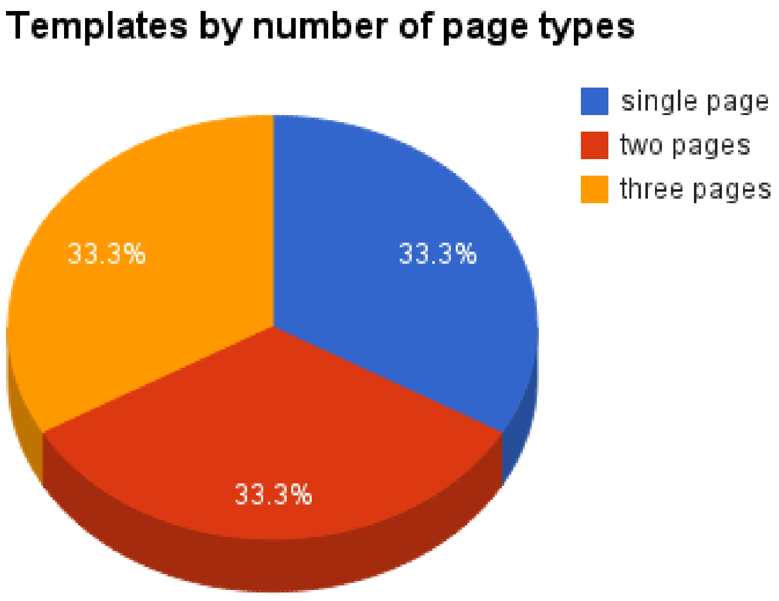


Figure 6.1: Graph showing percentual split of templates by page types

The evaluation showed that overall the need for manual modification is considerably small, because the underlying metamodel is quite general and the technique is based on reverse engineering principles that are common for most Web applications in the domain. The minimal number of manual intervention (for modification) was required in the single page type templates, which is expected, because of their simple and flat structure. Namely a total of 4 manual operations occurred for the whole set of twenty templates, i.e. 20% of the templates required one manual operation and 80% needed no modifications. Out of the two page types templates, 50% of them required no modifications either. 30% of them needed one manual intervention, 15% two manual interventions and 5% (just one of the templates) required a three manual modifications. This means a total of 15 manual operations was required overall during the parsing of the twenty two page types templates. For the set of templates with three page types 35% needed no manual modifications and another 35% needed just one modification. 20% of templates required twice a manual intervention after parsing and 10% required three manual interventions, leading to a total of 21 manual operations for the whole set of templates. No more than three manual interventions was needed for any of the templates included in the experiments. However,

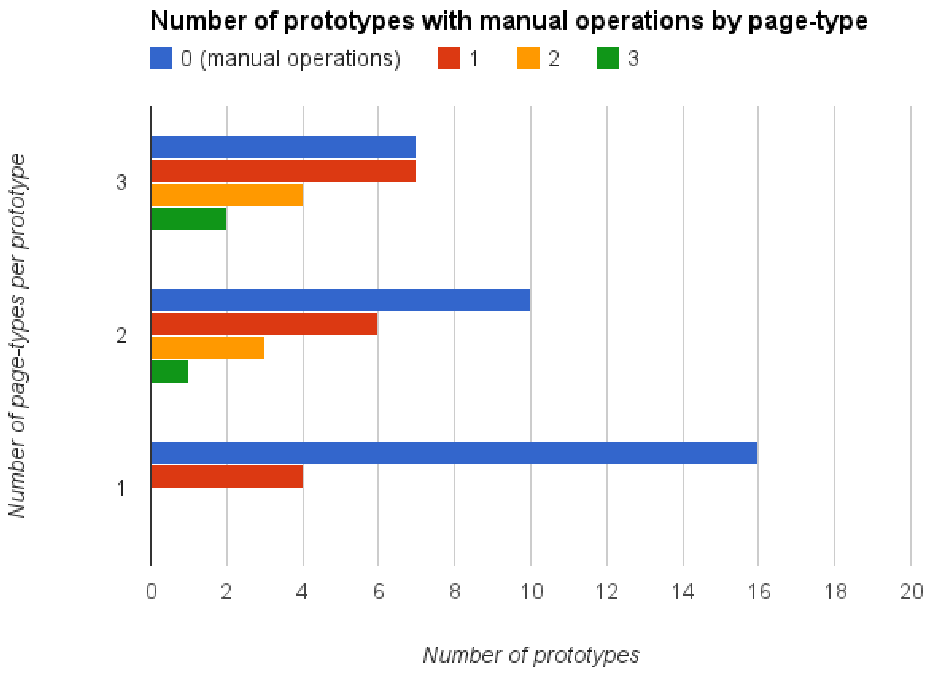


Figure 6.2: Graph of templates and number of manual operations

data tend to incline to a conclusion, that more complex templates require more manual operations, making the parsing less effective overall. On the other hand most content-based Web applications, based on empirical analysis of another set of over one hundred of other templates downloaded from the previously mentioned sources of free online templates, contain within one and five page types. Hence we can assume that the amount of manual operations in general would grow linearly based on the number of templates but then would saturate and not grow further because no more new page types would be introduced in a real-world project.

The amount of manual operations increases by the number of page types contained in the input prototype, see figures 6.2 and 6.3. However, the need for manual intervention can be further reduced in two ways: a) by introducing development conventions that could be used in the HTML prototype, b) by building more variations of the underlying model that are appropriate for specific types of projects. Each of these approaches has its pros and cons. Introduction of conventions is a simple solution but it is a hard task in reality, because of the very nature of Web applications and their context. Also, among the development community in Web engineering there are certain patterns that the developers of templates are used to follow and changing of those patterns is a very hard task execution-wise. The second solution of introducing further variations of the model of the Web application, i.e. leading to modification of the programming techniques involved in the parsing and generation of the CMS is also a hard task that requires time, but it is more straight-

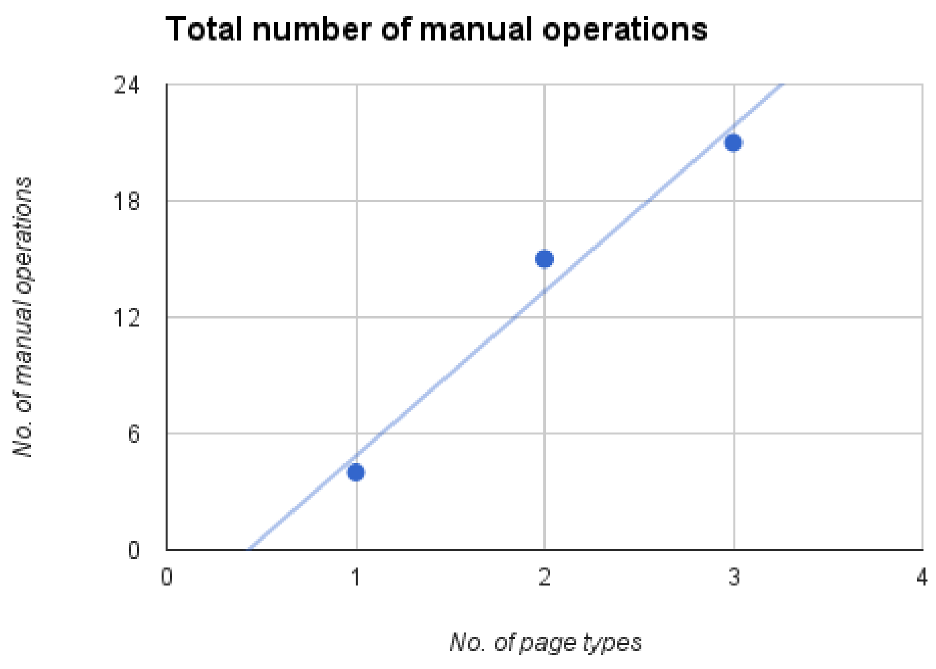


Figure 6.3: Total number of manual operations by number of page types and its trendline

forward and safe. Both these approaches will be considered for future works within our team of research.

6.3 Comparative Evaluation

In order to put the approach under a relative perspective with regards to other works in the Web engineering domain, we have done a comparative evaluation and assessment based on some chosen criteria. Such an assessment is important to prove the relevance and point out pros and cons of different possible approaches based on the environment of the project at hand.

To chose the comparative evaluation criteria we consider related studies, in order to be as much standard as possible in the evaluation process. Specifically we have used and adapted criteria from a reference model for the comparison of model-driven approaches in the Web engineering domain as defined by Saraiva J. et al. [42, 43]. We have found these two studies as those that are most relevant for our research and approach, because they focus on the comparison criteria for related modeling languages in Web engineering and the chosen aspects of these criteria are commonly used also in other similar works. Below we describe these criteria on a high-level for a better understanding of their concern and finally present the comparison chart between the our approach and the other approaches presented in the background chapter of this thesis.

DM: Support for Domain Modeling. The domain modeling support is the criteria that deals with the ability of the approach to represent concepts of the corresponding domain (in our case Web engineering) using appropriate modeling notions. This means that we evaluate, whether the approach covers visualization concepts such as diagrams, charts or other blueprints for the modeling of that domain. Domain modeling ability is an important factor for the understanding and further consistent development of every approach.

MAB: Ability to Model Application's Behavior. The ability to model application's behavior concerns the possibilities of the approach to support modeling of the Web applications dynamic behavior, so-called business logic. The most important aspects of application's behavior modeling are considered the possibility to control changes of the domain concepts and whether the process of these changes can be specified in a human-readable form. The ability to support modeling of business logic may vary, depending on the approach at hand, so the evaluation of this criteria can be subjective, however it is important to distinguish whether the given approach does focus in the business logic at all and if so, up to what extent.

NPF: Support for Navigation and Page Flow. The third criteria deals with the ability of expressing and presenting possible paths of navigation between different pages of the Web application. In a certain context a path can also exist within the same page, so generally this means covering and expressing of navigation paths between any source and destination pages and/or its elements. The support for modeling navigation and page flow is almost implicitly required for any approach that deals with Web applications, therefore this criteria is included in the evaluation and is considered as one of the most importants.

MUI: Modeling of User Interfaces. The ability to model the Web application from a user interfaces aspect is another evaluation criteria that we consider. Again, this is a fundamental aspect when dealing with Web applications in general, but besides the overall support for modeling user interfaces, here we also evaluate how the approach deals with this part. When the approach requires additional software or tools for the modeling of user interfaces, we evaluate this fact as only partial support.

TM: Transformation of Models. The fifth criteria that is considered in this study deals with the ability of the approach to support transformation of a source model to a destination model in an automated and error-prone way, without loss of precision and consistency. In our case we also consider within this part the ability to transform models to code, because the aim of model transformation in general is to reduce effort and problems with the generation of the application's artefacts. For example a model of the Web application produced by the business analyst can be transformed into another model for the application designer that would match his point of view to the application.

CGA: Completeness of Generated Application. In this criteria we factor the ability of the approach and its related tools to generate a complete and fully functional Web application. Another aspect that we consider here is the amount of manual intervention required by the project's team members during this process, as we discussed previously. This aspect, however, may vary depending on the type of application and the purpose of the given approach, so it is harder to give a holistic evaluation mark. Also through time, some of the approaches that are considered in the evaluation may change, because they are still under development. Nevertheless, this should give to the reader a general perception of the maturity and practical orientation of each approach.

PI: Platform Independency. The last evaluation criteria that we consider is the dependency of the approach on the technological aspect of the Web application and its underlying platform. Namely we consider here whether the approach is more a set of principles and methods that can be applied regardless of the platform under which the resulting application will run. This should give to the reader an idea of how well the given approach can fit under his specific technological requirements and how much it can bend in order to meet them.

In table 6.2 we summarize the comparative evaluation results for each of the approaches and works that were discussed in chapter 3 (State of the Art and Related Works). We give three types of values for each criteria. Y meaning Yes, as the approach does include sufficient support for that criterias aspects. N meaning No, as the approach does not include any support at all for the corresponding criterias aspects. P meaning Partially, in the case that the given approach has some support, or at least deals with some of the criterias aspects, although that is not sufficient to be fully considered as supported by that approach.

Summarizing this section, we could say that each of the studied approaches have their strengths and weaknesses. The WebML, as the most mature approach is most ahead in terms of covering needed aspects of the Web engineering from a modeling perspective, although its biggest disadvantage is the proprietary development of its tools that do not allow its development according to the needs of general community. Also WebML does not have appropriate expressiveness for some parts of the user interface and business process modeling aspects. UML on the other hand is most advanced in all of the modeling aspects, but it lacks the expressiveness needed for the Web engineering domain, for example user interfaces, therefore it cannot be used as standalone for an end-to-end Web application development. UWE is also enough mature and does pretty well in large-scale Web engineering projects. Most importantly the notions of UWE allow an iterative approach of application design and development, therefore addressing different concerns at different levels of detail on different stages. The lack of appropriate transformation techniques for its models is probably the biggest disadvantage of UWE. The CMS modeling language has

Table 6.2: Comparative evaluation criteria and their support by our and other related approaches and languages

Approach	**DM**	**MAB**	**NPF**	**MUI**	**TM**	**CGA**	**PI**
Web Modeling Language (WebML)	Y	P	P	Y	P	P	Y
Unified Modeling Language (UML)	Y	Y	Y	N	Y	N	Y
UML-based Web Engineering (UWE)	Y	Y	Y	P	Y	N	Y
CMS Modeling Language (CMS-ML)	Y	Y	N	Y	Y	P	Y
Web Engineering Method (WEM)	Y	P	Y	P	Y	N	Y
Two-way Metamodeling Approach	Y	P	Y	Y	Y	P	Y

DM — Support for Domain Modeling

MAB — Ability to Model Applications Behavior

NPF — Support for Navigation and Page Flow

MUI — Modeling of User Interfaces

TM — Transformation of Models

CGA — Completeness of Generated Application

PI — Platform Independency

its strength in the split of two levels, addressing needs of technical and non-technical user by different notions. However, the lack of expressiveness for the navigation flow makes it limited in terms of usage for Web applications that are heavily based on the presentation properties. The WEM suffers from a complex structure of its underlying modeling elements. , on-going evaluation projects that are being carried out via WEM show for promising results and improvements in the future. Our proposed approach stands out in content-based applications that are heavily based on the presentation properties. However, as previously discussed, the approach and tools will require further development to reduce the need for manual operations leading to a completeness of the delivered application.

Chapter 7

Conclusions and Discussion

This thesis focuses in improving the state of the art in the domain of Web engineering by using model-driven methodologies. The aim of the research is to design an approach, supported by a methodology and tools, for practical usage in Web engineering projects, specifically for the creation of content-based Web applications (CBWA). This approach focuses to reduce the communication gap between the stakeholders involved in the project. Moreover, it aims to reduce the required effort for the production of the final Web application's components and content.

The research has evolved around the following two specific problems:

Problem P1
"To define model(s) of the Web application that will allow the users to communicate conceptually on the same level, while also allow automation of some parts of the application development."

Problem P2
"To design a methodology for the creation of a content-based Web applications that will be enough simple for practical usage by the non-technical user, while still enable the technical user to adapt and extend it in a natural way."

7.1 Conclusions

To find a solution to the above-mentioned problems we have based the research on fundamentals coming from natural processes and best practices learned during real-world projects. We focus on reducing overhead and costs of adoption of the involved methodology by using existing artefacts as a basis for communication between stakeholders and for the generation of other artefacts during the development process. We apply automatization techniques to parts of the process that are common in most projects, such as the generation of structure and initial content of the CMS application. During our research we have studied related works, existing methodologies and analyzed their strengths and weaknesses to learn from what has been done already in the same domain.

Based on the acquired information, we have proposed a two-way metamodel-based approach that enables (semi-)automated creation of CBWA and its effective administration in the future. The first part of the approach concerns a methodology for the requirements-gathering phase. We propose the usage of a presentation prototype as the conceptual model of the CBWA to improve communication between stakeholders at the initial phases of the project (bottom-up phase). The second part deals with the application logic and content management. We have proposed a generic metamodel and a technique for producing the CMS automatically by parsing the HTML prototype, using some predefined rules and reverse engineering principles based on common patterns from popular CMS solutions in the Web engineering domain (top-down phase). The third part of the approach deals with the maintenance process of the Web application in the future. We propose the usage of the Web application's presentation as the CMS interface for common content administration operations. Within the research we have developed supporting proof-of-concept tools for this approach. These tools involve some platform-specific attributes, however the principles, of which they consist, can be applied regardless of the underlying technologies, respectively the tools can be further developed to support more platforms and technologies for the development of CBWA.

The approach has been applied and evaluated in real-world projects from a qualitative perspective as well as in simulated experiments from a quantitative perspective. Also a comparative evaluation between the presented proposal and related works is presented. The results show that a wider application of the methodology and tools in practice is promising and principles of the methodology can improve the state of the art in the domain of CBWA development, namely by improving the communication process between stakeholders and reducing the overall effort required for the Web application creation and operation.

There are certain constraints and limitations that were identified during the evaluation process. These constraints come from the used concepts in the research and the type of the projects that the approach has been optimized for, namely the content-based Web applications. These constraints impose further possibility for the development of the proposed approach, which may serve as an inspiration for further future work within our research group, or elsewhere in the Web engineering community. As discussed previously, the approach proposed in this thesis has been applied and optimized for content-based Web applications, as one of the most widely spread types of application in the Web engineering discipline. However, it is important to point out that the *principles* that have been introduced are not bound to this type of application and they can be applied to any Web application.

7.2 Discussion

In the next following paragraphs we will discuss some aspects of the proposed approach with regards to their strengths and weaknesses as they have been identified during the research, aiming to give an answer to the research question Q3.

For the first part, related to the bottom-up phase, using of a presentation prototype as a conceptual model for automatic transformation, the principle was evaluated very well. Little or no modification would be required, when applying this principle for another type of Web application containing at least some content-based parts. This comes due to the fact that any kind of Web application has a front-end presentation that is accessible to the users for the interaction with the application's back-end and dynamic operations. For certain types of applications that involve concepts not representable (or visible) in the front-end, another intermediate level of modeling may be introduced, to which similar code-generation techniques can be applied for the transformation of the models into application assets.

As per the underlying metamodel, the delivered results have performed well, fulfilling the expectations from this work. For another type of application than those that are content-based, the differences and needs for modification however would be considerable. This is because of the very basic principle of metamodeling itself, which is to abstract concepts of the system into models that can be generalized and used in many instances. The metamodel components are directly related with entities of typical content-based Web applications. Although on a high-level the generic metamodel, which has been defined, includes components that are common for more types of applications (users, content and navigation) they are generally not enough reliable to model another type of applications, as they would lack the expressiveness required to cover some application-specific concepts. Therefore, further development for the enrichment of the metamodel imposes possible future work in order to increase its strength and wider application possibilities.

With regards to the algorithm and parser tools (supporting the top-down phase of the approach), some application-specific concepts related to the content-based types of application are involved. The patterns and knowledge used for their creation are based on processes that are bound to content-based applications. However, the principle as such is applicable elsewhere. Generating backend logic for any type of application is possible based on patterns and predefined heuristic rules that can be observed in similar projects. Further analysis would be required for the purpose of finding general rules that may be applicable throughout the Web engineering discipline projects, regardless of the application type. Another specific aspect of the tools described in this research is related to the used technologies and programming languages. Their choice has been made pragmatically, due to the fact that proving of the concepts has been one of the key requirements since the beginning of the research, which need not be too much theoretical. Being platform specific is the only possible option when trying to develop such tools in practice. On the other hand the used technologies and languages are very popular in the real-world of Web engineering projects as well. In the case that other technologies would be required, portation of the algorithm and tools is possible without any loss of applicability as Web-related technologies are quite similar in terms of expressiveness, architecture and purposes. The main difference would be in the semantics and syntax. Finally, for the part of administration of content through the presentation, we could say that the principle is also applicable in several types of Web applications, because most of them involve content creation and its presentation at a certain point. The difference here would be in the users and their roles in the creation and consumption of content. Specific concepts related to the type of application would affect widely the benefits of such approach. This is also a potential area for further research and development, especially having shown within this thesis that the principle of simplification and improvement of the usability in any area of the application operation and administration process is a key factor for the successful embracement and application of the methodology in practice. For the content-based type of Web applications as such, further development would be helpful in finding more patterns and heuristic rules from real-world projects and improving applicability of the algorithm in more specific environments, while keeping a low number of manual operations required. Also, as mentioned previously, portation of tools for other common Web-related technologies and programming languages would also be useful to the community and it represents a potential for wider acceptance of the methodology in the real-world.

7.3 Summary

From an overall perspective the proposed approach performs well, when it is properly applied in the domain of content-based Web applications. As it was shown in the evaluation phase, using the proposed methodology for the requirements gathering phase, reduces the effort required for the proper communication between stakeholders, while still allowing them to fulfill their needs. Moreover, such an approach allows for the subsequent usage of the delivered prototype for a partially automated parsing process that delivers the application's content administration system and structure, which is understandable for the non-technical stakeholders. This gives them more freedom and less dependency from the technical team-members, reducing also the overall effort for the creation of the Web application. It also gives possibilities for further incremental functionality development, because the involved model of the application is very generic. From this point of view we consider that the thesis outcome has achieved positive answers to the defined problems **P1** and **P2** and has improved the situation in the area of research.

Besides this general achievement the thesis outcome also contributes with several insights in both the scientific and also practical perspectives of Web engineering projects. Namely we can mention, the "decoupling" of the administration interface from the application's structure and bringing it closer together to the presentation model, which is more intuitive and delivers higher efficiency in content administration process. This idea is quite unique in the domain and has extremely high potential once it can be properly integrated on a large scale.

Another benefit of this work is also that it proves the fact that combining low-level and high-level concepts within the *same process* is possible without loss of expressivity, which is needed in most cases. Specifically, although starting with the presentation prototype that can be arbitrarily exact and complete, the combination of this prototype with a generic representation of the applications model still leaves enough flexibility for a proper definition and finalization of the required Web application's components and content. Such approach can be exploited in many other areas, not just in the Web engineering, but also in other industries, where appropriate.

Finally we can say that the completion of this thesis has brought several beneficial information to the research area. After several years of development of the approach, the related methodology and tools, there is always room for more improvement. We hope that the outcome of this thesis will serve as a "spark" for future research works that will take advantage of the achieved results and will try to further move the impact of our work to the prosperity of the community.

Bibliography

[1] Action Research Method, Wikipedia.
https://en.wikipedia.org/wiki/action_research, (Last accessed December 2014).

[2] C. Benevolo and S. N. Evaluation of content management systems (cms): a supply analysis. In *Proceedings of the 13th European Conference on Information Technology Evaluation*, 2007.

[3] Best Free Responsive HTML5 CSS3 Templates.
http://www.html5xcss3.com/, (Last accessed December 2014).

[4] B. Boehm and A. Malik. Quantifying requirements elaboration to improve early software cost estimation. *Information Sciences*, 2009.

[5] B. Boiko. *Content Management Bible (2nd Edition).* Wiley Publishing, Inc, 2004.

[6] Bultwith Technology Lookup.
http://builtwith.com/, (Last accessed December 2014).

[7] A. Byron, A. Berry, N. Haug, et al. *Using Drupal.* O'Reilly Media, 2008.

[8] S. J. B. Castro, R. G. Crespo, and G. V. H. M. Antipatterns: a compendium of bad practices in software development processes. *International Journal of Interactive Multimedia and Artificial Intelligence*, pages 41–46, 2011.

[9] S. Ceri, M. Brambilla, and P. Fraternali. The history of webml lessons learned from 10 years of model-driven development of web applications. *Lecture Notes in Computer Science, Conceptual Modeling: Foundations and Applications*, pages 273–292, 2009.

[10] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, and M. Matera. *Designing Data-Intensive Web Applications.* Morgan Kaufmann, 2003.

[11] CMS Matrix.
http://www.cmsmatrix.org/, (Last accessed December 2014).

[12] Comparison of Content Management Systems.
http://en.wikipedia.org/wiki/comparison_of_content_management_systems, (Last accessed December 2014).

[13] J. Conallen. *Building Web Applications with UML.* Addison-Wesley, 2000.

[14] M. G. Danny and E. Visser. Weaving web applications with webdsl: Demonstration. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, 2009.

[15] Drupal - Open-source CMS.
https://www.drupal.org/, (Last accessed December 2014).

[16] M. Epner. Poor project management number-one problem of outsourced e-projects. *Research Briefs, Cutter Consortium*, 2010.

[17] Evolution of the Web.
http://www.evolutionoftheweb.com/, (Last accessed December 2014).

[18] P. Fraternali, G. Rossi, and S.-F. F. Rich internet applications. *IEEE Internet Computing*, pages 9–12, 2010.

[19] Free HTML5 Templates.
http://www.freehtml5templates.com/, (Last accessed December 2014).

[20] W. Gibbs. *Software chronic crisis.* Scientific American, 1994.

[21] High Productivity Web and Mobile App Dev Platform — WebRatio.
http://www.webratio.com/portal/content/en/home, (Last accessed December 2014).

[22] HTML5UP, Responsive HTML5 and CSS3 Site Templates.
http://html5up.net/, (Last accessed December 2014).

[23] Internet Live Stats.
http://www.internetlivestats.com/, (Last accessed December 2014).

[24] ISO/IEC 9126-1:2001 Software Engineering, Product Quality.
http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749, (Last accessed December 2014).

[25] Joomla! The CMS Trusted By Millions for their Websites.
http://www.joomla.org/, (Last accessed December 2014).

[26] G. Kappel, B. Prll, S. Reich, and R. W. *Web Engineering: The Discipline of Systematic Development of Web Applications.* Wiley, New York, 2006.

[27] N. Koch, A. Knapp, et al. Uml-based web engineering: An approach based on standards. *Web Engineering: Modelling and Implementing Web Applications*, pages 157–191, 2008.

[28] C. Kroiss, N. Koch, and A. Knapp. Uwe4jsf: A model-driven generation approach for web applications. *Web Egineering, Lecture Notes in Computer Science*, 5648:493–496, 2009.

[29] Magic Draw, Wikipedia.
https://en.wikipedia.org/wiki/magicdraw, (Last accessed December 2014).

[30] M. Maristella, F. Rizzo, and G. Carughi. Web usability: Principles and evaluation. *Web Engineering*, pages 143–180, 2006.

[31] A. McDonald and R. Welland. Agile web engineering process: Perceptions within a fortune 500 financial services company. *Journal of Web Engineering*, 4:283–312, 2005.

[32] E. Mendes, I. Watson, C. Triggs, et al. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8:163–196, 2003.

[33] Meta-Object Facility - Wikipedia.
https://en.wikipedia.org/wiki/meta-object_facility, (Last accessed December 2014).

[34] N. Moreno, J. R. Romero, and A. Vallecillo. An overview of model-driven web engineering and the mda. *Web Engineering: Modelling and Implementing Web Applications, Human-Computer Interaction Series*, pages 353–382, 2008.

[35] J. Nielsen. Usability for the masses. *Journal of Usability Studies*, 2005.

[36] B. Nuseibeh, C. Haley, and C. Foster. Securing the skies: In requirements we trust. *IEEE Computer*, pages 64–72, 2009.

[37] Object Management Group, OMG.
http://www.omg.org, (Last accessed December 2014).

[38] Overview of Action Research Methodology, Obrien R.
http://web.net/ robrien/papers/arfinal.html, (Last accessed December 2014).

[39] O. Pastor, J. Fons, and S. Abrahao. Conceptual modelling of web applications: The oows approach. *Web Engineering*, pages 277–302, 2006.

[40] R. S. Pressman. Applying web engineering. *Software Engineering: A Practitioner's Approach*, 2005.

[41] J. M. Rivero, J. Grigera, G. Rossi, et al. Towards agile model-driven web engineering. *Lecture Notes in Business Information Processing*, 107:142–155, 2012.

[42] J. S. Saraiva and A. Rodrigues. A reference model for the analysis and comparison of mde approaches for web-application development. *Journal of Software Engineering & Applications, Scientific Research*, 3:419–425, 2010.

[43] J. S. Saraiva and A. R. Silva. Evaluation of mde tools from a metamodeling perspective. *Principle Advancements in Database Management Technologies: New Applications and Frameworks*, pages 105–131, 2009.

[44] J. S. Saraiva and A. R. Silva. Web-application modeling with the cms-ml language. *Symposium of Informatics*, 2010.

[45] D. Schwabe and G. Rossi. Model-based web application development. *Web Engineering*, pages 303–333, 2006.

[46] R. Severdia and K. Crowder. *Using Joomla: Building Powerful and Efficient Web Sites*. OŔeilly Media'Reilly Media, 2009.

[47] B. Shneiderman and C. Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer- Interaction*. Addison-Wesley, 4th edition (international) edition, 2005.

[48] J. Souer, T. Kupers, R. Helms, and B. S. Model-driven web engineering for the automated configuration of web content management systems. *Web Engineering Lecture Notes in Computer Science*, 5648:121–135, 2009.

[49] The Web Modeling Language.
http://www.webml.org, (Last accessed December 2014).

[50] UML-based Web Engineering.
http://uwe.pst.ifi.lmu.de/index.html, (Last accessed December 2014).

[51] Unified Modeling Language, Object Management Group.
http://www.omg.org/spec/uml/, (Last accessed December 2014).

[52] Unified Modeling Language, Wikipedia.
https://en.wikipedia.org/wiki/unified_modeling_language, (Last accessed December 2014).

[53] UWE - MagicUWE.
http://uwe.pst.ifi.lmu.de/toolmagicuwe.html, (Last accessed December 2014).

[54] M. Vasko, E. Oberortner, and S. Dustdar. Collaborative modeling of web applications for various stakeholders. In *Proceedings of the 9th International Conference on Web Engineering (ICWE)*, 2009.

[55] W3Techs - Extensive and Reliable Web Technology Surveys.
http://w3techs.com/, (Last accessed December 2014).

[56] D. Wallace, I. Raggett, and A. J. *Extreme Programming for Web Projects*. Addison-Wesley, 2002.

[57] Wordpress, Blog Tool, Publishing Platform and CMS.
https://wordpress.org/, (Last accessed December 2014).

[58] J. Wright and J. Dietrich. Survey of existing languages to model interactive web applications. In *Proceedings of the 5th Asia-pacific Conference in Conceptual Modeling*, 2008.

Publications of the Author

Relevant Publications in Impacted Journals

[A.1] C. V. Nguyen (80%), X. Qafmolla (10%) and K. Richta (10%). Domain Specific Language Approach on Model-driven Development of Web Services. *Acta Polytechnica Hungarrica*, pp. 121-138, 2014 (vol. 11), ISSN 1785-8860.

Relevant Publications in Reviewed Journals

[A.2] X. Qafmolla (80%), C. V. Nguyen (10%) and K. Richta (10%). Metamodel-based Generation of Web Content Management Systems. *International Journal on Information Technologies and Security*, pp. 17-30, 2014 (vol. 4), ISSN 1313-8251.

[A.3] C. V. Nguyen (60%) and X. Qafmolla (40%). Model Transformation in Web Engineering and Automated Model Driven Development. *International Journal of Modeling and Optimization*, pp. 7-12, 2011 (vol. 1), ISSN 2010-3697.

The paper has been cited in:

- D. Arvidsson and F. Persson. Evaluating a Design Pattern for Black-Box Testing in Executable UML, *Master Thesis, Chalmers University of Technology, University of Gothenburg*, Sweden, 2012.
- M. Rahmouni and M. Samir. MDA-based ATL transformation to generate MVC 2 web models, *Proceedings of Transformation 2011 Conference*, 2011.

Relevant Publications Indexed by ISI/Scopus

[A.4] X. Qafmolla (50%) and C. V. Nguyen (50%). Automation of Web Services Development Using Model Driven Techniques. *Proceedings of the 2nd IEEE International Conference on Computer and Automation Engineering (ICCAE 2010)*, Singapore, Singapore, pp. 190-194, 2010 (vol. 3), ISBN 978-1-4244-5585-0.

The paper has been cited in:

- N. Tavares and V. Samyr. Towards Interoperability to the Implementation of RESTful Web Services: A Model Driven Approach, *ICONS 2014, The 9th International Conference on Systems*, 2014.
- B. Simon, B. Goldschmidt and K. Kondorosi. A Metamodel for the Web Services Standards, *Journal of grid computing*, pp. 735-752, 2013.
- N. Tavares, and V. Samyr. A Model Driven Approach for the Development of Semantic RESTful Web Services, *Proceedings of International Conference on Information Integration and Web-based Applications & Services, ACM*, 2013.
- L. O. Chih-Min and H. Sun-Jen. Applying Model-Driven Approach to Building Rapid Distributed Data Services, *Transactions on Information and Systems*, pp. 2796-2809, 2012.
- M. A. O. Mukhtar, M. F. B. Hassan and J. B. Jaafar. Optimizing Method to Provide Model Transformation of Model-driven Architecture as Web-based Services, *International Conference on Computer & Information Science (ICCIS)*, Kuala Lumpur, Malaysia, pp. 874-879, 2012, ISBN 978-1-4673-1937-9.
- J. Mtsweni, E. Biermann and L. Pretorius. SemServ: Facilitating the Implementation of Intelligent Semantic Services, *Proceedings of the Ninth International Network Conference (INC 2012)*, 2012.
- M. A. O. Mukhtar, A. Azween and G. D. Alan. A Proposed Compiler to Integrate Model Driven Architecture with Web ServicesRoad Map. *International Journal of Computer Applications*, pp. 1-7, 2011.
- Z. Zohrevand, Y. M. Bibalan and R. Ramsin. Towards a Framework for the Application of Model-Driven Development in Situational Method Engineering, *18th Asia Pacific Software Engineering Conference (APSEC)*, Ho Chi Minh, Vietnam, pp. 122-129, 2011, ISSN 1530-1362.

[A.5] C. V. Nguyen (67%) and X. Qafmolla (33%). On Instance-model Querying and Meta-model Transformation. *Proceedings of the 2010 International Conference on Software Engineering (IAENG)*, Hong Kong, China, pp. 710-715, 2010 (vol. 1), ISSN 2078-0958.

Other Relevant Publications

[A.6] X. Qafmolla (80%) and C. V. Nguyen (20%). Semiautomatic Generation of Content-based Web Applications: An Evaluation of Supporting Tools. *5th International Symposium on Information Management in a Changing World (IMCW2014)*, Antalya, Turkey, 2014.

[A.7] X. Qafmolla (50%) and C. V. Nguyen (50%). A Two-Way Meta-Modelling Approach in Web Engineering. *Procedia Information Technology and Computer Science, 3rd*

World Conference on Information Technology, Barcelona, Spain, 2012, ISSN 2147-5105.

[A.8] X. Qafmolla (50%) and C. V. Nguyen (50%). Model-driven Practices in Web Engineering. *Proceedings of Objekty 2009 Conference*, Hradec Králové, Czech Republic, pp. 185-195, 2009 (vol. 1), ISBN 978-80-7435-009-2.

[A.9] C. V. Nguyen (50%) and X. Qafmolla (50%). Towards Automated Model-driven Development with Model Transformation and Domain Specific Languages. *Proceedings of The International Conference on Computer and Software Engineering*, Kuala Lumpur, Malaysia, pp. 128-134, 2012, ISBN 978-93-82242-22-2.

[A.10] C. V. Nguyen (70%) and X. Qafmolla (30%). On Model Transformation Methods and Testing of Model Transformation to Support Automated Model Driven Development. *Proceedings of the 3rd International Conference on Computer and Automation Engineering*, Chongqing, China, pp. 171-175, 2011 (vol. 1), ISBN 978-1-4244-9462-0.

[A.11] C. V. Nguyen (50%) and X. Qafmolla (50%). Agile Development of Platform Independent Model in Model Driven Architecture. *Proceedings of theThird International Conference on Information and Computing (ICIC)*, Beijing, China, pp. 344-347, 2010 (vol. 2), ISBN 978-1-4244-7081-5.

The paper has been cited in:

- F. J. Pereda and A. Molina. Model driven architecture for engineering design and manufacturing, *Management and Control of Production and Logistics*, pp. 400-407, 2013.
- G. Botturi et al. Model-driven Design for the Development of Multi-platform Smartphone Applications, *Specification & Design Languages (FDL), 2013 Forum on IEEE*, pp. 1-8, 2013.
- E. Brandtzaeg. A DSL for Model-based Realization of Applications in the Cloud, *Master Thesis, University of Oslo*, Norway, 2012.

[A.12] C. V. Nguyen (50%) and X. Qafmolla (50%). Metamodel Transformation with Kermeta. *Proceedings of Objekty 2008 Conference*, Žilina, Slovakia, pp. 109-116, 2008, ISBN 978-80-8070-927-3.

Other Less Relevant Publications

[A.13] C. V. Nguyen (50%) and X. Qafmolla (50%). A Practical Approach on Implementing Scrum in Mid-size Companies. *Proceedings of Objekty 2009 Conference*, Hradec Králové, Czech Republic, pp. 185-195, 2009 (vol. 1), ISBN 978-80-7435-009-2.

[A.14] T. Fluori (50%) and X. Qafmolla (50%). Linear Pattern Matching with Swaps for Short Patterns. *Proceedings of the 10th International PhD Workshop on Systems*

and Control, Hluboká nad Vltavou, Czech Republic, pp. 109-116, 2009, ISBN 978-80-903834-3-2.

Appendix A

Anotace

Tato disertacní práce se zabývá novým modelem řízeným přístupem (model-driven approach) ve webovém inženýrství. Tento přístup má za cíl zmenšit komunikační propast, která existuje mezi zainteresovanými subjekty v počátečních fázích webových projektů. Zároveň je jeho cílem usnadnit celkové vývojové aktivity potřebné na přípravu finální webové aplikace. Navržený přístup je ilustrován na oblasti webových aplikací pro správu obsahu (content-based Web applications) a je podporován demonstrativními softwarovými nástroji (proof-of-concept tools) a příslušnou metodikou.

V oblasti webového inženýrství již existuje několik modelem řízených přístupů, které byly vyvinuty v posledním desetiletí. Nicméně, většina z nich je už zastaralých a nejsou již používané. Ty, co jsou stále aplikované, většinou vyžadují celkem specifické prostředí pro jejich použití a zároveň představují velké náklady při zavádění do praxe. Hlavním důvodem je, že modelovací část aplikovaných metod je orientovaná na přístup shora-dolů (top-down), tzn. na začátku se členové týmu zaměří na přípravu konceptuálního modelu s vysokou úrovní abstrakce. Nad tímto modelem jsou pak aplikované transformační techniky pro vygenerování finálních komponent. Takový přístup má svá úskalí. Za prvé, komunikace mezi zainteresovanými subjekty na vysoké úrovni abstrakce v ranných fázích projektu je obtížná a nepřesná, což přináší problémy a zvýšenou námahu při další práci v pozdějších fázích. Za druhé, samotná transformace abstraktního modelu je komplexní záležitost, obnáší velké množství chyb a potřebu častého ručního zásahu do jednotlivých komponent. Na rozdíl od těchto metod, je náš návrh založen na dvojím přístupu (two-way approach) při stvoření a transformaci modelů. Technický i netechnicky zdatní členové týmu využívají při přípravě požadavků konceptuální model založený na nízké úrovňi abstrakce, jako např. rychlý prototyp (rapid prototype) webové aplikace v jazyce HTML, nebo jiný podobný prototyp, pro fázi sdola-nahoru (bottom-up). Po několika iteracích, jakmile je vytvořený model dostatečně kompletní, je použit jako vstup pro (semi-)automatickou tvorbu příslušných komponent finální aplikace a jejího obsahu, tj. pro fázi shora-dolů (top-down).

Navržený přístup byl kvalitativně vyhodnocen v reálných případových studiích (real-world case-studies) a kvantitativně přes simulaci velkého množství projektů, použitím volně dostupných (open-source) šablon webových aplikací. Dosažené výsledky jsou slibné a představují velký potenciál pro zlepšení situaci v praxi, při aplikaci technik modelem

řízeného vývoje v oblasti webového inženýrství.

Klíčová slova:
Webové inženýrství, webové aplikace na správu obsahu, modelem řízený vývoj, generování kódu.