

Sebastian Heil

Web Migration Revisited: Addressing Effort and Risk Concerns

Doctoral Dissertations in Web Engineering and Web Science
Volume 5

Prof. Dr.-Ing. Martin Gaedke (Series Editor)

Sebastian Heil

Web Migration Revisited

Addressing Effort and Risk Concerns



TECHNISCHE UNIVERSITÄT
CHEMNITZ

**Universitätsverlag Chemnitz
2021**

Impressum

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über <https://www.dnb.de> abrufbar.



Das Werk - ausgenommen Zitate, Cover, Logo TU Chemnitz und Bildmaterial im Text - steht unter der Creative-Commons-Lizenz
Namensnennung 4.0 International (CC BY 4.0)
<https://creativecommons.org/licenses/by/4.0/deed.de>

Titelgrafik: Sebastian Heil

Satz/Layout: Sebastian Heil

Technische Universität Chemnitz/Universitätsbibliothek

Universitätsverlag Chemnitz

09107 Chemnitz

<https://www.tu-chemnitz.de/ub/univerlag>

readbox unipress

in der readbox publishing GmbH

Rheinische Straße 171

44147 Dortmund

<https://www.readbox.net/unipress/>

ISSN 2199-5354 print - ISSN 2199-5362 online

ISBN 978-3-96100-125-5

<https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-723455>



TECHNISCHE UNIVERSITÄT
CHEMNITZ

Faculty of Computer Science
Distributed and Self-Organizing Systems Group

Web Migration Revisited Addressing Effort and Risk Concerns

Dissertation
submitted in fulfillment of the
requirements for the degree of
Doktoringenieur (Dr.-Ing.)

by
Sebastian Heil M.Sc.

September 24, 2020

Dissertation Committee:
Prof. Dr.-Ing. Martin Gaedke
Prof. Dr. Maximilian Eibl
Assoc.-Prof. Maxim Bakaev, PhD

Sebastian Heil M.Sc.

Web Migration Revisited: Addressing Effort and Risk Concerns

Dissertation Committee:

Prof. Dr.-Ing. Martin Gaedke (Technische Universität Chemnitz),

Prof. Dr. Maximilian Eibl (Technische Universität Chemnitz)

Assoc.-Prof. Maxim Bakaev, PhD (Novosibirsk State Technical University)

Submitted on December 12, 2019

Defended on September 24, 2020

Technische Universität Chemnitz

Faculty of Computer Science

Distributed and Self-Organizing Systems Group

Straße der Nationen 62

09111 Chemnitz

Abstract

Web Systems are widely used and accepted due to their advantages over traditional desktop applications. Modernization of existing non-Web software towards the Web, however, is a complex and challenging task due to the characteristics of Legacy Systems. Independent Software Vendors are struggling to commence Web Migration because of the involved effort and risk. Through systematic field research and problem analysis, this situation is further analyzed, deriving a set of requirements that represent the effort and risk concerns and which are used to assess the state of the art in the field. Existing Web Migration research exhibits gaps concerning dedicated approaches for the initial phase and feasibility of the proposed strategies with limited resources and expertise.

This thesis proposes a solution to address the shortcomings outlined above and to support Independent Software Vendors to commence Web Migration, focusing on their concerns about effort and risk. The main idea is to provide a set of dedicated solutions to close the identified gaps in the form of a methodology and a supporting toolsuite that transfer paradigms successfully solving similar problems in other areas of computer science into the Web Migration domain. These solutions constitute the proposed approach called Agile Web Migration for SMEs (AWSM), consisting of methods, tools, principles, and formalisms for reverse engineering, risk management, customer impact control, and migration strategy selection.

The thesis describes the research on the devised ideas in the context of a collaboration project with an Independent Software Vendor. Applicability and feasibility of the concepts are demonstrated in several evaluation experiments, integrating empirical user studies and objective measurements. The thesis concludes with an evaluation based on requirements assessment and on the application of the solutions in the application scenario, and it provides an outlook towards future work.

Acknowledgment

I would like to express my gratitude to all who have accompanied me throughout the years of research as a PhD student, in particular to:

My supervisor Prof. Dr.-Ing. Martin Gaedke for introducing me to the challenging but exciting world of research, for his inspiring vision and ideas, and for giving me the opportunity to learn and grow under his guidance in a supportive and motivating environment that he created and to collaborate with many international researchers.

To all my colleagues and dear friends with whom I had the honor to collaborate on various research projects, who have widened my research interests and from whom I have learned so many things, I owe so much. Here, I would like to especially mention Maxim Bakaev, Mahda Noura, Stefan Wild, Alexey Tschudnowsky, Julian Grigera, José Matías Rivero, Bahareh Zarei, and Maximilian Speicher.

I am very grateful to all students and colleagues who contributed to my research presented in this thesis: Felix Förster, Valentin Siegert, Tobias Lang, Thomas Weber, Frank Siegel, Anna Scholz, Philipp Oehme, Markus Keller, Stefan Staude, and Hao Li, and to Ralph Sontag, who has always encouraged me and spent countless hours helping me to typeset this thesis. I cordially thank my friend Mahsa Sodeiri for helping me design the book cover for the print version of this thesis.

I thank all present and former colleagues at VSR – André Langer, Maik Benndorf, Michael Krug, Fabian Wiedemann, Jörg Anders, Markus Ast, Christoph Brandt, Dang Vu Nguyen Hai, Felix Kettner, and Verena

Traubinger – for inspiring and fruitful discussions, continuous knowledge sharing, and a friendly and welcoming atmosphere that has lead me through the harder times in my PhD student life. Prof. Dr.-Ing. Martin Gaedke did an excellent job in putting the VSR team together.

Thank you also to my three reviewers for their encouraging feedback and support to successfully write, defend, and publish this thesis, as well as to the members of the PhD commission, Prof. Dr.-Ing. Janet Siegmund, Dr. Andreas Müller, and Dr. Frank Seifert.

Also, I like to thank all student assistants and students advised by me for enabling me to focus on my research and teaching me how to communicate ideas and organize research activities.

Finally, I am very grateful to my mother for her ongoing encouragement, appreciation, understanding, love and care, and to Michalina Kunecka, Sven R. Kunze, Christian Jager, Bastian Weiß, Gabriela Bosetti, Radka Nacheva and many more of my friends whom I have not already mentioned above, who all supported me, raised me up and kept close contact in spite of the sometimes limited availability resulting from the PhD challenge.

Dissemination

In order to substantiate this work and comply with high academic standards, parts of the research results this dissertation is based on have been disseminated before. All relevant publications are listed in the following:

- Heil, Sebastian, Maxim Bakaev, and Martin Gaedke (2016). “Measuring and Ensuring Similarity of User Interfaces: The Impact of Web Layout”. In: *Web Information Systems Engineering – WISE 2016*. Ed. by Wojciech Cellary, Mohamed F. Mokbel, Jianmin Wang, et al. Cham: Springer International Publishing, pp. 252–260.
- Heil, Sebastian, Felix Förster, and Martin Gaedke (2018). “Exploring Crowdsourced Reverse Engineering”. In: *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*. Funchal, Portugal: SciTePress, pp. 147–158.
- Heil, Sebastian and Martin Gaedke (2016). “AWSM – Agile Web Migration for SMEs”. In: *Proceedings of the 11th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*. Rome, Italy: SciTePress, pp. 189–194.
- (2017). “Web Migration – A Survey Considering the SME Perspective”. In: *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*. Porto, Portugal: SciTePress, pp. 255–262.
- Heil, Sebastian, Valentin Siegert, and Martin Gaedke (2018). “ReWaMP: Rapid Web Migration Prototyping Leveraging WebAssembly”. In: *Web*

- Engineering: Proceedings of 18th International Conference on Web Engineering (ICWE2018)*. Ed. by Tommi Mikkonen, Ralf Klamma, and Juan Hernández. Caceres, Spain, pp. 84–92.
- Heil, Sebastian, Valentin Siegert, and Martin Gaedke (2019). “Crowd-sourced Reverse Engineering: Experiences in Applying Crowdsourcing to Concept Assignment”. In: *Evaluation of Novel Approaches to Software Engineering, Revised Selected Papers, Communications in Computer and Information Science*. Ed. by Ernesto Damiani, George Spanoudakis, and Leszek Maciaszek. Springer, pp. 215–239.

- Bakaev, Maxim, Sebastian Heil, Vladimir Khvorostov, and Martin Gaedke (2018). “HCI Vision for Automated Analysis and Mining of Web User Interfaces”. In: *Web Engineering: Proceedings of the 18th International Conference on Web Engineering (ICWE2018)*. Ed. by Tommi Mikkonen, Ralf Klamma, and Juan Hernández, pp. 136–144.
- (2019). “Auto-Extraction and Integration of Metrics for Web User Interfaces”. In: *Journal of Web Engineering* 17.6, pp. 561–590.
- Bakaev, Maxim, Sebastian Heil, Nikita Perminov, and Martin Gaedke (2019). “Integration Platform for Metric-Based Analysis of Web User Interfaces”. In: *Web Engineering: Proceedings of 19th International Conference on Web Engineering (ICWE2019)*. Ed. by Maxim Bakaev, Flavius Frasincar, and In-Young Ko. Daejeon, Korea: Springer, pp. 525–529.
- Bakaev, Maxim, Vladimir Khvorostov, Sebastian Heil, and Martin Gaedke (Sept. 2017a). “Evaluation of User-Subjective Web Interface Similarity With Kansei Engineering-Based ANN”. In: *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, pp. 125–131.

- (2017b). “Web Intelligence Linked Open Data for Website Design Reuse”. In: *Web Engineering: Proceedings of the 17th International Conference on Web Engineering (ICWE2017)*. Ed. by Jordi Cabot, Roberto De Virgilio, and Riccardo Torlone, pp. 370–377.
- Bakaev, Maxim, Tatiana A. Laricheva, Sebastian Heil, and Martin Gaedke (Oct. 2018). “Analysis and Prediction of University Websites Perceptions by Different User Groups”. In: *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*. IEEE, pp. 381–385.
- Rivero, José Matías, Sebastian Heil, Julián Grigera, Martin Gaedke, and Gustavo Rossi (2013). “MockAPI: An Agile Approach Supporting API-first Web Application Development”. In: *Web Engineering: Proceedings of the 13th International Conference on Web Engineering (ICWE2013)*. Ed. by Florian Daniel, Peter Dolog, and Quing Li. Aalborg, Denmark: Springer Berlin Heidelberg, pp. 7–21.
- Rivero, José Matías, Sebastian Heil, Julián Grigera, Esteban Robles Luna, and Martin Gaedke (2014). “An Extensible, Model-Driven and End-User Centric Approach for API Building”. In: *Web Engineering: Proceedings of the 14th International Conference (ICWE2014)*. Ed. by S. Casteleyn, G. Rossi, and M. Winckler. Vol. 8541. Toulouse, France: Springer, Cham, pp. 494–497.

Contents

1	Introduction	1
1.1	Situation	1
1.2	Motivation	6
1.3	Problem Statement	8
1.4	Scope	11
1.5	Contributions	13
1.6	Outline	14
1.7	Summary	15
2	Requirements Analysis	17
2.1	Scenario	17
2.1.1	Company & Software Development Characteristics	17
2.1.2	Legacy Codebase Characteristics	19
2.1.3	Migration Objectives	22
2.2	Problem Analysis	23
2.2.1	Research Process	24
2.2.2	Results	27
2.3	Requirements	29
2.3.1	Scope Requirements	31
2.3.2	Stakeholder Requirements	34
2.4	Summary	39
3	State of the Art	41
3.1	Standards and Reference Model	42
3.1.1	Architecture-Driven Modernization (ADM)	43
3.1.2	Reference Migration Process	47

3.2	Web Migration Approaches	49
3.2.1	Migration to SOA	51
3.2.2	Migration to Cloud	63
3.2.3	Web Systems Evolution	77
3.2.4	Migration to Web Applications	81
3.3	Analysis of Assessment Results	91
3.3.1	Results by Requirements	92
3.3.2	Results by Research Areas & Target Architectures	95
3.3.3	Results by Overall Methodologies Used	97
3.3.4	Results by Cross-cutting characteristics	99
3.4	Shortcomings of Existing Approaches	102
3.5	Summary	104
4	Addressing Effort and Risk Concerns in Web Migration	105
4.1	Research Objectives	105
4.2	Overview	108
4.3	Methods	110
4.3.1	AWSM Reverse Engineering Method	112
4.3.2	AWSM Risk Management Method	114
4.3.3	AWSM Customer Impact Control Method	116
4.4	Tools	118
4.4.1	AWSM Strategy Selection Decision Support Syst.	119
4.4.2	AWSM Annotation Platform	121
4.4.3	AWSM Rapid Web Migration Prototyping Tools .	123
4.4.4	AWSM Customer Impact Tools	124
4.5	Principles	127
4.6	Formalisms	129
4.6.1	Legacy System	130
4.6.2	Source Code Knowledge Model	131
4.6.3	Legacy User Interface	135
4.7	Summary	137

5 AWSM Reverse Engineering Method	139
5.1 Analysis	139
5.1.1 Situation and Challenges	140
5.1.2 Requirements	142
5.1.3 Related Work	142
5.2 Research Questions	145
5.3 Knowledge Rediscovery	145
5.3.1 Setup Process	147
5.3.2 Concept Assignment Process	149
5.3.3 Management and Usage Process	150
5.4 Annotation Platform	151
5.4.1 User Functionality	152
5.4.2 Integration in ISV environments	156
5.4.3 Queryable Legacy Knowledge Base Repres.	160
5.5 Crowdsourced Reverse Engineering	164
5.5.1 Automatic Crowdsourcing Task Extraction & Creation	169
5.5.2 Balancing Controlled Disclosure with Readability	170
5.5.3 Aggregation of Results and Quality Control	171
5.6 Evaluation	173
5.6.1 Assessment of Requirements	174
5.6.2 Experimental Evaluation of CSRE	175
5.6.3 Research Results	188
5.7 Summary	188
6 AWSM Risk Management Method	189
6.1 Analysis	190
6.1.1 Situation and Challenges	190
6.1.2 Requirements	192
6.1.3 Related Work	193
6.2 Research Questions	195

6.3	Rapid Web Migration Prototyping	196
6.3.1	Overview	196
6.3.2	Integration with Ongoing Development	199
6.3.3	API Prototyping using MockAPI	200
6.4	Reuse of Business Logic	203
6.4.1	Web Migration Support Technology Assessment .	203
6.4.2	Rapid Web Migration Prototyping leveraging WebAssembly	206
6.4.3	Rapid Web Migration Prototyping Toolchain . .	210
6.4.4	Guided Web Migration Prototyping Assistance .	213
6.5	User Interface Transformation	216
6.5.1	Mapping Legacy to Web Layouts	217
6.5.2	User Interface Transformation through Evolutionary Optimisation	219
6.5.3	User Interface Transformation Tool	223
6.6	Evaluation	231
6.6.1	Assessment of Requirements	232
6.6.2	Experimental Evaluation of ReWaMP	233
6.6.3	Experimental Evaluation of RWMPA	242
6.6.4	Experimental Evaluation of UI Transformer . . .	247
6.6.5	Research Results	254
6.7	Summary	255
7	AWSM Customer Impact Control Method	257
7.1	Analysis	257
7.1.1	Situation and Challenges	258
7.1.2	Requirements	260
7.1.3	Related Work	261
7.2	Research Questions	264
7.3	UI Similarity Analysis	264
7.3.1	Visual Analysis of UI Similarity	265

7.3.2	Calibration	268
7.3.3	Computing Visual UI Similarity	270
7.3.4	Integration with Ongoing Development	275
7.4	Visual UI Element Detection	276
7.4.1	Conceptual Model	277
7.4.2	Visual UI Element Detector	279
7.5	Evaluation	284
7.5.1	Assessment of Requirements	284
7.5.2	Experimental Evaluation of AWSM:CI	285
7.5.3	Research Results	296
7.6	Summary	297
8	Evaluation	299
8.1	Requirements Evaluation	299
8.1.1	Scope Requirements	299
8.1.2	Stakeholder Requirements	301
8.2	Comparison with State of the Art	305
8.3	Application Scenarios	312
8.3.1	Reverse Engineering and Annotation Platform in ISV Production Codebase	312
8.3.2	Rapid Web Migration Prototyping of Medical Appointment Scheduling	314
8.3.3	Layout Similarity Measurement in x.concept	316
8.4	Summary	317
9	Conclusion and Outlook	319
9.1	Thesis Summary	319
9.2	Lessons Learned	322
9.3	Detailed Contributions	324
9.4	Ongoing and Future Work	325
9.4.1	Methodology and Toolsuite Improvements	326

9.4.2	Open Questions	329
A	Scenario Materials	333
B	Problem Analysis Materials	335
C	Solution Materials	339
D	AWSM:RE Materials	341
D.1	SCKM Ontology	341
D.2	SCKM Ontology SWRL Rules	345
E	AWSM:RM Materials	349
F	ReWaMP Evaluation Materials	353
F.1	ReWaMP Questionnaire	354
F.2	ReWaMP Evaluation Data	356
G	RWMPA Evaluation Materials	359
G.1	RWMPA Questionnaire	360
G.2	RWMPA Evaluation Data	362
H	UI Transformer Evaluation Materials	365
H.1	UI Transformer Questionnaire	365
H.2	UI Transformer Evaluation Data	366
I	AWSM:CI Evaluation Materials	375
I.1	Visual Similarity Questionnaire	375
I.2	Visual Similarity Task Lists	376
I.3	Visual Similarity Evaluation Data	377
Glossary		379
Bibliography		387

List of Acronyms	419
List of Figures	425
List of Tables	429
List of Listings	433
List of Algorithms	435

Introduction

1

The topic of this thesis is the application of scientific approaches to address effort and risk concerns for Independent Software Vendors (ISVs)¹ when moving from their existing non-Web desktop Legacy Systems to modern Web-based systems. The introduction motivates the need for a dedicated approach to address the initial unwillingness and resistance of these companies to commence a Web Migration. High effort and risk due to the lack of a dedicated approach supporting the initial phase of a Web Migration are identified as main challenges. This thesis contributes to the solution of these problems by providing a set of methods and technical infrastructure to support the initiation of Web Migration.

1.1 Situation

World Wide Web (Web)-based Systems (Web Systems) are widely used and familiar for users, as they offer significant advantages over traditional Desktop Applications such as instant deployment, a common standards-based target development platform, and a high level of interactivity. This section provides a brief overview of the current situation of (Web Systems) and migration to (Web Systems), highlighting the benefits of migrating to the Web and major research areas. *Web Migration* is a particular type of Software Migration – the process of moving an

¹<https://www.gartner.com/it-glossary/isv-independent-software-vendor/>
Retrieved: 7.12.2019

existing software system from one environment to another (IEEE Computer Society, 2014) – moving a *non-Web source system* to a *Web-based target environment*. In terms of the joint International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) and Institute of Electrical and Electronics Engineers (IEEE) 14764 standard for software maintenance (ISO/IEEE, 2006), it is *adaptive and perfective maintenance* adapting software systems for Web-based environments and improving functionality and maintainability. Several related concepts exist in the context of *Web-based environments*: *Web Sites*, *Web Applications*, *Web Services*. Since Web Migration research addresses different types of Web-based systems as migration targets, we use the term *Web System* as umbrella term following (Kienle and Distante, 2014) and adapting definitions from (Gaedke, 2000; Kappel et al., 2006).

Definition 1: Web System

A Web System is a software system based on technologies and standards of the World Wide Web Consortium (W3C) that provides Web specific resources.

In contrast to this broad definition, we use the term Web Application for a specific type of Web System that includes a Web-based user interface. Web Applications, as defined by Kappel et al., are the main focus of this thesis as Web Migration target:

Definition 2: Web Application (Kappel et al., 2006)

A Web Application is a software system based on technologies and standards of the W3C that provides Web specific resources such as content and services through a user interface, the Web browser.

One of the main characteristics of *Web Engineering* – the “application of systematic, disciplined, and quantifiable approaches to development, operation, and maintenance of Web-based applications” (Deshpande et

al., 2002) – is the continuous evolution of Web technologies (Gitzel et al., 2007). All steps in this evolution represent targets of Web Migration and major research areas within Web Migration research: Web Application migration, migration to Web Services, Cloud migration, and migration within these steps, Web Systems Evolution.

Modern Web Applications are the result of this evolution (Kienle and Distante, 2014), which started with the proposal of Hypertext as Information Management system by Tim Berners Lee in March 1989 (Berners-Lee, 1990). Quickly evolving from the original textual content-focused system, the Web started to be perceived as “a universal, standards-based integration platform” (Knorr, 2003) and the introduction of the term *Web Application*, indicating that their user experience became similar² to that of *Desktop Applications* (Kienle and Distante, 2014).

Composing applications from *Web Services*, i.e., reusable pieces of functionality invokable via HyperText Transfer Protocol (HTTP), soon became a focus in both research and industry (Kienle and Distante, 2014). The architectural paradigm of Service-oriented Architecture (SOA) (MacKenzie et al., 2006) is closely related to this development and an ecosystem of Web protocols (Chinnici et al., 2007; Mitra, Lafon, et al., 2003; OASIS, 2007) was created.

Cloud Computing allows Web Systems to be built based on scaleable sets of rapidly provisioned, shared resources (Mell and Grance, 2011). The related *Software as a Service (SaaS) paradigm* not only influenced the architecture of Web Applications, but also impacted end users’ perception and expectations of software (Politis, 2017;

²cf. features defined in (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012)

Fowley, Elango, et al., 2017) allowing ubiquitous access without installation. SaaS is the largest segment of the public cloud market with increasing tendency (Statista, 2018a).

High popularity and use of mobile devices (Statista, 2018b) in private and work contexts have changed end users' expectations with regard to high levels of interactivity and social interactions (Bitkom, 2013), putting pressure on ISVs to renew their existing systems accordingly.

The above evolution of Web Systems has brought various advantages over traditional Desktop Applications making Web Migration desirable for ISVs, such as:

- Standardized Target Development Platform
- Instant Deployment
- Low Access Requirements
- High level of Interactivity and Social Interactions

In contrast to Desktop Applications (Puder, 2004), Web Applications enable ISVs to focus on one single target development platform based on *open standards*. This significantly reduces the development workload to provide software for many different platforms and devices, since only one version of the software based on one set of technologies is needed (O'Reilly, 2007). Development efficiency is increased by only requiring one development infrastructure and set of tools. *Open Web Standards* like HTTP, HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript form the core of Web Applications, lowering the risk of technology depreciation.

The Web's underlying client-server architecture allows for *instant deployment* by enabling to perform updates in one place (Gitzel et al., 2007), the server. The ability to instantly deploy new versions is of great ad-

vantage for ISVs since it allows for shorter development cycles and thus faster reactions on changing or new requirements and reduced time-to-market (Khadka, Batlajery, et al., 2014; Fowley, Elango, et al., 2017).

Web Applications impose *low requirements for accessing software* compared to Desktop Applications. They can be used from any system with a Web browser without installation, reducing time and complexity for users. The resulting high *application portability* (Gitzel et al., 2007) fosters a ubiquity of software that allows for new work patterns (Bitkom, 2013). The lower complexity and effort give software vendors the chance to address potentially larger user bases (Forrester Research, 2011).

Social interactions in potentially large user bases have become widespread in Web Applications for end users, allowing the formation of instant online communities (Bressler and Grantham, 2000) resulting in higher levels of user engagement (Bressler and Grantham, 2000) and increased information sharing in work contexts (cf. *Shareconomy* Bitkom, 2013).

Due to these advantages, Web Applications are becoming dominant over Desktop Applications, with new software being built as Web Applications (Politis, 2017) and existing software being replaced by Web Applications (Pettey and Meulen, 2012). For many companies, the Web is the main target of migration activities (Torchiano et al., 2008). Cloud computing/SaaS has seen the most interest from the business perspective (Statista, 2018a). The shift from traditional on-premise software to SaaS-based cloud software is closely related to Web Applications as interfaces and communication are implemented using standard Web protocols, and most SaaS solutions use browsers as clients. The total size of the public cloud SaaS market was already

91.75 billion USD worldwide in 2016 (Statista, 2018c) and is growing (Statista, 2018c; Politis, 2017; Chan, 2018). The number of SaaS applications that organizations use is rising, and many companies will be running purely on SaaS soon (Politis, 2017).

Software vendors react to these developments by no longer building traditional on-premise software. The vast majority of end users consider SaaS applications more helpful than desktop alternatives (Politis, 2017). Existing solutions are becoming not only extended by but increasingly replaced with SaaS solutions (Pettey and Meulen, 2012). Studies show that acceptance of Web Applications is high even in traditional sectors like banking (Pols et al., 2016).

1.2 Motivation

Despite the widespread use and familiarity of Web Applications for end users and their advantages over traditional Desktop Applications as described above, there are still many *non-Web Legacy Systems* and modernization remains an essential topic in the industry (Batlajery et al., 2014; Khadka, Batlajery, et al., 2014; Gartner Research, 2013; Pettey and Meulen, 2013; NASCIO, 2016; Forrester Research, 2011). This section motivates the thesis by arguing that Web Migration is difficult due to the characteristics of Legacy Systems.

Legacy Systems and legacy modernization have been an interest for research for a long time. Several definitions exist in literature. An overview is given by Wagner (2014). In this thesis, we follow the original definition of Brodie and Stonebaker (Brodie and Stonebraker, 1995):

Definition 3: Legacy System (Brodie and Stonebraker, 1995)

Any systems that cannot be modified to adapt to continually changing business requirements and their failure can have a severe impact on business.

Legacy Systems are *business-critical*, i.e., they represent great value for the ISV due to the vast amount of *knowledge* about business processes, rules, etc. (Aversano et al., 2001; Harry M. Sneed et al., 2010b; Wagner, 2014) resulting from high investments (Lucia, Penta, et al., 2009) and generate much revenue (“cash cows” Khadka, Batlajery, et al., 2014). This value can only be preserved through migration into new technological environments (Fuhr et al., 2013). Legacy Systems *resist modification* (Bisbal et al., 1999). Legacy Systems are *poorly documented* or completely lack documentation (Harry M. Sneed et al., 2010b; Warren, 2012; Batlajery et al., 2014; Lucia, Francese, Scanniello, and Tortora, 2008). There is a *lack of experienced workforce* in the ISV which developed the system (Batlajery et al., 2014) due to changes or retirement of the original staff (Lucia, Francese, Scanniello, and Tortora, 2008). Legacy Systems are *based on obsolete technologies* that are incompatible with current/future technological environments and potentially discontinued (Pérez-Castillo, García-Rodríguez de Guzmán, et al., 2013; Batlajery et al., 2014; Heil and Gaedke, 2016). Legacy Systems are *too rigid to comply with new business requirements* (Batlajery et al., 2014). Obsolete technologies (Yli-Huumo et al., 2016) and degraded architecture (Z. Li et al., 2015) of Legacy Systems constitute *Technical Debt* (Cunningham, 1992; Avgeriou et al., 2016), causing high cost and financial risk. Studies over large software repositories determine an average Technical Debt per line of code from 3.61 USD (Sappidi et al., 2011) to 15 USD (Curtis, Sappidi, et al., 2012), meaning that larger systems with more than 300 KLOC accrue between 1 to 4.5 million USD of Technical Debt.

While the above characteristics cause high effort for maintenance and implementation of new requirements, many Legacy Systems still exist in the industry (Fuhr et al., 2013). The main reason for their existence is their business-critical role for the company. Surveys (Khadka, Batlajery, et al., 2014; Batlajery et al., 2014) show that they are perceived as stable, reliable, proven technology, and performance-optimized. Respondents' answers indicate a tendency towards the “never touch a running system” mindset (Batlajery et al., 2014).

Legacy modernization is continuously among the top ten technology priorities for CIOs (Gartner Research, 2013), at position 5 in both industry (Pettey and Meulen, 2013) and public sector (NASCIO, 2016), with the adoption of cloud computing ranked third (Pettey and Meulen, 2013; NASCIO, 2016) and mobile second (Pettey and Meulen, 2013), with high expectations (Forrester Research, 2011).

1.3 Problem Statement

Making a transition from non-Web Legacy Systems to Web-based systems is desirable for ISVs due to the advantages of the Web platform and shortcomings of existing systems detailed above. What is more, this is “widely recognized as a must for keeping competitive in the dynamic business world” (Aversano et al., 2001) to be able to keep pace with changing user expectations and be able to implement new requirements (Fuhr et al., 2013; Lucia, Francese, Scanniello, and Tortora, 2008).

Commencing a Web Migration is hard, however. The above reports on *Software Modernization* indicate that ISVs are still struggling to modernize their Legacy Systems and there are still many non-Web Legacy Systems in existence. This section defines the specific problem of the thesis: the lack of suitable approaches for the initial phase of Web Migra-

tion addressing ISVs concerns about effort and risk. Both the business and technical perspectives make modernization difficult (Khadka, Batlajery, et al., 2014). *Effort* and *Risk* (Heil, Siegert, et al., 2018; Khadka, Batlajery, et al., 2014; Canfora, Cimitile, et al., 2000; Bisbal et al., 1999) are rendering ISVs hesitant to commence a Web Migration.

Effort (cf. ISO/IEEE, 2017b, def. 3.1350). Effort for modernization comprises not only direct development activities, but also managing the modernization project and training users on the new systems (ISO/IEEE, 2006; Harry M. Sneed et al., 2010b; Seacord et al., 2003). Even case studies conducted to demonstrate the efficiency of proposed modernization approaches required one person-year or longer. For instance, Distante, Canfora, et al. (2006) report 12+ PM for a medium-sized 112 KLOC Visual Basic project, Bernhart et al. (2012) 18+ PM for 250 KLOC COBOL, Aversano et al. (2001) 8PM for 130 KLOC COBOL and Maenhaut et al. (2016) report about 14 PM for two projects. Modernization effort is influenced by the complexity of the legacy architecture (Khadka, Batlajery, et al., 2014) and the decomposability (Aversano et al., 2001; Lucia, Francese, Scannielo, Tortora, and Vitiello, 2006; Brodie and Stonebraker, 1995). *Resistance from within the organization* (Harry M. Sneed et al., 2010a) is another factor influencing effort. With the unwillingness of staff to share knowledge and resistance towards change in general and new technologies, in particular, effective communication of the necessity and benefits of modernization is required (Khadka, Batlajery, et al., 2014). According to Gartner (2016), “Cultural issues are at the root of many failed business transformations”.

Risk (cf. ISO/IEEE, 2017b, def. 3.3512). Modernizing Legacy Systems involves high *risk* (Heil, Siegert, et al., 2018; Khadka, Batlajery, et al., 2014; Canfora, Cimitile, et al., 2000; Bisbal et al., 1999; Seacord et al., 2003). Uncertainty of success makes ISVs hesitant to start projects. The

risk of failure in modernization projects is high (Gartner, 2014; Forrester Research, 2011) due to feasibility threats like lack of experienced staff for modernization itself (Harry M. Sneed et al., 2010b; Seacord et al., 2003) and for Web development and due to the complexity of the Legacy System (Khadka, Batlajery, et al., 2014). Modernization poses the risk of *loss of knowledge* (Khadka, Batlajery, et al., 2014). Existing Legacy Systems represent valuable knowledge about business processes, rules, etc. (Aversano et al., 2001; Harry M. Sneed et al., 2010b; Wagner, 2014), often resulting from years of requirements elicitation and experience. However, similar to *tacit knowledge* in organizations which is not expressed explicitly but guides human behavior (Nonaka, 2008), the knowledge in Legacy Systems is not explicitly documented but governs how they operate. Thus, modernization bears the risk of losing this valuable knowledge (Khadka, Batlajery, et al., 2014). *Desirability* of modernization can be uncertain for companies because it is difficult to predict the acceptance of the new system by customers, the utility and usability of a Web-based solution, and the overall Return on Invest (Khadka, Batlajery, et al., 2014).

Similar to Razavian (2013) for SOA Migration, we observed that Web Migration is considered sufficiently addressed in academia, but practitioners still face difficulties in commencing a Web Migration project. Surveys show that the majority of the industry does not use academic resources for modernization (Batlajery et al., 2014).

Many existing migration approaches do not sufficiently address the above problems by supporting ISVs in the initial phase (cf. analysis of phase support in our survey (Heil and Gaedke, 2017)), often assuming that the decision to modernize is already taken.

Problem Statement. The problem addressed by this thesis is the lack of dedicated approaches for the initial phase of Web Migration by taking into account concerns about effort and risk. This leaves ISVs struggling to commence Web Migration.

1.4 Scope

Thus, this thesis is dedicated to the following research questions:

Research Question RQ1

How to support ISVs to commence Web Migration?

This guiding *central research question* (Creswell, 2014) is complemented by the following two detail research questions:

Research Question RQ2

How to support commencing and conducting Web Migration with limited effort?

Research Question RQ3

How to address concerns about risk when commencing and conducting Web Migration?

These research questions were used for ideation using the *How Might We (HMW)* method from the Human-Centred Design (HCD) design kit (IDEO, 2015), as described in section 4.1. Their HMW re-formulation has helped to create a solution consisting of the three core contributions, as outlined in section 1.5. While RQ1 to RQ3 define the scope of this thesis, the following related research questions are considered out of its scope:

- The thesis does not consider purely technical Transformations between programming languages. While there is a large body of work available in the field of Program Transformation, fully automatic Transformation on the code level (cf. family F1 in Razavian, 2013) has shown not sufficient for complex real-world systems when also a change of the basic paradigm is involved, e.g., from procedural to object-oriented (Harry M. Sneed et al., 2010b), which is valid for Web Migration. Instead, Reengineering (ISO/IEEE, 2017b; IEEE Computer Society, 1998) supported by tools and knowledge extracted from the Legacy System is required.
- Database Migration is a related but distinct field of research with a large body of research focusing on migration from Relational to NoSQL databases (Karnitis and Arnicans, 2015; Zhao et al., 2014; Rocha et al., 2015) in particular in the context of Cloud (Strauch et al., 2013), from databases to Extensible Markup Language (XML)³ (W. Li et al., 2014; Tzvetkov and Xiong Wang, 2005) and to Semantic Web (Vavliakis, Grollios, et al., 2013; Vavliakis, Symeonidis, et al., 2011; Xuan Fan et al., 2010). Unlike, e.g. changes in the user interface, these modernizations are not a necessary consequence or pre-requisite of Web Migration. If they have to be undertaken, this should not be done at the same time as Web Migration. This thesis focuses only on modernizations that are required by the change into a Web-based environment.
- This thesis does not address the modernization of business processes (cf. ADM Business Domain modernization@Perez-Castillo2011KDM; and family F3 in Razavian, 2013). While the thesis does address the business perspective of Web Migration by supporting migration decision making and therefore changes in the business model, changes in business processes in order to optimize them or align them for a service-oriented architecture

³<https://www.w3.org/TR/xml11/> Retrieved: 7.12.2019

(Razavian, 2013; Razavian, Nguyen, et al., 2010; Nguyen et al., 2009) are not addressed. Optimization of business processes represents a modernization that is not a consequence or pre-requisite of Web Migration and should, therefore, be conducted separately. A SOA is not the specific target architecture of this thesis, and therefore business process modifications towards SOA are not addressed.

- Technical Debt is part of the motivation of this thesis as it constitutes an important reason why Legacy Systems need to be modernized, but this thesis does not directly address its identification (Z. Li et al., 2015), estimation (Curtis, Sappidi, et al., 2012) and management (Yli-Huumo et al., 2016; Z. Li et al., 2015). The methodology presented in this thesis does indirectly affect Technical Debt in two ways. The knowledge recovery reduces Technical Debt by identification and management of valuable knowledge from the Legacy System, facilitating program understanding for maintenance and reducing the risk of changing the code. The Web Migration per se introduces a change of technology towards open Web standards replacing legacy technology which is part of Technical Debt (Yli-Huumo et al., 2016). These two indirect effects are a consequence of the overlap of Web Migration with modernization. An overview of research directly addressing Technical Debt is given by Z. Li et al. (2015).

Figure 1.1 shows Web Migration, Software Modernization, and their relation to the above out-of-scope topics.

1.5 Contributions

To answer the research questions RQ1 to RQ3, we define an approach for addressing the concerns about effort and risk when commencing a Web Migration project by providing solutions that address the ini-

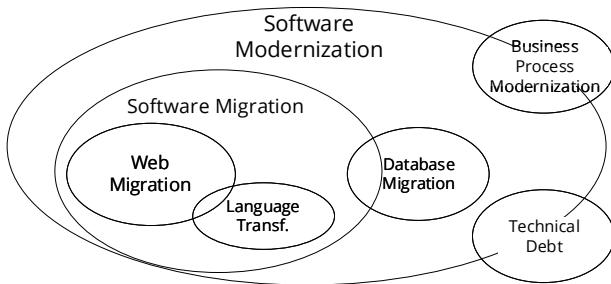


Figure 1.1: Web Migration and Related Topics as Venn Diagram

tial fears and resistance outlined above. This approach is AWSM (Agile Web Migration for SMEs) (Heil and Gaedke, 2016). AWSM provides three core contributions:

- AWSM allows to identify and maintain existing valuable knowledge through crowdsourced Concept Assignment supported by a Web-based annotation platform.
- AWSM minimizes risk through migration pilots and demonstrates desirability and feasibility of a potential Web-based version of the Legacy System applying the Rapid Prototyping paradigm to Web Migration.
- AWSM allows controlling the impact of Web Migration on customers through analysis of visible changes caused by Web Migration in user interfaces.

1.6 Outline

The rest of this thesis is organized as follows: chapter 2 introduces a motivation scenario based on experience from a three-years industrial research project in the context of which the research presented in this thesis was conducted, and based on a systematic field research and problem analysis process, it elicits six requirements representing the

thesis scope, scenario, and stakeholder characteristics. In chapter 3, the state of the art in the field of Web Migration is reviewed with regard to the requirements, and shortcomings of existing approaches are identified. Chapter 4 reports on the solution proposed in this thesis to address the shortcomings. It provides an overview of the methods, principles, and formalisms of the AWSM Methodology and tools of the AWSM Toolsuite. The subsequent three chapters are dedicated to the techniques and tools of the three methods of the AWSM Methodology. Chapter 5 describes the Reverse Engineering Method based on Concept Assignment that allows to recover and manage valuable knowledge in legacy codebases integrated with ongoing development processes and environment and supported by Crowdsourcing. Chapter 6 presents the Risk Management Method based on Rapid Prototyping, which enables to quickly demonstrate feasibility and desirability of a Web-based version of the Legacy System with limited resources and Web Engineering expertise supported by a guided Transformation process. Chapter 7 provides the Customer Impact Method based on Visual User Interface Analysis that facilitates the computation of measurements estimating the perceived empirical similarity between legacy and Web user interface tailored to the characteristics of the target user group to control the customer impact of Web Migration. The overall evaluation is reported in chapter 8. Chapter 9 summarizes the contributions of this thesis and concludes with indications of further research directions.

1.7 Summary

This chapter has introduced Web Systems in general and Web Applications in particular as widely used and accepted software solutions due to their advantages over traditional desktop software. It motivated the thesis by showing that any modernization of existing Legacy Systems is hard due to their specific characteristics. In spite of the notion that

Web Migration is sufficiently addressed in academia, many ISVs are still struggling to commence Web Migration due to concerns about effort and risk, and dedicated approaches addressing these concerns in the initial phases of Web Migration are scarce. From this, the central research question RQ1 of this thesis, “How to support ISVs to commence Web Migration?”, was derived and detailed in RQ2 and RQ3, defining the scope of this thesis. The chapter provided an outline of the main research contributions and the structure of the thesis. The following chapter further explores the defined problem domain through analysis of a motivation scenario and elicitation of requirements for a suitable Web Migration approach addressing the research questions.

Requirements Analysis

2

In this chapter, we further investigate the problem raised in the introduction – concerns about effort and risk that render ISVs hesitant to commence a Web Migration due to insufficient support in initial phases – by outlining the research context in an industrial scenario, analyzing the specific problems within the scenario and through the derivation of concrete requirements.

2.1 Scenario

The following scenario describes the external industrial context in which the research presented in this thesis was conducted. It provides a concrete real-world example of the problems described in section 1.3. The scenario's characteristics with regard to company, software development, and source system provide the basis for the detailed problem analysis in section 2.2 and provide contextual information for the solution elaborated in chapter 4.

2.1.1 Company and Software Development Characteristics

The scenario is based on the situation of *medatixx GmbH & Co. KG* at the beginning of the joint industrial research project “eHealth Research Laboratory” in October 2014. Medatixx is a Small and Medium-sized Enterprise (SME)-sized ISV of about 130 software developers plus 360 additional service and sales employees (*medatixx GmbH & Co. KG*,

2018). Like most SME ISVs (Rose et al., 2016), medatixx is successfully specialized in one particular sector, providing software solutions and IT Services for all kinds of resident doctors' offices and is the largest software provider in this sector in Germany with a market share of about 22% (medatixx GmbH & Co. KG, 2018). The core software development activities focus on information systems for patient management, so-called Patient Management Systems (PMSs). This type of software is subject to various regulatory constraints: general rulings like the GDPR to ensure data privacy (EU General Data Protection Regulation, 2016) as well as regulations specific to the medical sector like required certifications of PMS through the federal association of statutory health insurance physicians (KBV) (Kassenärztliche Bundesvereinigung, 2018). Updated regulations also impose a strict regime of quarterly release cycles that influences developer workload, maintenance activities, and evolution practices of the software products. It is worth noting that the business processes in many doctors' offices are based on the processes represented in the PMS, making it hard to introduce more substantial changes in user interaction patterns or even replace the PMS. As a result of several mergers and existing contracts with customers for maintenance and updates, the company continuously develops and evolves four similar main software products with overlapping features, different technologies, and partially shared codebases. Modernization activities are viewed in an in-house context; outsourcing is not desired. The development staff is experienced in the technology bases and the functionality of the existing software products, but due to time and the mergers, the original developers and expertise are not entirely available anymore. Web Engineering expertise is only limited; Web Migration expertise was not observed. Incremental modernization of older software components is ongoing, focussing on re-development in C#.NET.

The software development activities are organized following agile practices and adoption of agile culture and mindset in work, and decision processes are at an advanced level for both software developers and the management of the development department. The software developers are organized in teams of about 5-7 persons and integrate domain experts as testers. These teams complete work from backlogs managed by product managers in a self-organized way. Except for one team dedicated to corrective maintenance, which adopts a Kanban (Anderson, 2010) process, the teams follow a Scrum-based (Beedle and Schwaber, 2001) iterative development model with minor variations across different teams. Implementations of new features need to be approved by a group of User Interaction (UIX) experts (Team U) to ensure consistently high usability. Further agile methods and technologies such as physical sprint backlogs, time-framed efficient meetings with voluntary participation, continuous integration, and automated software testing as well as knowledge sharing approaches like a gamification-driven wiki are present. The above description characterizes medatixx as a typical instance of the *main stakeholder role* of this thesis: a capable modern SME-sized ISV with the problem of bringing its non-Web Legacy Systems to the Web.

2.1.2 Legacy Codebase Characteristics

For our joint research, we were kindly given access to the source code of one of the PMS called *x.concept* by medatixx. It provides functionality encompassing patient documentation and health records, managing appointments, prescriptions, and billing, focused on Create, Read, Update, Delete (CRUD) operations. The application is a traditional *stand-alone Desktop Application with a Graphical User Interface (GUI)*, distributed via optical media and deployed via local installation on Windows-based personal computers in doctors' offices. For distributed access from several workstations in different rooms, some doctors' offices are using MS

Remote Desktop connections to a central PC that plays a server-like role. To provide an overview of the general characteristics of the codebase, we used the source code analysis tool *cloc*⁴ version 1.80. The source code consists of 34269 files, 30840 of which are unique (redundancy is mainly due to duplicate C/C++ Header files and some amount of code duplication), with an aggregated number of 8.8 million Source Lines of Code (SLOC). Source code analysis identified a heterogeneous landscape of 28 different programming languages and technologies, the vast majority of which (16.6k files, 6.5M SLOC) is Visual C++. Other programming languages include C# (7.2k files, 1.5M SLOC), C (244 files, 163k SLOC), and Pascal (150 files, 100k SLOC), technology-related *software artifacts* include HTML (588 files, 149k SLOC), Windows Resource Files (767 files, 107k SLOC), MSBuild Scripts (437 files, 55k SLOC) and XAML⁵ (122 files, 17.8k SLOC). The main technologies are Visual C++ as programming language, Microsoft Foundation Class Library (MFC)⁶ as GUI framework and FoxPro for persistence. Further details can be found in table A.1. The software is structured in mostly isolated components, in some cases even stand-alone ones with several different communication mechanisms such as COM, IPC/Sockets, and MFC SendMessage. While advanced metrics like functional size measurement (FSM) (ISO/IEC, 2009) would provide more insight into the amount of functionality, automated calculation of function points is still not mature, and due to correlation with lines of code metrics (Albrecht and Gaffney, 1983) their improved objectivity has been questioned. In any case, the size of 8.8M SLOC clearly qualifies x.concept as a large⁷ Legacy System.

⁴<https://github.com/AlDanial/cloc> Retrieved: 6.12.2019

⁵<http://msdn.microsoft.com/en-us/library/ms747122.aspx> Retrieved: 6.12.2019

⁶<http://msdn.microsoft.com/de-de/library/d06h2x6e.aspx> Retrieved: 6.12.2019

⁷only 12% of the applications in the Appmarq repository — representing 1.03 billion Lines of Code (LOC) of 1850 applications — are larger than 1M LOC (Curtis, Muller, et al., 2017)

Within x.concept, ZMS is an isolated stand-alone component, that compiles into a separate executable handling the management of medical appointments. It provides standard calendar functionality like day, 3-day, week and month calendar views, a task list, a vacation planner, a business hours schedule, and management of patient appointments involving resources from medical staff and rooms. Data flows for appointments go directly to an MS SQL Server via ODBC, whereas resource data is loaded through Window-to-Window communication with the main application via MFC SendMessage. ZMS serves as scenario application basis in this thesis. Table 2.1 shows abstract characteristics of the type of legacy software which we focus on in this thesis along with their instantiation in the ZMS scenario application.

Table 2.1: Abstract Technical Characteristics of Legacy Software & Scenario Instantiation (adapted from Heil, Siegert, et al., 2018)

Abstract Characteristics	Instantiation in ZMS scenario application
Legacy language	Visual C++
CRUD functionality	CRUD for medical appointments
Complex Business Logic	find next available time slot
Desktop GUI	Microsoft Foundation Class (MFC)
Component Communication	Window-to-Window via MFC SendMessage
Third-party dependencies	DLLs via assembly loading
File and Data Base Persistence	Files and MS SQL Server via ODBC

2.1.3 Migration Objectives

Current PMS like x.concept primarily support doctors and nurses. Future healthcare applications, however, should support all roles involved in ambulant healthcare, such as patients, pharmacists, or transport providers. The situation is comparable to the banking sector twenty years ago when banking information systems only supported bank clerks as actors. With the increasing ubiquity of Web-based systems, the extension of these formerly closed systems empowering customers as system actors has lead to the successful establishment of online banking as the de-facto standard for internet users (Pols et al., 2016) and enabled the creation of new business models creating the FinTech sector (Schueffel, 2016). Similar effects have been observed for the travel sector. Empowering the physicians' customers, i.e. the patients, in the context of the ZMS scenario application means the following: to provide them with functionality to book their medical appointments based on availability and personal preferences, to re-schedule or cancel appointments, to receive reminders and delay notifications and to access this information independent of the doctors' business hours and integrate the appointments in calendar applications on the patients' personal mobile devices. Comparable functionality is familiar for end users, e.g. in the context of flight bookings. Providing this increased functionality and comfort to patients – the customers of the company's customers – generates an added value and a potential competitive advantage for doctors – the customers of the company – and therefore is desirable for the PMS of the ISV.

To enable this, however, a distributed, Web-based system is required, using open Web standards to provide a high degree of interoperability. The current locally installed, closed PMS cannot deliver the required interactivity, because its installations run on PCs in the doctors' offices – often only during business hours – and cannot be accessed

or interacted with from outside. Integration of further third-party roles like pharmacists and transport providers for federated scenarios would require standardized communication interfaces and authorization mechanisms. The required Web Migration poses a challenge for the ISV, and initiating this process is difficult as outlined in section 1.3 and further analyzed in section 2.2.2.

2.2 Problem Analysis

Effort and risk were identified as the two main problems in chapter 1. This section analyzes the resulting problems for SME-sized ISVs with legacy, non-Web, Desktop Application products, and large existing user base in the context of the scenario situation described above to provide a basis for deriving the solution requirements in section 2.3. Through systematic field research and analysis, it identifies specific causes why ISVs are hesitant to commence a Web Migration and what their main concerns that create organizational resistance are in order to derive solution requirements. To achieve that, the section outlines the research process in section 2.2.1, and presents problem analysis results in section 2.2.2.

The analysis follows the research process shown in fig. 2.1 to systematically derive requirements representing the stakeholder situation. Methods from two methodologies have been employed in this research process in addition to traditional elicitation techniques: Human-Centered Design⁸ and Logical Framework Approach⁹. These are briefly introduced, and the application of methods is described in the following.

HCD is a paradigm for designing solutions to problems with a focus on human needs and desires. In this thesis, it is used as a tool for

⁸cf. <http://www.designkit.org/>

⁹cf. https://ec.europa.eu/europeaid/multimedia/publications/publications/manuals-tools/t101_en.htm

ideation, that provides concrete guidelines and methods for conducting *field research* in a design science context. It has been defined with a technical scope for interactive systems and usability by ISO standard 9241-210 (ISO, 2010). IDEO¹⁰, one of the leading proponents and contributors of HCD has assembled a systematic description of the methodology (IDEO, 2015). The three-phase process comprises activities and artifacts that are used to get a deep understanding of problems from the human needs perspective, generate and select solution ideas, and plan their implementation.

LFA is a project planning and management methodology employed by the European Commission (European Commission, 2004) and other authorities. In this thesis, it is employed as a tool for structuring observations gathered through HCD methods into concrete problem descriptions in this thesis. LFA focuses on systematic *problem and stakeholder analysis*, and *objective setting* in the analysis stage. In particular, *LFA problem trees* allow to identify and represent cause-effect-relationships in the problem domain and are used to analyze the situation of the thesis' main stakeholder, ISVs with legacy software as characterized in section 2.1.1.

2.2.1 Research Process

The research process comprises four phases:

- Field Research
- Consolidation
- Problem Analysis
- Requirements Analysis

The first three phases are briefly described in the following, chapter 2 describes the Requirements Analysis. An overview of the

¹⁰<https://www.ideo.com/> Retrieved: 6.12.2019

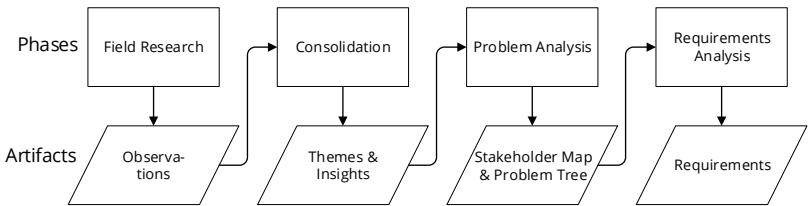


Figure 2.1: Research Process for Requirements Analysis

four-phase research process combining methods from HCD and LFA is presented in Figure 2.1. Table 2.2 provides a mapping of these methods on the phases.

Field research was conducted at the headquarters of the SME-sized ISV medatixx (cf. section 2.1.1) to elicit observations as basis for systematic problem analysis. Methods from HCD’s Inspire phase were employed: *Recruiting Tools* helped identify relevant stakeholders for interviewing, including senior management (“Leiter Softwareproduktion” – Head of Development), middle management (“Abteilungsleiter Softwareproduktion” – Head of Software Development Department) and staff (software engineers, software testers, scrum masters, product owners, DevOps, maintenance) and, following the *Extremes and Mainstream* method including both proponents and opponents of Web Migration. A *Guided Tour* was taken to get to know the different development teams, organizational structures, and work environment. The observations were elicited using individual and group *Interviews* and using in-context *Immersion*. This method was particularly fruitful, comprising a one-week integration in a scrum team, actively participating in development activities and meetings. This provided a good understanding of the daily development practice, expertise level of staff and a solid trust basis for interviews and subsequent feedback cycles,

in particular for the development of the Annotation Platform presented in section 4.4. More than 50 observations of the field research phase were captured and organized using Trello¹¹ cards.

Consolidation was used to abstract from the concrete observations into themes that capture relevant and recurring problem patterns. This was supported by HCD's Ideation phase methods. *Find Themes* in combination with the collaborative filtering proposed in the *Top Five* method was used to cluster the observations and identify the most relevant problem areas, e.g. migration effort, available staff expertise, and unknown potentials of a Web-based solution. *Insight statements* were created for the themes from observations. The themes and insights were captured on post-its to provide the input for problem analysis.

Problem Analysis considered the relationships between the insights and themes to create a systematic view of the problem domain. According to HCD's *Create Frameworks* method, an initial relational map was created. LFA *stakeholder analysis* was conducted to capture the stakeholder map, which systematically describes characteristics and intentions of the involved roles. The problem hierarchy was identified bottom-up from the insights, and themes and their cause-effect relationships were represented as an LFA *problem tree*, identifying three major sub-trees of overloading, doubts about feasibility and doubts about desirability. The results of the problem analysis phase were iteratively improved through a feedback loop with the ISV and are outlined in the next subsection.

¹¹<https://trello.com/> Retrieved: 6.12.2019

Table 2.2: Methods used in Research phases

Phase	Methods
Field Research	HCD Recruiting Tools, Interviews, Extremes and Mainstream, Immersion, Guided Tour
Consolidation	HCD Top Five, Find Themes, Create insight statements
Problem Analysis	HCD Create Frameworks, LFA Stakeholder Analysis, LFA Problem Tree

2.2.2 Results

Problem analysis phase comprises two parts: stakeholder analysis and problem tree analysis. For stakeholder analysis, we identified stakeholders and their characteristics in terms of their interests and how they are affected by a Web Migration, their capacity and motivation, and possibilities to address and engage them. For problem tree analysis, we identified stakeholder problems and their relationships. The stakeholder and problem tree analysis results were presented and validated with the scenario stakeholder, and feedback was incorporated in several feedback cycles.

Stakeholder Analysis. Table B.1 shows the results of stakeholder analysis. It comprises three groups of stakeholders: ISV stakeholders, customer stakeholders, and external stakeholders. The ISV stakeholders are divided into two distinct roles: *Management* and *Software Engineers*. Management is responsible for the decision to migrate and is aware of the risks and thus must be addressed through *risk management* and communication of benefits. Software Engineers are the group of potential actors of Web Migration; they define the requirements for migration activities through their expertise and development pro-

cesses, and tailored methods need to be provided that reduce migration workload. We use the term *Migration Engineer* to refer to a Software Engineer who is performing Web Migration activities. Migration Engineers are to be considered a sub-class of Software Engineers, i.e. all super-class characteristics apply. Two customer stakeholders exist: the *doctor's offices* are the direct customer of the PMS ISV whereas *patients* are the customers of the customers and benefit the most from innovative features and improved usability and interactivity. Furthermore, *competitors* are external stakeholders affected by the effects on the market share of increased competitiveness of the ISV through Web Migration and *companies with similar situation*, i.e. other ISVs with non-WebLegacy Systems and large user bases can benefit from the dissemination of results to apply to their situation.

Problem Tree Analysis. The knowledge about the stakeholders together with themes and insights from the consolidation phase feeds into the analysis of problems and problem relationships. The problem hierarchy is presented in fig. B.1. While it was created bottom-up, starting with insights, we briefly outline the problem tree top-down for easier understanding. The arrows in the LFA tree indicate cause-effect relationships, with effects represented in the direction of the arrows and causes in the opposite direction. The root-level problem is the competitiveness of the ISV company. The effects of this problem, such as losing market shares to the competition, are not shown for brevity. This competitiveness is jeopardized by both the consequences of maintaining a Legacy System and Technical Debt (cf. section 1.1) leading to *overloading* of staff and the *hesitation to commence Web Migration* due to *doubts about feasibility and desirability* (cf. also to resistance from organization in Khadka, Batlajery, et al., 2014; Harry M. Sneed et al., 2010a). The legacy-characteristic causes of overloading, such as low maintainability due to *side-effects, technological depreciation, multi-*

platform environment, and limited time & resources, are pushing factors in favor of a Web Migration. On the other hand, risk and effort of a Web Migration constitute *doubts about feasibility and desirability*. In particular, *limited time and resources* for conducting the migration, *unknown plausibility of a Web-based version*, *difficult integration into ongoing development*, and a *lack of knowledge of methods and technologies* regarding migration and staff with *Web Engineering expertise* feed into feasibility doubts. Likewise, *lack of a deeper understanding of potential benefits*, the *risk of losing poorly documented knowledge*, and *potential impact on existing customers* through abrupt changes form doubts about the desirability of Web Migration. The root causes on the lowest level of the problem tree are *architectural degradation* of the Legacy Systems, *missing documentation and experience of legacy code*, a *strict release cycle regime* due to regulatory and contractual constraints, lack of *staff with Web Engineering expertise* and a large existing *customer base whose workflows are oriented on the user interaction of the software products*. These situational problems impose constraints on potential solutions and are represented as stakeholder requirements in the next section.

2.3 Requirements

In this section, requirements for assessing the state of the art are elicited based on the motivation and problem outlined in the introduction and the scenario-based problem analysis above. These requirements are divided into two groups:

- **Scope requirements** are detailing criteria that ensure that a solution addresses the scope set by the main research question RQ1.
- **Stakeholder requirements** are detailing criteria that ensure the appropriateness of a solution for an ISV as detailed in the scenario based on the problem analysis results in section 2.2.2, representing a more detailed view on effort and risk from research questions RQ2 and RQ3.

Table 2.3 provides an overview of all scope and stakeholder requirements, introduces short IDs for use in the following text, shows their relation to the research questions, and indicates their significance according to RFC 2119 levels MUST¹² and SHOULD¹³ (IETF, 1997). Their relationships with the research questions are shown in fig. 2.2.

In the following, each requirement is detailed, and a mapping onto a three-level assessment scheme (not satisfied, partially satisfied, satisfied) is presented. The mapping is defined based on satisfaction criteria per requirement.

Table 2.3: Scope and Stakeholder Requirements

ID	Requirement	Level
Scope Requirements		
S1 Initial	Initial Phase Support	MUST
S2 Web	Web Application Target	MUST
Stakeholder Requirements		
C1 Risk	Risk Management	SHOULD
C2 Reuse	Reuse of Legacy Assets	SHOULD
C3 Exp	Expertise & Tool Support	SHOULD
C4 Agile	Agile Development Process Integration	SHOULD

¹²for absolute requirements fulfillment of which is mandatory

¹³for requirements fulfillment of which is recommended

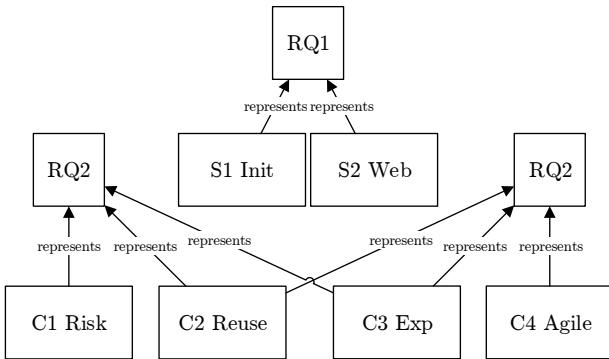


Figure 2.2: Concept Map of Relationships between Research Questions and Requirements

2.3.1 Scope Requirements

The following two requirements detail the scope of this thesis to consider Web Migration approaches that support the initial phase of a Web Migration and which are designed to produce Web Applications from Legacy Systems as of RQ1.

S1 Initial Phase Support

ISVs find it particularly difficult to commence a Web Migration (Heil and Gaedke, 2017; Heil, Siegert, et al., 2018). This requirement, therefore, assesses approaches with regard to how much they support an ISV to overcome the initial resistance. According to the Reference Migration Process (ReMiP) (Harry M. Sneed et al., 2010a), any software migration process can be divided into four phases: Preliminary Study, Conceptualization and Design, Migration and Transition, and Closing down. A similar phase distinction is also found in EU FP7 REMICS (Krasteva et al., 2013): requirements and feasibility, recover, migrate, validation. The first two phases in both reference models comprise

the initial activities before the actual Transformation is started in the third phase, with activities including evaluation of the Legacy System, legacy analysis, and definition of migration strategy and target system. Technically required for any Reengineering or Transformation approach, recovery of knowledge plays a special role in migration and is therefore considered separately in *C2 Reuse*. Thus, approaches covering activities from the first two phases that are not related to recovery are considered to support the initial phase to some extent. However, the mainly technical perspective of the reference models' phases ignores the importance of communication of the necessity and benefits of modernization – communicating the business value is crucial for creating a *business case*¹⁴ for migration (Amazon Web Services Inc., 2018; Khadka, 2016; Menychtas, Konstanteli, et al., 2014; Batlajery et al., 2014; Khadka, Batlajery, et al., 2014) that justifies the project cost with benefits (Harry M Sneed, 1995) – to address migration resistance within organizations (Khadka, Batlajery, et al., 2014; Harry M. Sneed et al., 2010a). A suitable approach should, therefore, additionally provide appropriate methods and tools to supply artifacts that can be used as a basis for communicating the necessity and benefits of migration and supporting the decision making.

The satisfaction criterion of initial phase support is the support for both the technical and the communication perspective of the initial phase, which means that activities prior to the actual migration are addressed and include support for communicating necessity and benefits. The requirement is partially satisfied if activities prior to actual migration are addressed, but communication is ignored. Finally, the requirement is considered not satisfied if only activities from the third and fourth ReMiP/REMICS phase are addressed.

¹⁴“a documented economic feasibility study used to establish validity of the benefits of a selected component lacking sufficient definition and that is used as a basis for the authorization of further project management activities” (ISO/IEEE, 2017b)

S2 Web Application Target

Due to the advantages of the Web platform (Gitzel et al., 2007; Knorr, 2003) and to meet ISVs' migration objectives with regard to the integration of roles and interactivity as described in section 1.1, a Web-based system is required. Software Migration is the process of moving an existing software system from one environment to another (IEEE Computer Society, 2014). Generic software migration approaches, however, do not sufficiently address the Web paradigms, like the asynchronous request-response communication model, client-server separation in the spatial and technological dimension, or URL-based resources and addressable User Interface (UI) states/navigation patterns (Heil and Gaedke, 2017). In the context of Web Migration, the *target environment* is the Web platform. Web Migration approaches comprise different types of target Web Systems (cf. Definition 1) such as Web Applications, SOA-based, and Cloud-based systems (Heil and Gaedke, 2017). Only a subset of these approaches explicitly focuses on Web Applications as defined in Definition 2. Since the migration objectives require a Web Application, including a Web-based user interface, approaches for SOA or Cloud migration are only partially applicable. It is important to note that, according to Definition 2, not every Web System with a Web-based UI is a Web Application since in addition, provision of Web-specific resources is required.

The satisfaction criterion of the Web Application Target requirement is the Web Application nature of the software that is produced by an approach, which means that the resulting software system is based on Web technologies and standards and provides Web-specific resources through a Web-based user interface. The requirement is partially satisfied if the software is a Web System, but lacks the aspect of a Web-based user interface. Finally, the requirement is considered not satisfied if an approach only addresses the development or migration of software in general without consideration of the Web as target environment.

2.3.2 Stakeholder Requirements

The following four requirements detail concerns about commencing a Web Migration from an ISV's perspective by further detailing risk and effort from RQ2 and RQ3.

C1 Risk Management

As outlined section 1.3, any Software Modernization involves a high risk, which makes ISVs hesitant to commence this process (Khadka, Batlajery, et al., 2014; Canfora, Cimitile, et al., 2000; Bisbal et al., 1999; Heil, Siegert, et al., 2018). The risks can be divided into two categories: *risks of migration process failure* and *risks of failure of the resulting new system*. Both categories require appropriate strategies to manage and decrease the risk level. Redevelopment approaches from scratch without consideration of the existing software system and artifacts, called “Cold Turkey” (Brodie and Stonebraker, 1995) and holistic cutover strategies, called “Big Bang” (Bisbal et al., 1999), have a high risk of failure (Harry M. Sneed et al., 2010b; Bisbal et al., 1999). Typical risk management (ISO/IEEE, 2017b) and mitigation strategies for large migration projects are to follow an incremental migration process (Colosimo et al., 2007; Harry M. Sneed et al., 2010b) – originally introduced as package-oriented incremental “chicken little” strategy (Brodie and Stonebraker, 1995) – and to use pilot projects as trial migrations to identify obstacles such as feasibility threats early on (Amazon Web Services Inc., 2018; Harry M. Sneed et al., 2010b). *Feasibility studies* assessing the technical feasibility and economic viability (ISO/IEEE, 2017b) are another means of basic risk management commonly observed. *Portfolio analysis* is a risk management method to identify potential migration candidates using a quadrant graph of business value and technical quality (Seacord et al., 2003; Harry M Sneed, 1995). For these candidates, stakeholders and requirements are identified to finally make the business case, a docu-

ment to support decision making and planning (Amazon Web Services Inc., 2018; Seacord et al., 2003). Advanced risk management methods should thus contribute to the business case (Seacord et al., 2003) in order to address the desirability doubts in section 2.2.2. Modernization bears the risk of losing valuable tacit knowledge about business processes, rules, etc. (Aversano et al., 2001; Distante, Tilley, and Canfora, 2006; Harry M. Sneed et al., 2010b; Wagner, 2014) which are not explicitly documented but only implicitly represented by the legacy source code (Khadka, Batlajery, et al., 2014; Heil and Gaedke, 2017). This knowledge represents high financial value, resulting from high investments in the past (Lucia, Penta, et al., 2009). As shown in section 2.2.2, the risk of losing knowledge is a problem for ISVs. A *risk-managed modernization* strategy (Seacord et al., 2003) therefore, should provide methods to re-discover, sustain, and manage this knowledge to make it useable for subsequent modernization activities.

The satisfaction criterion of the Risk Management requirement is the risk management of the Web Migration, which means that at least one basic risk management practice beyond an incremental process is combined with advanced strategies contributing to the business case and the securing of existing knowledge against loss during migration. The requirement is partially satisfied if the approach only employs basic risk management practices beyond an incremental process model, but lacks contribution to the business case and the securing of knowledge. Finally, the requirement is considered not satisfied if an approach does not take into account risk management.

C2 Reuse of Legacy Assets

Redevelopment from scratch – sometimes also referred to under the category of *replacement* (Almonaies et al., 2010) – is a major strategy for legacy modernization (Wagner, 2014; Khadka, 2016; Harry M. Sneed

et al., 2010b; Almonaies et al., 2010; Bisbal et al., 1999). However, its lack or limited reuse of existing legacy assets incurs significant development cost (Khadka, 2016) and bears the risk of the new system not being as functional as the old one (Almonaies et al., 2010). Maintaining a system's functionality throughout any Software Modernization means to maintain the domain knowledge and business logic (Wagner, 2014). From an end user's perspective, it also means continuity in the user interface and user interaction. As seen in section 2.2.2, companies aim at maintaining the look and feel of the legacy user interface to avoid forcing end users to change their working habits (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, and Tortora, 2008; Distante, Perrone, et al., 2002). Existing *legacy artifacts*¹⁵ of an ISV comprise the legacy source code and the running Legacy System itself, whereas *assets*¹⁶ like documentation, requirements, or models originally used for production are typically missing. (Wagner, 2014; Bisbal et al., 1999; Harry M. Sneed et al., 2010b; Warren, 2012; Batlajery et al., 2014; Lucia, Francese, Scanniello, and Tortora, 2008). Thus, original requirements and models need to be rediscovered from existing software artifacts to enable reuse, employing appropriate Reverse Engineering techniques such as static analysis of legacy source code or dynamic analysis of system behavior (Lucia, Francese, Scanniello, and Tortora, 2008). A suitable approach must promote rediscovery,

¹⁵In this thesis we distinguish between artifacts and *software assets*. “A software artifact is a tangible machine-readable document created during software development. Examples are requirement specification documents, design documents, source code, and executables.” (Object Management Group, 2016a), (cf. *physical asset* ISO/IEEE, 2017b)

¹⁶An asset is an “item, thing or entity that has potential or actual value to an organization” (ISO/IEEE, 2017b). “A software asset is a description of a partial solution … or knowledge … that engineers use to build or modify software products.” (Object Management Group, 2016a) Parts of software can be an artifact and an asset at the same time (e.g. documentation). Unless explicitly codified, we consider requirements, models, rules, etc. represented by the legacy source code as assets but not artifacts, since they are not tangible (cf. *intangible assets* ISO/IEEE, 2017b).

management, and reuse of assets in existing software artifacts to increase the continuity of functionality (Almonaies et al., 2010) and user interaction between legacy and new system in order to reduce the development cost (Khadka, 2016) and accelerate modernization compared to redevelopment from scratch (Harry M. Sneed et al., 2010b).

The satisfaction criterion of the Reuse of Legacy Assets requirement is the degree of reuse of assets from existing software artifacts, which means that legacy source code and the running Legacy System itself should be used as the source for maintaining functionality and user interaction across legacy and new system. The requirement is partially satisfied if either only functionality or only user interaction is maintained. Finally, the requirement is considered not satisfied if existing assets are not taken into account for the continuity of functionality or user interaction.

C3 Expertise & Tool Support

Expertise of the available staff is a crucial resource for any Software Modernization (Khadka, Batlajery, et al., 2014; Batlajery et al., 2014; Harry M. Sneed et al., 2010b; Seacord et al., 2003). While expertise in the source technology base in ISVs with continuous maintenance and development activities due to existing contracts and strict release cycles as described in section 2.1.1 is typically high, experience with older parts of the Legacy System can be significantly lower due to retirement or change of job of the original developers (Khadka, 2016; Batlajery et al., 2014; Khadka, Batlajery, et al., 2014), called *erosion of soft knowledge* (Khadka, Batlajery, et al., 2014). Lack of staff experienced in Software Modernization itself is a feasibility threat (Harry M. Sneed et al., 2010b; Seacord et al., 2003). For Web Migration, additional Web Engineering expertise is required due to the specifics of the target environment (cf. S2 Web), but typically not existing in ISVs of traditional non-Web Desktop Applications (Fowley, Elango, et al., 2017), as shown

in section 2.2.2. For an ISV to be able to commence a Web Migration with existing staff and expertise, approaches need to consider the four experience levels explained above in terms of its requirements and provide supporting infrastructure for partial or full *automation* and proper *guidance* in the migration process. The effort for Reverse Engineering and migration activities can be significantly reduced by suitable support tools (Lucia, Penta, et al., 2009), which only see limited use in the industry (Torchiano et al., 2008). Additional staff or outsourcing¹⁷ the entire migration is not desirable due to limited financial resources and complexity of the Legacy System, respectively (cf. section 2.1.1).

The satisfaction criterion of the Expertise & Tool Support requirement is the expertise requirements, which means that suitable approaches are feasible with existing staff based on experience levels in legacy technology, Legacy System, target technology, and migration, supported, if necessary, through appropriate tools and proper guidance in the migration process. The requirement is partially satisfied if the approach only meets the staff expertise, but lacks tool support. Finally, the requirement is considered not satisfied if additional staff or outsourcing is required.

C4 Agile Development Process Integration

Software providers have *ongoing development and maintenance activities*, as described in section 2.2.2. Organization structures are representing these activities with teams' responsibilities assigned to maintenance of products or components and implementation of new features, following *agile development methodologies* that govern work organization in terms of process, roles and artifacts (Beedle and Schwaber, 2001). The importance of supporting existing ways of working has only recently been acknowledged in lean perspectives on migration research (Razavian

¹⁷cf. also to (Razavian and Lago, 2012) for strategy differences between in-house and outsourced migration projects

and Lago, 2014). Additional activities required by Web Migration thus need to be integrated with day-to-day development activities of existing teams to avoid re-structuring or even exclusively dedicating teams to migration (cf. *modernization teams* in Krasteva et al., 2013; *migration factory teams* in Amazon Web Services Inc., 2018). Of the seven disciplines according to ReMiP (Harry M. Sneed et al., 2010a; Gipp and Winter, 2007), target design, implementation, test, and deployment are very similar to traditional forward software engineering and therefore easier to integrate. In contrast, requirements analysis – for migration, this means eliciting un-documented requirements from legacy artifacts, mainly the legacy code – legacy analysis, and strategy selection are migration specific activities that are significantly different from regular software engineering activities and therefore more difficult to integrate. Integration is not only required at the process level in terms of activities, but also for artifacts that are created and drive the development.

The satisfaction criterion of the Agile Development Process Integration requirement is the completeness of development integration, which means that all migration activities are integrated into ongoing development processes, and integration is also achieved at the artifacts level. The requirement is partially satisfied if not all activities are integrated, or integration of artifacts is lacking. Finally, the requirement is considered not satisfied if an approach does not integrate with ongoing development and is designed to be stand-alone.

2.4 Summary

This chapter introduced a guiding scenario of an SME-sized ISV struggling to bring its non-Web Legacy Systems to the Web and detailed the characteristics of the company, software development, and Legacy System, and described migration objectives. The problem analysis has

detailed the effort and risk factors and introduced a model of stakeholder roles for devising a suitable solution. Based on the problem analysis results, six requirements were elicited. The two scope requirements represent the problem identified as the scope of this thesis in RQ1, i.e., support for the initial phases of Web Migration and Web Applications as target architecture. The four stakeholder requirements addressing RQ2 and RQ3 are derived from the problems of the scenario stakeholder regarding the management of risk, reuse of legacy assets to ensure continuity of functionality and user interaction, suitability for available expertise and tool support, and integration into ongoing agile development and maintenance activities. For each of the requirements, a three-level assessment scheme was introduced, enabling the evaluation of existing Web Migration approaches regarding their suitability in the following chapter.

State of the Art

3

The following analysis of migration methods is focused on dedicated Web Migration approaches, i.e. methods that move Legacy Systems to a Web-based target environment. For that, this chapter

- introduces relevant standards and a migration reference model as basis for the description of Web Migration approaches,
- assesses Web Migration approaches according to the requirements,
- analyzes assessment results from several group and cross-cutting perspectives, and
- identifies shortcomings of existing Web Migration approaches to be addressed by this thesis.

The topic of Web Migration is located in the overlap of two major areas of research: Web Engineering and Software Migration. It requires profound knowledge from both areas to successfully plan and conduct a Web Migration. Knowledge of Web paradigms, architectures, technologies, and development processes is required due to the specific characteristics of the Web as target platform of the migration. Knowledge of Legacy Systems, migration strategies, Reverse Engineering, and migration planning is required due to the specific characteristics of Legacy Systems and different nature of migration activities. Thus, isolated consideration of approaches from either only Web Engineering or Software Migration is not sufficient.

Dedicated *Web Engineering approaches* such as Web Modeling Language (WebML)/ Interaction Flow Modeling Language (IFML) (Object Management Group, 2015), UWE (Koch et al., 2008), AWE (McDonald and Welland, 2005) or OOHDM (Schwabe et al., 1996) fail to address any Web Migration aspects beyond Forward Engineering (in particular requirements S1 and C2). Most Web Engineering approaches start from scratch, not from an existing Legacy System (cf. *brownfield software development* (Hopkins and Jenkins, 2008)) which requires a different set of methods, e.g. requirements elicitation instead of re-discovery of requirements. *Generic Software Migration and Modernization approaches* such as Chicken Little (Brodie and Stonebraker, 1995), Butterfly (Bing Wu et al., 1997), XIRUP(Fuentes-Fernández et al., 2012) or Renaissance (Warren and Ransom, 2002) fail to address the specific characteristics of the Web as target platform (requirement S2).

The selection of approaches assessed in the following analysis is based on their orientation towards the thesis' overarching research question RQ1 and the availability of published research results within Web Engineering and Software Migration communities. Suitability of the assessed approaches for supporting ISVs to commence a Web Migration is evaluated employing the requirements and evaluation scheme introduced in the previous chapter.

3.1 Standards and Reference Model

This section provides a brief introduction to relevant standards and reference models from the fields of Software Modernization and software migration. The standards represent the technological basis of many Web Migration approaches, and the Horseshoe model is the conceptual

standard model of Reengineering approaches. The reference model for software migration provides a common basis for description of the assessed approaches and the solution presented in this thesis.

3.1.1 Architecture-Driven Modernization (ADM)

Extending Model Driven Architecture (MDA) (Object Management Group, 2014) principles to Software Modernization, Architecture-Driven Modernization (ADM) provides a family of related Object Management Group (OMG)¹⁸ standards¹⁹ that form the technological basis of many Web Migration approaches assessed in this chapter. ADM specifies a process of understanding and evolving existing software assets for migration, restructuring, refactoring, translation to other languages, and porting. The ADM Horseshoe²⁰ Model (Pérez-Castillo, De Guzmán, et al., 2011; Pérez-Castillo, Ignacio García-Rodríguez de Guzmán, et al., 2011; Khusidman and Ulrich, 2007) is the standard model of most Reengineering approaches. As shown in fig. 3.1, it consists of three stages: Reverse Engineering, restructuring, Forward Engineering, embedded into a standards-based model-driven methodology. These stages consider modernization in the business and IT domain on the three architectural levels: business architecture, application/data architecture, technical architecture (Khusidman and Ulrich, 2008). Figure 3.2 shows ADM standards used at different levels and stages in the ADM Horseshoe. The following paragraphs briefly summarize the four most relevant standards which have been observed in use in various Web Migration methods in the subsequent state of the art analysis, as indicated in table 3.1.

¹⁸<https://www.omg.org/> Retrieved: 6.12.2019

¹⁹cf. <https://www.omg.org/spec/category/software-modernization>

²⁰adapted from the original SEI horseshoe reengineering model (Kazman, Woods, et al., 1998)

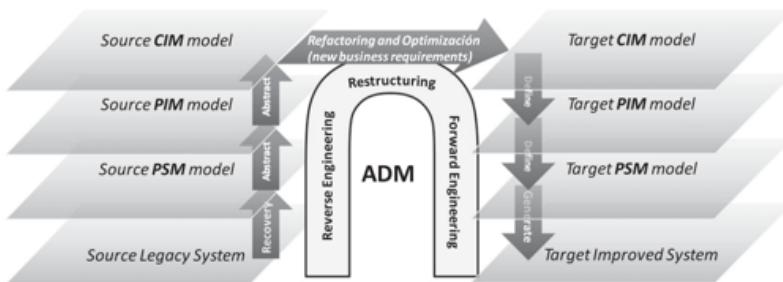


Figure 3.1: Model-Driven Reengineering in the ADM Horseshoe Model
 (Pérez-Castillo, Ignacio García-Rodríguez de Guzmán, et al., 2011)

Knowledge Discovery Metamodel (KDM) (Object Management Group, 2016a) is the central ADM standard, around which the other ADM standards are defined (Pérez-Castillo, De Guzmán, et al., 2011) and was adopted as ISO/IEC 19506:2012 standard²¹. The usage of Knowledge Discovery Meta-Model (KDM) in this thesis is based on version 1.4 of September 2016. KDM specifies a conceptual model for the representation of knowledge in Legacy Systems and its mapping onto XML-based OMG standards such as Meta Object Facility (MOF)²² and XML Metadata Interchange (XMI)²³, allowing integration and interoperability across Software Modernization tools. The KDM *metamodel* is a MOF-compliant Entity-Relationship model, describing physical artifacts of a Legacy System (e.g. source files, executables), code elements (e.g. modules, classes, interfaces), control-flow and data-flow relationships (read/write, call, exceptions), runtime resources (e.g. relational tables, events, processes) and abstractions (e.g. conceptual roles, flows, relationships). KDM provides an ontology of Legacy Systems (Pérez-Castillo, De Guzmán, et al.,

²¹<https://www.iso.org/standard/32625.html> Retrieved: 6.12.2019

²²<https://www.omg.org/spec/MOF/> Retrieved: 6.12.2019

²³<https://www.omg.org/spec/XMI/> Retrieved: 6.12.2019

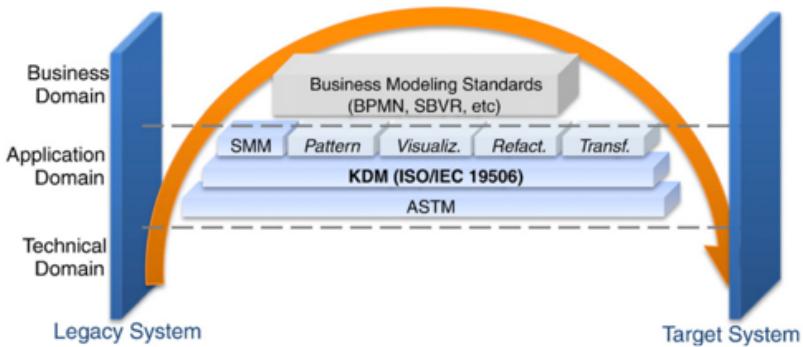


Figure 3.2: ADM standards in the ADM Horseshoe Model (Pérez-Castillo, De Guzmán, et al., 2011)

2011) describing Legacy Systems at a granularity above procedure level. Throughout this thesis, mappings to KDM are provided in order to clarify the semantics of the terms used and facilitate interoperability by providing in brackets the identifier of the KDM concept preceded by the word KDM and a colon. For instance, (KDM: *SourceFile*) is to be understood as a shorthand to refer to the *SourceFile* concept in KDM 1.4 (Object Management Group, 2016a).

Abstract Syntax Tree Metamodel (ASTM) (Object Management Group, 2011) is a complementary ADM standard to KDM. Combined with KDM, it provides a universal software modeling framework, adding the capability of describing Legacy Systems at a granularity below procedure level used by several model-driven Web Migration approaches for low-level software models. To achieve high applicability for any programming language, the syntax is represented using *Abstract Syntax Trees (ASTs)*, a hierarchical representation of all abstract source code constructs. ASTM defines a Generic Abstract Syntax Tree Metamodel (GASTM) using MOF-compliant Unified Modeling Language (UML)

(Object Management Group, 2017), that allows specifying Platform-Independent Models (PIMs) of ASTs. The Platform-Specific Models (PSMs) for various programming languages are supported through the Specific Abstract Syntax Tree Metamodel (SASTM) extensions.

Structured Metrics Metamodel (SMM) (Object Management Group, 2012) specifies a metamodel for representing measurement information related to structured information models, in particular MOF-compliant models. It addresses the need for Web Migration metrics in combination with KDM/ASTM. SMM allows to define measures in terms of basic calculation concepts (counts, mathematic operators, averages) as well as pre-defined metrics (e.g. cyclomatic complexity, LOC), observations including metadata (e.g. provenience, creation time) and measurements (results of the application of measures to observations). Measurements are applicable to any KDM/ASTM element, allowing to represent characteristics of Legacy Systems such as maintainability index (Coleman et al., 1994). SMM's conceptual model of construction of metrics based on computable measures is used in the Web Migration solution presented in this thesis in the area of user interface similarity.

MOF Query/View/Transformation (QVT) (Object Management Group, 2016b) specifies model-to-model Transformations in highly model-driven Web Migration approaches. In the ADM horseshoe, it connects the source and target models as well as models of different levels of abstraction (PSM/PIM/Computation-Independent Model (CIM)) through Transformations. Queries are formal expressions to select parts of a model; views are complex queries which allow selecting complex model subsets; Transformations define mappings from one model to another and make use of queries and views. QVT defines two basic types of Transformation languages: QVT-relational and QVT-operational. QVT-relational are declarative languages and

support bi-directional Transformation, whereas QVT-operational are imperative languages for unidirectional Transformations. For both types, the QVT standard specifies a concrete Transformation language: the Relations language QVTr and the Operational Mappings language QVTo. QVT's conceptual model of queries on the legacy for Transformation is used in the Web Migration solution presented in this thesis for integration with model-driven approaches through queriable external representations of discovered knowledge in the Legacy System.

Table 3.1: Usage of ADM standards in Web Migration methods

ADM	KDM	ASTM	SMM	QVT
REMICS, PRECISO, CloudMIG, L2CMH, MIGRARIA, serviciFi	ARTIST, REMICS, CloudMIG, MIGRARIA	ARTIST, REMICS, MIGRARIA	ARTIST, REMICS, CloudMIG	PRECISO, MIGRARIA

3.1.2 Reference Migration Process

Sneed et al. introduced a reference model for software migration, the Reference Migration Process (ReMiP) (Harry M. Sneed et al., 2010a; Gipp and Winter, 2007) based on a synthesis of existing software evolution and software development methods. Requirements S1 and C4 are specified against its Phases and Disciplines. It serves as generic and adaptable framework for understanding and describing arbitrary Web Migration processes assessed in the following analysis. The Web Migration solution presented in this thesis is described using ReMiP taxonomy and proposed methods are mapped to ReMiP disciplines in order to use the established software migration semantics and facilitate the integration of the proposed methods with other Web Migration approaches. The following paragraph outlines the main aspects of ReMiP.

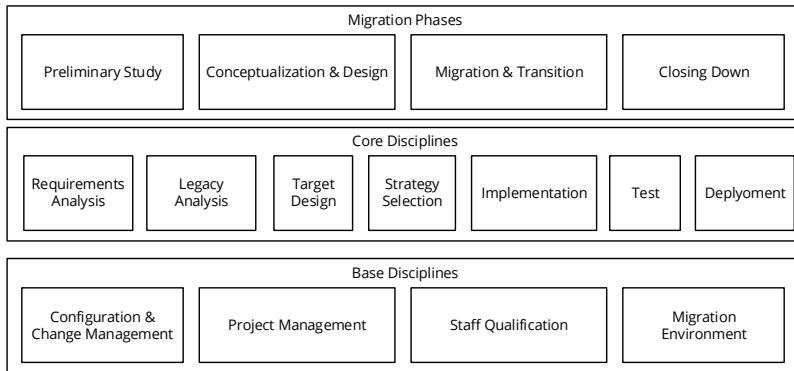


Figure 3.3: ReMiP Overview (adapted from Harry M. Sneed et al., 2010a)

A software migration is considered as a project, and ReMiP's focus is on the management perspective of this project. ReMiP structures relevant activities according to temporal and logical aspects (cf. fig. 3.3). From the temporal perspective, activities are assigned to *migration phases*. From the logical perspective, activities are grouped into migration-specific *core disciplines* and cross-cutting supporting *base disciplines*. ReMiP distinguishes four phases: *Preliminary Study*, which comprises technical, economical and organizational feasibility analysis, *Conceptualization and Design*, addressing migration planning activities such as legacy analysis, strategy selection, and target architecture specification, *Migration and Transition*, conducting iterative package-wise Transformation, testing and delivery, and *Closing down*, comprising activities to archive project documentation and ensure continuous operation of the migrated system. The migration-specific core disciplines are *requirements analysis*, *legacy analysis*, *target design*, *strategy selection*, *implementation*, *test* and *deployment* and represent activities from (forward) software engineering extended for migration (e.g. requirements analysis or target design) as well as activities intrinsic to the migration

domain (e.g. legacy analysis or strategy selection). The core disciplines are supported by base disciplines addressing cross-cutting activities to provide the operational context of the migration project.

3.2 Web Migration Approaches

This section surveys the state of the art of Web Migration approaches. It is organized into four groups:

- **SOA** migration approaches
- **Cloud** migration approaches
- **Web Systems Evolution** approaches
- **Web Application** migration approaches

These groups are defined by the *target environment* (SOA, Cloud, Web Application) and *source system* (Web Systems Evolution) and represent the common classification based on areas of interest in Web Migration research (Kienle and Distante, 2014). We confirmed these groups in our systematic mapping study (Heil and Gaedke, 2017) which serves as the basis for this state of the art analysis. While the scope of this previous survey was wider, comprising 122 primary studies and tools resulting from inclusion/exclusion criteria-based study selection from 870 initial search results from queries and snowballing across six data sources including dedicated methods and tools for specific migration disciplines (cf. ReMiP), the following 22 approaches represent a condensed and updated overview of the state of the art of Web Migration in 2019, focusing on comprehensive approaches with high *migration discipline coverage* (cf. (Heil and Gaedke, 2017)), multi-publication approaches and larger-scale research projects.

The characteristics of individual approaches within the same group, however, differ significantly. In our Survey (Heil and Gaedke, 2017), we

observed that this is partially rooted in the different overall methodologies which they use. These form independent dimensions of description of migration approaches, which are orthogonal to the four groups introduced above. We identified three main methodology groups:

- **Encapsulation** approaches, that wrap the unchanged Legacy System or parts of it and expose a new interface (cf. to *external interface, internal implementation* in *encapsulation* definition ISO/IEEE, 2017b) which is then integrated with the target system,
- **Reengineering** approaches, that employ *reverse engineering* techniques to extract information from the Legacy System and follow up with *forward engineering* to create the target system (cf. also to ISO/IEEE reengineering definition in ISO/IEEE, 2017b; IEEE Computer Society, 1998) and
- **Transformation** approaches, that process the legacy source code by a series of *automatic steps* turning it into the target system's source code.

Table 3.3 provides an overview of the approaches assessed in this section in the two grouping schemes *research areas and target architectures* and *methodologies*.

The following subsections report on corresponding approaches in terms of their name, related publications, research project, overall aim, main focus, method, and evaluation results according to the requirements introduced in section 2.3. The following symbols represent the ratings according to the three-level assessment scheme introduced in 2.3: ○: not satisfied, ◉: partially satisfied, ●: satisfied.

²⁴addresses only phases before migration

²⁵migration planning framework without concrete encapsulation/reengineering/transformation methodology

Table 3.3: Overview of web migration approaches with regard to research areas/target architectures and methodologies

	Encaps.	Reeng.	Transform.	Other
SOA	PRECISO, Marchetto2008, Gaps2Ws	serviciFi	SOAMIG, SAPIENSA	SMART ²⁴
Cloud	AMS, NCHC	IC4	REMICS, L2CMH, ARTIST, CloudMIG	AWS Mi- gration ²⁵
WSE			MIGRARIA, MigraSOA	
Web Application	MELIS, M&S SW, CellEST, DAS		UWA/UWAT+TUIMigrate	

3.2.1 Migration to SOA

Web Services are one of the major target environments of Web Migration. Web Services are often implemented as either Representational State Transfer (REST) (Fielding et al., 2017) Web Service, or as *SOAP*²⁶ service, typically with a Web Service Description Language (WSDL) service interface description and SOAP over HTTP communication (G. Lewis, Morris, Smith, et al., 2008). A significant share of Web Migration research is on migration to Web Services, with the goal to “reengineer the Legacy Systems into a set of services which can be dynamically selected and are distributed across organization boundaries” (Razavian and Lago, 2014). We employ the term *SOA migration* to refer to the

²⁶<https://www.w3.org/TR/soap/> Retrieved: 6.12.2019

entire range of Web Migration towards SOAP and REST Web Services in both the federated SOA application integration and smaller-grained services architectural perspective for maintaining naming consistency with existing research (Khadka, Saeidi, et al., 2013; Razavian and Lago, 2011; Almonaies et al., 2010) that addresses migration to Web Services.

SMART

CMU SEI's SMART method (G. Lewis, Morris, Smith, et al., 2008; G. Lewis, Morris, O'Brien, et al., 2005) initially published in 2005 as Service Migration and Reuse Technique is a decision-making and planning approach for the migration of legacy components to services. SMART provides a process for legacy analysis, service identification and definition of the target SOA environment, a service migration interview guide (SMIG), a tool for automating data collection from SMIG and artifact templates for various analysis results. The iterative SMART process starts with establishing the migration context, including analysis of business and technical context, stakeholder analysis, Legacy System understanding and service identification following a top-down – based on business and mission goals and processes – and bottom-up – based on existing legacy functionality – approach resulting in a business process to service mapping. Great emphasis is put on the migration feasibility decision, taking into account migration potential according to a set of proposed determinations. A subset of candidate services is identified and specified in further detail, information about the Legacy System is gathered in more detail through interviews with technical staff, and the target SOA environment is defined. Gap analysis is used to provide estimates of cost, risk, and effort to migrate the candidate service to the target SOA environment. All information gathered in the previous activities feeds into the definition of the migration strategy that can include a pilot project, migration guidelines, migration path (Encapsulation, Transformation, Reengineering), etc. Based on experience in applying

SMART, SEI realized that many organizations were not ready, did not know enough about SOA to consider migration or had not identified a particular system to migrate, so the scope of SMART was extended along with a redefinition of the acronym to SOA Migration, Adoption, and Reuse Technique (G. Lewis, 2010). In its extended version²⁷, the original SMART method is now one of five variations of the SMART process, which form a family of related techniques addressing different organizational needs. SMART-MP (migration pilot) represents the original perspective, SMART-AF (adoption feasibility) supports organizations' decision making whether to migrate to SOA, SMART-ESP (enterprise service portfolio) supports to select from several Legacy Systems which parts to expose as services, SMART-ENV (SOA environment) supports building the required SOA infrastructure and SMART-SYS (Service-Oriented Systems Development) combines all of the before mentioned variations. SMART has influenced following migration approaches such as REMICS (Benguria et al., 2013; Mohagheghi and Sæther, 2011).

SMART in all variations has a dedicated focus on phases prior to migration but does not include communication of necessity and benefits of the migration, half-satisfying requirement S1. It is designed for migration to SOA-based Web Systems, Web-based user interfaces are not considered achieving half fulfillment for S2. C1 is prominently addressed, including basic techniques like feasibility assessment, incremental activities updating artifacts and incremental process as well as migration pilots. Regarding C2, data about cost, risk, and effort are gathered contributing to the business case and relevant knowledge is captured from the Legacy System. SMART enables reuse of functionality, but not user interaction, yielding half fulfillment for C2. Expertise requirements (C3) for SMART are not high due to the straightforward,

²⁷based on the 2013 SMART materials available at <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508038>

well-defined process and activities and the tools and templates guiding each step. Integration into ongoing development is described neither on process nor artifact level, leaving C4 unaddressed.

Table 3.5: SMART Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	○	●	○	●	○

SAPIENSA

SAPIENSA (Service-enAbling Pre-existing ENterpriSe Assets) (Razavian, Nguyen, et al., 2010; Razavian, 2013; Razavian and Lago, 2012; Razavian and Lago, 2014; Razavian, 2009) is a SOA migration research project²⁸ under the Dutch Joint Academic and Commercial Quality Research and Development (Jacquard) program on Software Engineering. In the context of SAPIENSA, Razavian et al. developed a SOA migration method based on knowledge management focusing on a rational investigation of legacy assets as candidate services, isolation of their properties and Transformation into business services. SAPIENSA results also contributed to EU FP7 project S-CUBE²⁹. The SAPIENSA method aims at exploiting architectural knowledge to enable the Transformation to services on different *SOA abstraction levels* (Razavian and Lago, 2010). Its focus on knowledge management acknowledges the importance of knowledge in migration projects.

Similar to SMART, SAPIENSA employs a *middle-out* strategy, combining top-down domain and business decomposition with bottom-up legacy

²⁸<https://www.nwo.nl/en/research-and-results/research-projects/i/19/4019.html> Retrieved: 6.12.2019

²⁹<https://cordis.europa.eu/project/reference/215483> Retrieved: 6.12.2019

understanding. The SAPIENSA process consists of three phases: architectural knowledge (AK) elicitation, Transformation, and service composition. The AK elicitation extracts and externalizes problem-related (e.g. business processes and rules) and solution-related (e.g. structure, design decisions and rationale) knowledge and identifies candidate services. Transformation is carried out on all levels of abstraction: design elements are reshaped, architecture is restructured, and business models and strategies are altered. The service composition phase focuses on service-based Forward Engineering using the service composition model (SCM) leveraging services obtained from the Legacy System, newly developed services, and external services. Model-driven gap analysis (Nguyen et al., 2009) supports Transformation between as-is and to-be parts of AK such as as-is business process portfolio and to-be business process portfolio, identifying discrepancies and identifying realization strategies. These strategies enable decision making about re-using or re-structuring legacy assets or re-developing them and serve as input to lower-level service Transformation. The SAPIENSA approach has been elaborated in the "lean and mean" migration strategy (Razavian and Lago, 2012; Razavian and Lago, 2014), generalizing core migration activities from common industry practices.

SAPIENSA addresses phases prior to migration but does not contribute to communicating necessity and benefits thus half-satisfying S1. The target system is a service-based Web System; migration of GUI is not considered resulting in half fulfillment of S2. No risk management (C1) is proposed. Reuse of legacy assets is employed to retain functionality, but user interaction is not considered, which half-satisfies C2. Since the SAPIENSA process is not bound to sophisticated Transformation methods or technologies, expertise requirements are not high, but no supporting tools are provided. Therefore the criteria for C3 are half satisfied. Integration into ongoing development is not addressed, yielding

no fulfillment of C4 SAPIENSA contributed to systematizing research on SOA migration by producing the conceptual SOA migration reference framework SOA Migration Framework (SOA-MF) which has contributed to the understanding of migration processes and knowledge in Legacy Systems of this thesis as described in section 4.6.

Table 3.6: SAPIENSA Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	●	○	●	●	○

ServiciFi

The ServiciFi method for SOA Migration presented by Khadka, Reijnders, et al. (2011) and Khadka (2016) is part of the ServiciFi project³⁰ which aims at transforming monolithic software of the financial services domain into services to enable the creation of new SOA-based applications. As typical for SOA migration approaches, it focuses on service identification, extraction, and Transformation. The ServiciFi method itself was created using method engineering to assemble method fragments from three popular service-oriented development methods (SODDM, WSIM, SOMA).

It addresses phases prior to migration and includes analysis of technical feasibility and economic viability but no consideration of communication desirability and necessity. Thus, S1 is half-satisfied. The migration target is a Web system, but lacks consideration of user interface aspects for S2. Portfolio analysis is used to provide a cost-benefit overview and comprises preliminary return on investment in terms of estimated maintenance cost and business value in relation to market needs. The

³⁰<https://servicifi.wordpress.com> Retrieved: 6.12.2019

main part of the process is iterative. C1 is satisfied. While legacy artifacts like documentation, or UML diagrams are analyzed if existing, there is no systematic recovery and reuse of legacy assets (C2) apart from what is technically needed for service extraction. This extraction is carried out in three phases: manual service identification, specification, and construction & testing. ServiciFi employs *concept slicing*, a method that combines *program slicing* with *Hypothesis-Based Concept Assignment (HB-CA)* to extract an executable concept slice (Gold et al., 2005). However, ServiciFi does not provide tool support for this core activity, requiring manual concept slicing even in the small-scale migration case studies presented in its evaluation, thus not addressing C3. The process itself is designed as a stand-alone process with no integration into existing development processes as in C4. It is noteworthy that the situation of Legacy Systems in the financial services sector described as the context for ServiciFi is similar to the situation described in section 2.1. The ServiciFi research project has furthermore produced insightful surveys (Khadka, Batlajery, et al., 2014; Batlajery et al., 2014) on legacy software and Software Modernization from industry practitioners' perspective which have contributed to the motivation and the problem understanding of this thesis as described in chapter 1.

Table 3.7: serviciFi Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	●	●	●	○	○

Marchetto2008³¹

The approach presented by Marchetto and Ricca (2008) aims at migrating legacy Java GUI Desktop Applications based on the Model-View-

³¹this identifier was assigned to the approach due to lack of a name given by the authors

Controller (MVC) pattern³² into Web-service based versions. It focuses on a stepwise migration process that aims at providing concrete guidance to developers and on recommending concrete existing support tools from the Java world for these steps due to the identified lack of such concrete guidance in published approaches. The approach makes use of the UML4SOA UML profile to describe the service-oriented target architecture, consisting of entity, task and utility services described through WSDL (Chinnici et al., 2007) service descriptions.

The six-phase-process includes five phases prior to migration, but these are focused on knowledge recovery, not satisfying S1. Migration to SOA is achieved by replacing method invocations by service invocations in the existing Java desktop GUI. The resulting Web-based system with a non-Web GUI is half-satisfying S2. The process is incremental, successively identifying and migrating more services. Knowledge re-discovery is limited to functionality, that is then codified in use case diagrams. As no basic risk management mechanisms are described, the approach does not satisfy C1. The authors indicate the determination of the underlying business process but without a specified methodology or codification e.g. as Business Process Model and Notation (BPMN) (Object Management Group, 2013) diagram etc. Due to the equality of source and target programming platform, Java, the Web Services are created by wrapping existing source code fragments in the back-end. This is achieved through a semi-automatic wrapper approach, employing the Apache Axis2³³ toolchain to create the WSDL descriptions and client stubs to invoke the services. The GUI is left unchanged and therefore reused in its entirety, thus both functionality, and user interaction are maintained. This wrapping therefore satisfies C2 in both functionality and user interaction reuse. Since the resulting application

³²<http://wiki.c2.com/?ModelViewController> Retrieved: 6.12.2019

³³<http://axis.apache.org/axis2/java/core> Retrieved: 6.12.2019

is still a Java Desktop Application with a Web service backend, the expertise requirements are not very high. Staff with expertise in the legacy environment can conduct the concrete step-wise process with available tool support, yielding fulfillment of C3. C4, integration with ongoing development processes, is not considered.

Table 3.8: Marchetto2008 Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

SOAMIG

SOAMIG (Fuhr et al., 2013; Winter et al., 2011; Zillmann et al., 2011) is a research project³⁴ funded by the German ministry of education and research (BMBF) under the KMU Innovativ programme which aims at the creation of a universally applicable process model for the semi-automatic migration of Legacy Systems into service-oriented architectures based on model-driven techniques and code Transformation. It focuses on model-driven and service identification and Transformation towards Java-based Web services.

The SOAMIG process (Zillmann et al., 2011) has four phases: preparation, conceptualization, migration, and transition. The SOAMIG migration method extends IBM's Service-Oriented Modeling and Architecture (SOMA) method (Arsanjani et al., 2008). It integrates SOMA's Forward Engineering approach with graph-based Reengineering technologies. A TGraph approach is employed to represent and analyze legacy code, sup-

³⁴<http://www.soamig.de/> accessible through Wayback Machine snapshot from March 7, 2018: <https://web.archive.org/web/20180307090643/www.soamig.de>, see also <https://www.offis.de/offis/projekt/soamig.html> Retrieved: 6.12.2019

porting subsequent service identification and implementation. SOMA is applied to specify and design services and TGraph technology-based querying and Transformation techniques are applied to transfer the legacy code into a service implementation. The Legacy System is parsed into TGraph models following TGraph schemas specified in grUML UML profile to describe the structure of the legacy source code, and the models are stored in a graph repository together with manually modeled business processes and the target TGraph schema. Further Transformation is based on two domain-specific languages: GReQL for queries over graph repository and GReTL for Transformation to target TGraph schema. *Dynamic analysis* using AspectJ traces manual execution runs of scenarios by users to identify services for the modeled business processes and is supported by the SOAMIG Business Process Tracer tool. SOAMIG proposes two methods for service implementation: *program slicing* based on the results of static analysis and allocation of classes to business processes based on method invocation frequencies based on results from dynamic analysis. GReQL queries support both methods and help to identify service implementation code in the legacy codebase. Results are verified by a developer and then transformed into Java code and WSDL for the Web services based target implementation using GreTL and GraBaJa for code generation. While early publications indicated application also to non-Java Legacy Systems, COBOL in particular, only Java is fully supported in the Transformation approach two years after project end.

Phases before migration are addressed but only focus on technical details, half-satisfying S1. The target system is a service-oriented Web System, but SOAMIG does not address Web-based user interfaces, half-satisfying S2. Basic risk management methods like technical feasibility study and an iterative process model are present half-satisfying C1, but the business case is not addressed, and extracted knowledge is

technically required for the Transformation. Reuse focuses on functionality; user interaction is not addressed, resulting in half-fulfillment of C2. Expertise requirements are high due to the complex model-driven methods, tools and querying and Transformation languages used. Thus, C3 is not satisfied. The stand-alone process does not address integration, not satisfying C4.

Table 3.9: SOAMIG Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	●	●	●	○	○

Gaps2Ws

The GUI Applications to Web Services (Gaps2Ws) approach (Grechanik, Conroy, et al., 2007) aims at migrating closed, monolithic third-party desktop GUI applications (GAPs) into Web services to enable interoperability. The focus lies on enabling migration to Web services without source code modifications and by non-programmer end users. To achieve this, Gaps2Ws repurposes accessibility technologies and combines them with a visual programming approach. Gaps2Ws uses GUIs as Application Programming Interfaces (APIs), using the accessibility layer of the operating system. Gaps2Ws records the sequence of GUI states and user interactions as a state machine. The visual service designer allows end users to specify Web services by connecting input parameters with GUI elements by drag-and-drop and generates and deploys the Java service implementation.

Gaps2Ws is a GUI-based wrapper solution without explicit process model and belongs to the Migration & Transition phase of ReMiP. Thus phases prior to migration as per S1 are not addressed. The target system

is a service-based Web System without GUI, half-satisfying S2. Risk management and knowledge recovery are not addressed, resulting in not satisfying C1. Reuse is high for legacy functionality due to the Encapsulation approach, but lacks user interaction required for complete fulfillment of C2. Expertise requirements are low since the only required human interaction is simple drag-and-drop service definition supported by a visual programming tool, thus satisfying C3. Integration into ongoing development (C4) is not considered.

Table 3.10: Gaps2Ws Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

PRECISO

The PRECISO SOA migration method (Pérez-Castillo, García-Rodríguez de Guzmán, et al., 2013; Pérez-Castillo, Ignacio García-Rodríguez de Guzmán, et al., 2009) aims at automatic recovery and implementation of Web services from Legacy Systems' relational databases. It focuses on applying an ADM-based horseshoe Reengineering process. Thus, the PRECISO process addresses the three standard ADM stages: Reverse Engineering, Restructuring, Forward Engineering. A platform-specific model is reverse engineered from the legacy database schema information according to an SQL metamodel. This PSM is transformed into a platform-independent model represented as UML2. The PIM is transformed into a Web service PSM, which is then used to generate the implementation code. Model-driven pattern matching is employed to discover services in the database PSM. QVT or manually coded Transformations support the creation of the PIM and derivation of the final PSM. WSDL is used as the metamodel for the Web service PSM and the Web service implementation is based on SOAP.

PRECISO is a SOAP-based database wrapper migration solution without explicit process model, located in the Migration & Transition phase of ReMiP. No phases prior to migration are addressed as required by S1. The target system is a service-based Web System without GUI, half-satisfying S2. Risk management is not addressed; knowledge recovery considers only the data model resulting in no fulfillment of C1. Reuse is low for legacy functionality and user interaction because the Encapsulation approach only addresses the persistence layer, whereas application logic and user interface are not considered. Therefore, C2 is not satisfied. Expertise requirements are low due to the limited scope and high automation achieved in the PRECISO tool, satisfying C3. Integration into ongoing development as in C4 is not considered.

Table 3.11: PRECISO Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	○	●	○

3.2.2 Migration to Cloud

Cloud computing, allowing Web Systems to be built based on scaleable sets of rapidly provisioned, shared resources (Mell and Grance, 2011) is another major step in the Web's evolution (Kienle and Distante, 2014), that has resonated in Web Migration research (Heil and Gaedke, 2017). In particular, the SaaS paradigm has influenced the architecture of Web Applications and thus Web Migration target architectures and methods. However, even more than SOA, cloud computing is an independent principle applying to software systems in general (Kienle and Distante, 2014). Additional requirements due to the cloud environment such as scalability (Jamshidi, Ahmad, et al., 2013), multi-tenancy with regard

to security (Menychtas, Konstanteli, et al., 2014), cloud resource and provider selection (Frey and Hasselbring, 2011b) and IT operations (Amazon Web Services Inc., 2018) are addressed in cloud migration approaches. In the following, we consider cloud migration approaches towards Web Systems – mainly SaaS applications – according to requirement S2 Web. A more detailed overview of cloud migration research can be found in dedicated surveys (Jamshidi, Ahmad, et al., 2013; Pahl et al., 2013; Fahmideh et al., 2018)

AWS Migration

Amazon provides a set of resources³⁵ to support companies to migrate to the cloud, specifically to Amazon Web Services (AWS). The proposed cloud migration approach is called AWS Migration (Amazon Web Services Inc., 2018; Amazon Web Services Inc., 2017). It aims at enabling cloud migration for companies through a set of methods and best practices that are combined into an iterative approach. AWS Migration focuses on the business and planning perspective of cloud migration. The Amazon Cloud Adoption Framework (CAF) (Amazon Web Services Inc., 2017) is part of AWS Migration and gives recommendations how to create an actionable plan for cloud adoption in companies by assessing readiness in terms of gaps in skills and processes in business and technical perspectives: business, people, governance and platform, security, operations.

AWS Migration acknowledges the importance of organizational culture to motivate change and proposes the AWS Organizational Change Management(AWS OCM) framework. Six different cloud migration strategies, called the 6 R's, are recommended, and selection criteria explained: re-host, re-platform, re-factor/re-architect, re-purchase, retire,

³⁵<https://aws.amazon.com/cloud-migration/> Retrieved: 6.12.2019

retain. Guidelines to build a business case for migration that serves as the data-driven rationale for initiating migration are provided, including cost and business value. A migration readiness assessment (MRA) process produces gaps and actions address these gaps in the six CAF areas. Discovery of existing assets in the on-premise environment is supported by discovery tools³⁶ and feeds into application portfolio analysis, but is very coarse-grain, addressing only the system level like servers, databases, OS versions, etc. and not knowledge in systems. Migration planning is recommended to employ traditional project management methods. Security, Operations and Platform perspectives are briefly explained, referencing the non-migration-specific AWS whitepapers on these topics. Prior to execution, smaller-scale migration pilots are recommended to test processes and gain experience. The migration execution follows a cyclic six-phase process of discover, design, build, integrate, validate, cutover.

AWS migration addresses phases prior to migration in full detail, including communication aspects and satisfying S1. While it can be used to migrate to SaaS, the AWS migration method addresses more general cloud migration and does therefore not satisfy the target environment Web requirement S2. Risk management is considered in terms of feasibility assessments, portfolio analysis, migration pilots, iterative process model and building the business case. However, knowledge discovery is too coarse-grain due to the generic nature of AWS migration, thus only half-satisfying C1. C2 is not satisfied as no specific reuse of functionality or user interaction is addressed. Expertise requirements are high as AWS Migration proposes to first create a CCoE (Cloud Center of Excellence) with staff experienced in migration and target environment. The process

³⁶<https://aws.amazon.com/application-discovery/> Retrieved: 6.12.2019

further is based on dedicated cloud teams forming a migration factory³⁷ within the company, targeting large scale companies with appropriate Human Resources and support through external partners from the AWS Migration Acceleration Program³⁸ (AWS MAP) and AWS Professional Services³⁹. This inhibits fulfillment of C3. Tool support, on the other hand, is extensive and bundled in the AWS Migration Hub⁴⁰. Regarding C4, integration into ongoing development is not addressed.

Table 3.12: AWS Migration Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	○	●	○	○	○

REMICS

REMICS (Krasteva et al., 2013; Wendland et al., 2013; Barbier, Henry, et al., 2013; Abhervé et al., 2013; Barbier, Deltombe, et al., 2013; Mohagheghi and Sæther, 2011; Sadovskykh et al., 2011; Mohagheghi, Berre, et al., 2010) is an EU FP7 project⁴¹ that aims at providing a tool-supported model-driven methodology for migrating legacy applications to interoperable service cloud platforms. Its focus is on the development of model-driven methods and tools for the migration of Legacy Systems to loosely coupled systems using an ADM-based bottom-up approach that combines recovery of the Legacy System architecture with restructuring towards SOA and deployment and verification in the cloud environment.

³⁷cf. reengineering factory (Borchers, 1996) idea that migration can be factory-like product line through standardized processes and organization

³⁸<https://aws.amazon.com/migration-acceleration-program/> Retrieved: 6.12.2019

³⁹<https://aws.amazon.com/professional-services/> Retrieved: 6.12.2019

⁴⁰<https://aws.amazon.com/de/migration-hub/> Retrieved: 6.12.2019

⁴¹<http://www.remics.eu> Retrieved: 11.12.2018

The REMICS modernization method adheres to the ADM horseshoe model (Pérez-Castillo, De Guzmán, et al., 2011; Pérez-Castillo, Ignacio García-Rodríguez de Guzmán, et al., 2011; Khusidman and Ulrich, 2007): architectural recovery feeds into Transformation which is followed by Forward Engineering. The recover activity identifies business processes, business rules, components, implementations and test specifications in UML and requirements in RSL (Abhervé et al., 2013). Semi-automatic model-to-model Transformations create PIM4Cloud profile SoaML deployment models which are used in forward MDA (Object Management Group, 2014) engineering to generate the code of the service cloud implementation using model-to-code Transformations. Model-driven test derivation through manual requirements recovery in meetings with legacy developers, refactoring of RSL requirements and test generation using UML state machines (Wendland et al., 2013). REMICS employs and provides a high number of model-driven technologies and tools, among them OMG Standards ADM, KDM and ASTM with extensions like RSL supported by ReDSeeDS and SoaML, UML with several profiles and BPMN supported by the Modelio modeling tool, BLU AGE for architectural recovery, Eclipse-based Focus!MBT as test modeling environment. REMICS agile extension (Krasteva et al., 2013) maps the REMICS onto a Scrum-oriented methodology: progressing in modernization sprints by modernization teams and employing meetings and artifacts from Scrum, it employs an iterative, incremental and agile model.

REMICS addresses phases prior to migration, but the communication of migration necessity and benefits for complete fulfillment of S1 is not addressed. The target architecture is an SaaS system following the Service Cloud Paradigm, a combination of cloud computing and SOA, and includes a Web-based GUI which satisfies S2. Basic risk management is addressed in terms of feasibility evaluations and the iterative process of the REMICS agile extension. Knowledge recovery into models is

present, but the business case is not addressed, thus half-satisfying C1. Reuse of functionality is addressed in detail through the SOA migration parts of REMICS, but maintaining user interaction is not part of REMICS which leads to half-fulfillment of C2. Expertise requirements are high due to the plethora of model-driven techniques and tools employed, therefore not satisfying C3. REMICS has been mapped into an agile process, and even though not explicitly described, integration on both process and artifact level would be possible due to high overlap to Scrum process and artifacts, leading to a half rating for C4.

Table 3.13: REMICS Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	○	○	●

ARTIST

ARTIST (Advanced software-based seRvice provisioning and migration of legacy Software) (Brunelière, Cabot, Izquierdo, et al., 2015; Menyctas, Konstanteli, et al., 2014; Brunelière, Cabot, Dupé, et al., 2014; Menyctas, Santzaridou, et al., 2013; Orue-Echeverria et al., 2014; Konstanteli et al., 2015; Brunelière, Cánovas, et al., 2013) is an EU FP7 research project⁴² that aims at providing a comprehensive, tailorabile end-to-end cloud migration methodology addressing both business and technical aspects. ARTIST focuses on the Transformation and modernization of legacy software assets and businesses to facilitate automated evolution towards cloud-based SaaS.

⁴²<http://www.artist-project.eu/> accesible through Wayback Machine snapshot from May 4, 2018: <https://web.archive.org/web/20180504233035/http://www.artist-project.eu/> Retrieved: 6.12.2019

The ARTIST migration methodology defines roles, activities, artifacts and a process model consisting of four phases: pre-migration, migration, post-migration and migration artifacts reuse & evolution. Pre-migration addresses technical and economic feasibility assessment, requirement analysis and migration decision making. The model-driven technical migration phase comprises UML-based Reverse Engineering using MoDisco⁴³, modernization based on model-to-model Transformations, model annotation and partial code generation, business and process-related modernization considering legal aspects (SLAs) and business model updates. ARTIST's post-migration phase proposes test-based verification, validation of migration goals, availability SLA checking, and cloud provider certification. An additional focus on the life cycle of the migrated application is represented in the migration artifacts reuse & evolution phases which addresses model reuse through a repository or Web-based public marketplace as well as maintenance and evolution management aspects.

ARTIST thoroughly considers phases prior to migration, including migration decision making and benefits, completely satisfying S1. The target architecture is a cloud-based SaaS system; a Web-based GUI is not addressed, thus only half-satisfying S2. C1 is satisfied as risk management is part of the methodology in terms of technical and economic feasibility assessment, business goals conformance checking and SLA compliance verification, and extensive model-driven knowledge recovery is also specified. The model-driven Transformation process addresses reuse of functionality, which half-satisfies C2, but maintaining user interaction is not part of ARTIST. Expertise requirements are high due to the wide

⁴³part of the preceding European FP6 MODELPLEX project, cf. <https://www.eclipse.org/MoDisco/> Retrieved: 6.12.2019

range of model-driven Reverse Engineering and Transformation techniques and tools employed, resulting in no fulfillment of C3. C4 is not satisfied as ARTIST is specified as a stand-alone migration.

Table 3.14: ARTIST Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	○	●	○	○	○

CloudMIG

The CloudMIG approach (Frey and Hasselbring, 2010; Frey and Hasselbring, 2011b; Frey and Hasselbring, 2011a; Frey, Hasselbring, and Schnoor, 2012) aims at assisting reengineers in performing semi-automatic migration of enterprise software systems to scalable and resource-efficient cloud-based Platform as a Service (PaaS)/Infrastructure as a Service (IaaS) environments. It focuses on the SaaS provider perspective, balancing resource-efficiency and scalability.

The CloudMIG method reverse engineers the Legacy System into an architectural representation using OMG's KDM and a utilization model through log analysis and the Kieker⁴⁴ software monitoring tool. CloudMIG's architectural models of the Legacy System and the cloud environment models are complying to UML metamodels that were aligned with KDM through the piggyback pattern of Domain-specific Language (DSL) realization. The utilization model represents resource usage and performance characteristics using OMG's Software Metrics Meta-model (SMM)⁴⁵. Target cloud environment selection is enabled through modeling available cloud provider IaaS and PaaS offerings. A subse-

⁴⁴<http://kieker-monitoring.net/> Retrieved: 6.12.2019

⁴⁵<https://www.omg.org/spec/SMM/> Retrieved: 6.12.2019

quent generation step transforms these models into the target architecture, a mapping model and a list of constraint violations. Rule-based heuristics allow feature allocation onto the available cloud resources. Manual adaptions prepare the models for the final model-to-code generation step. CloudMIG Experiments highlight the shortcomings of simple cloud migration approaches for enterprise software, demonstrating that mere deployment in IaaS does not solve over- and under-provisioning of resources. CloudMIG also models cloud environment constraints, automatically detects violations and supports to assess technical suitability before and alignment after migration, defining a cloud suitability and alignment hierarchy. The CloudMIG method is supported by the CloudMIG Xpress tool⁴⁶.

CloudMIG is a cloud migration method that does not address phases prior to migration as in S1 apart from knowledge recovery. The target system is a cloud-based SaaS application and with a Web-based UI, satisfying S2. Risk management (C1) is not considered. Reuse is high for legacy functionality, but user interaction is not regarded resulting in half-fulfillment of C2. Expertise requirements are high due to the complex MDRE and Transformation techniques and therefore do not satisfy C3. Likewise, integration into ongoing development for C4 is not considered.

Table 3.15: CloudMIG Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	○	○

⁴⁶<https://sourceforge.net/projects/cloudmigxpress/> Retrieved: 6.12.2019

IC4

The IC4 (Irish Centre for Cloud Computing and Commerce) approach (Fowley, Elango, et al., 2018; Fowley, Elango, et al., 2017; Jamshidi, Pahl, et al., 2015; Fowley and Pahl, 2018) aims at reengineering Legacy Systems to cloud-native architectures. It focuses on the migration planning from the perspective of SME-sized ISVs with limited cloud expertise that want to transform to SaaS providers by leveraging PaaS cloud resources with a focus on experimentation-oriented feasibility studies on architecture and cost concerns to drive allocation of software components to cloud resources.

IC4 proposes an incremental and pattern-based migration process with early experimentation and performance measurements. The IC4 process consists of four stages: Consultation with ISV CEO, ISV PaaS Infrastructure Assessment and Requirements, ISV Developer and Software Development and ISV Provisioning. Architectural Reengineering from on-premise via virtualization and PaaS to cloud-native is supported by a migration pattern catalog (Jamshidi, Pahl, et al., 2015) comprising isolated architectural Transformations. Experimentation with prototypes is proposed to address technical feasibility, performance benchmarking and compare different Reengineering options and related licensing models. IC4 links different technical cloud-based application architectures with business models in terms of costs, pricing and licensing. It uses scenario-based risk assessment methods in combination with efficiency, complexity and security metrics.

IC4 is a cloud migration planning method that addresses phases prior to migration. However, the communication of necessity and benefits is not described, so S1 can only be considered half-satisfied. Targeting PaaS-supported SaaS applications with Web-based UI, IC4 satisfies S2. Basic risk management is considered in the planning through

experimentation for analysis of technical feasibility and financial viability analysis through cost model comparisons. The partial Prototyping during experimentation also contributes to the business case, providing concrete benchmarking results, however, systematic recovery and management of knowledge are not addressed. Therefore, C1 is only half-satisfied. Due to the coarse-grain backend component focus, reuse is high for legacy functionality, but user interaction is not regarded as required for complete fulfillment of C2. Expertise requirements are high due to the required manual Reengineering and lack of tool support leading to non-fulfillment of C3. Integration (C4) into ongoing development is not considered.

Table 3.16: IC4 Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	○	○	○

AMS

AMS (Application Migration Solution) (Meng et al., 2011) proposes migration of legacy desktop GUI applications to the cloud through GUI recognition and reconstruction technology . It focuses on the migration of Legacy Systems without source code. This is realized through technology-specific GUI recognition based on memory and handle analysis for programs based on the Microsoft Component Object Model (COM)⁴⁷ platform. For each window in the legacy GUI, an HTML template is generated using the NVelocity Template Engine. The AMS Server acts as a mediator between the legacy and the Web UI, translating user interaction events to synchronize UI states. AMS achieves

⁴⁷<https://docs.microsoft.com/en-us/windows/desktop/com/component-object-model--com--portal> Retrieved: 6.12.2019

application-specific desktop virtualization in the browser for multiple users using Sandboxie⁴⁸ as a virtualization environment to run several instances of the legacy application simultaneously.

AMS is a GUI-based Encapsulation solution without explicit process model. It belongs to the Migration & Transition phase of ReMiP. Thus phases prior to migration (S1) are not addressed. The target system is a Web System with a Web-based UI mirroring the legacy UI, but not a Web Application, which half-satisfies S2. Risk management and knowledge recovery required by C1 are not addressed. Reuse is high for legacy functionality and user interaction due to the Encapsulation approach creating and synchronizing a one-to-one Web representation of the legacy Desktop Application, satisfying C2. Expertise requirements are low since the software focus of AMS does not require manual action, making it suitable for C3. Integration into ongoing development as per C4 is not considered.

Table 3.17: AMS Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

NCHC

Wang and Chang (2014) from Taiwan's National Center for High Performance Computing (NCHC)⁴⁹ propose a cloud migration approach for desktop applications based on Apache Guacamole⁵⁰ technology. It

⁴⁸<https://www.sandboxie.com> Retrieved: 6.12.2019

⁴⁹<http://www.nchc.org.tw/> Retrieved: 6.12.2019

⁵⁰<http://guacamole.apache.org/> Retrieved: 6.12.2019

aims at providing cloud-based desktop virtualization through Web technologies. The main focus is to enhance legacy Desktop Applications with cloud platform advantages such as collaboration, resilience, and scalability without the need for specific remote desktop clients. Instead, a platform-independent, clientless HTML5-based remote desktop virtualization running in the browser is proposed. Sessions are handled by a terminal proxy serving as a gateway to the virtualized terminal servers. This virtualization manager controls the KVM-based hypervisor through the libvirt⁵¹ API and handles sessions and authentication. Guacamole is used as a clientless remote desktop gateway, translating from protocols like Microsoft RDP, VNC, and SSH to the Guacamole protocol, which communicates via WebSockets to the JavaScript Guacamole client that implements the HTML5-rendering in the browser.

The NCHC cloud migration is a software virtualization solution without explicit process model, belonging to the ReMiP Migration & Transition phase. Thus phases prior to migration (S1) are not addressed. The target system is a Web System including a Web-based UI, however, this UI is an identical copy of the OS desktop running the Desktop Application's UI and therefore does not support Web functionality such as bookmarking, half-satisfying S2. C1 is not fulfilled due to lack of risk management and knowledge recovery. Reuse is high both for legacy functionality and user interaction due to the Encapsulation approach, which satisfies C2. Expertise requirements are low due to the tool-centric approach satisfying C3. C4, integration into ongoing development, is not considered.

⁵¹<https://libvirt.org/> Retrieved: 6.12.2019

Table 3.18: NCHC Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

L2CMH

The Legacy-to-Cloud Migration Horseshoe (L2CMH) framework (Ahmad and Babar, 2014) aims at migrating Legacy Systems to cloud-based architectures through software Reengineering. The focus lies on architecture-driven migration, abstracting source code level details to preserve properties of Legacy Systems during cloud migration, and migration planning. Similar to REMICS, L2CMH follows a Reengineering horseshoe model inspired by OMG's ADM (Pérez-Castillo, De Guzmán, et al., 2011; Pérez-Castillo, Ignacio García-Rodríguez de Guzmán, et al., 2011; Khusidman and Ulrich, 2007) applied to cloud migration. It precedes the three traditional recover, transform and develop phases with a plan phase. The migration planning comprises feasibility (effort/cost estimation) and requirements analysis, cloud provider and migration strategy selection and produces a migration plan. L2CMH performs architecture recovery through static code analysis and graph-based architecture model abstraction. Transformation is driven by graph Transformations applying atomic change operators. For Forward Engineering the transformed cloud architecture is defined in SCA⁵² and implementation code is generated.

The plan phase of L2CMH cloud migration addresses aspects prior to migration but does not consider communicating the necessity and benefits of migration, which only half-satisfies S1. The target system is a

⁵²<http://oasis-opencsa.org/sca> Retrieved: 6.12.2019

cloud-based Web System, however, half-satisfying S2, a Web-based UI is not described. Basic risk management is addressed through a feasibility study, but the migration business case is not considered beyond effort/-cost lacking complete fulfillment of, C1. The Transformation approach has high reuse for legacy functionality, but user interaction is not considered, resulting in half-fulfillment of C2. Expertise requirements are high due to the graph-based modeling and Transformation techniques required and lack of concrete tool support. This means that C3 is not satisfied. Integration into ongoing development (C4) is not considered.

Table 3.19: L2CMH Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	●	●	●	○	○

3.2.3 Web Systems Evolution

Within Web Migration research, Web Systems Evolution (WSE) takes a special role because not only the target system but also the source system is a Web System. Approaches from this group address changes of existing Web-based Legacy Systems towards other types of Web Systems, e.g. from MVC to SOA, from code-based to model-driven or from static HTML to AJAX. They are Software Modernization approaches that emphasize the perfective perspective more than the adaptive perspective (ISO/IEEE, 2006) in the dimensions of architecture, design, and technology evolution (Kienle and Distante, 2014). While the majority of WSE approaches focuses on technological changes and therefore has a low migration discipline coverage, the following WSE approaches are from comprehensive research projects and are comparable to other ap-

proaches from the SOA and Cloud groups. For more details about Web Systems Evolution research refer to (Kienle and Distante, 2014) and the International Symposium on Web Systems Evolution proceedings series.

MIGRARIA

MIGRARIA (Sosa-Sánchez, Clemente, Sanchez-Cabrera, et al., 2014; Sosa et al., 2013; Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Rodríguez-Echeverría, Conejero, Linaje, et al., 2010) is a project⁵³ that aims at modernization of legacy Web Applications to new Web paradigms: Rich Internet Applications (RIAs) and Service-oriented architecture. The early works on RIA (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Rodríguez-Echeverría, Conejero, Linaje, et al., 2010) described in this section propose the MIGRARIA method, the later works on SOA propose a different method called MigraSOA and are assessed separately in the following section MigraSOA (Sosa-Sánchez, Clemente, Sanchez-Cabrera, et al., 2014; Sosa et al., 2013).

The MIGRARIA method aims at the modernization of legacy Web Applications to Rich Internet Applications (RIAs). Rodríguez-Echeverría, Conejero, Linaje, et al. (2010) present an aspect-oriented Transformation approach from WebML⁵⁴ to WebML with RIA extensions (Bozzon et al., 2006; Manolescu et al., 2005; Carughi et al., 2009). Later work focuses on the Transformation of Java J2EE based Web Applications to RIAs-based on ADM principles (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012). While their early work (Rodríguez-Echeverría, Conejero, Linaje, et al., 2010) mainly considers client-side storing and

⁵³<http://www.eweb.unex.es/eweb/migraria/> Retrieved: 6.12.2019

⁵⁴the Web Modeling Language, cf. <http://webml.deib.polimi.it/> Retrieved: 6.12.2019

processing and asynchronous communication, later work (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012) formally introduces a more comprehensive list of RIA features.

S1 is not satisfied as phases before migration beyond knowledge recovery are not addressed. Both legacy and target system are full Web Applications including Web-based GUIs and thus satisfying S2. Risk management (C1) is not addressed. As Transformation approach, the degree of reuse is high; functionality is maintained regardless of whether moved to the client side or kept on the server side. User interaction is considered as maintained, even though switching from a traditional Web Application to the single page paradigm has an impact. However, this change is arguably less significant than for non-Web to Web Migrations and regeneration of the “look&feel” of the legacy Web Application is stated as a requirement of the MIGRARIA method. Thus MIGRARIA satisfies C2. The expertise requirements in particular for the migration itself are high due to the employed technologies like KDM for describing the Legacy System and the required Model-to-Model (M2M) Transformations like ATLAS Transformation Language (ATL)⁵⁵ or QVT, inhibiting fulfillment of C3. Integration into ongoing development processes as in C4 is not addressed.

Table 3.20: MIGRARIA Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	○	○

⁵⁵cf. <https://www.eclipse.org/atl/> Retrieved: 6.12.2019

MigraSOA

The MIGRARIA project aims at the modernization of legacy Web Applications to new Web paradigms. Under the name MigraSOA, this section describes the more recent works the project (Sosa-Sanchez, Clemente, Prieto, et al., 2017; Sosa-Sanchez, Clemente, Sanchez-Cabrera, et al., 2014; Sosa et al., 2013) which focus on SOA as target Web paradigm. Like the MIGRARIA method, MigraSOA belongs to the Web Systems Evolution group and not to SOA Migration, because the source system is already a Web System: it proposes a model-driven Transformation of MVC Web Applications to SOA. Java Struts is the concrete source MVC technology. The focus is on model-driven service identification and service code generation (Sosa et al., 2013) and alignment of services with BPMN Models (Sosa-Sanchez, Clemente, Sanchez-Cabrera, et al., 2014). MigraSOA uses the Reverse Engineering step of the MIGRARIA method to extract an instance of the MIGRARIA MVC metamodel consisting of data, operations, controlflows with Modisco, services are manually identified in this model, using model-to-model Transformations based on ATL rules a Simple-SoaML⁵⁶ is derived, and model-to-text Transformations using Acceleo⁵⁷ are used to generate the WSDL, server skeleton and wrapper invocation code to be weaved into the legacy Web Application using AspectJ. For the alignment of business processes with the service layer derived from the legacy Web Application (Sosa-Sanchez, Clemente, Sanchez-Cabrera, et al., 2014), businesses processes are manually modeled using BPMN 2.0. Additionally, a semantic dictionary describing the domain is created by experts, containing terms and related terms (synonyms, hyponyms, meronyms). For each task in the BPMN, a Wang-Ali similarity matrix based selection of services is then performed, comparing the synonym sets of task and service. BPMN busi-

⁵⁶<https://www.omg.org/spec/SoaML> Retrieved: 6.12.2019

⁵⁷<https://www.eclipse.org/acceleo/> Retrieved: 6.12.2019

ness processes are then extended to become executable (Sosa-Sanchez, Clemente, Prieto, et al., 2017) by adding the invocation information of the aligned services using Apache Activiti⁵⁸.

S1 is not satisfied as phases before migration are not addressed beyond Reverse Engineering. Being a WSE approach, both the input and the result are Web Applications, satisfying S2. Risk management is not addressed; knowledge recovery is conducted only as required for the Transformations. Therefore, C1 is not satisfied. Functionality and user interaction are completely reused since the original Web Application is only internally re-structured to SOA, satisfying C2. Expertise requirements are high, considering that an ISV formerly developing a non-model-based MVC Web Application is required to perform a model-driven migration using complex Transformation technology like ATL, Acceleo to migrate into a model-driven SOA target environment. This inhibits fulfillment of C3. The proposed process does not address C4, the integration into ongoing development activities.

Table 3.21: MigraSOA Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	○	○

3.2.4 Migration to Web Applications

This group comprises approaches that support the defining adaption of Web Migration – from non-Web Legacy Systems to Web Systems – that do not belong to one of the two established fields of SOA or cloud migration. Typical source systems are desktop Text-based User

⁵⁸<https://www.activiti.org> Retrieved: 6.12.2019

Interface (TUI) or GUI applications and mainframe systems. Target Web Systems, on the other hand, are diverse, ranging from near-identical copies of legacy user interfaces in HTML to completely re-engineered model-driven Web Applications (Heil and Gaedke, 2017).

MELIS

Lucia et al. present a migration strategy (Lucia, Francese, Scanniello, and Tortora, 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006) and a tool, called MELIS (Colosimo et al., 2007), developed in the context of an industrial technology transfer project with an SME-sized ISV. MELIS is part of the METAMORPHOS project (Lucia, Penta, et al., 2009), that aims at facilitating Reverse Engineering and migration techniques and tools in the industry with a focus on Web-based target architectures. The MELIS migration strategy and tool are designed for migration of monolithic multi-user COBOL Legacy Systems to a multi-tier Web-based architecture. Due to low decomposability, an incremental migration strategy consisting of Reengineering of the UI as Web-based UI and re-structuring and wrapping of the Legacy System is proposed. The resulting target system is a Web System with a Web-based UI, however, not a Web Application since the UI is only a wrapper of the legacy GUI, therefore only half-satisfying S2. The presented migration strategy addresses phases prior to the actual migration, including technical assessment of the Legacy System to identify the degree of *decomposability*, the definition of the target environment and identification of technical problems/risks related to concrete technologies and decomposability.

Due to lack of communication of necessity and benefits, however, S1 is only half-satisfied. Basic risk management is addressed through technical risk assesment half-satisfying C1, but advanced risk management or recovery of knowledge are not described. Legacy functionality is reused due to the wrapping approach but requires manual re-structuring of the

interactive COBOL programs into separate batch programs, RMI or SOAP is used to integrate resulting components when executed distributedly. The re-engineered Web UI is automatically generated from the legacy GUI following the MVC pattern. A similar look and feel of the Web UI is reported as a constraint by the partner company, and the automatic generation process aims at producing Web UIs similar to the original ones, satisfying C2, but no systematic procedure and no concrete measures or tools are provided to reach that aim. The proposed migration strategy requires expertise in legacy technology and migration due to the necessary restructuring of the code to be wrapped and some Web expertise for the UI Reengineering, in particular for the manual adaptions of the CSS files. This inhibits fulfillment of C3 in spite of process support by the MELIS eclipse plugin, which generates the basic Web UIs and automates the deployment of the wrapped components to an application server. Integration into ongoing development (C4) is not enabled with the presented migration strategy being designed as stand-alone process. The survey on the state of practice of software migration (Torchiano et al., 2008) conducted as part of the METAMORPHOS research project (Lucia, Penta, et al., 2009) has contributed to the motivation and the problem understanding of this thesis as described in chapter 1.

Table 3.22: MELIS Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	○	○	●	○	○

TUIMigrate

TUIMigrate (Karampaglis et al., 2014) proposes an approach to migrate text-based user interface Desktop Applications to the Web. The main focus lies on mitigating security threats of the target system operating in the “hostile execution environment” of the Web without

manual modifications of the source code. Karampaglis et al. argue that moving software towards SaaS introduces security-related risks due to exposure to a large and distributed user base that may accidentally or intendedly provide harmful inputs. The proposed method employs middleware for automatic translation of user interactions between the TUI and a Web-based UI and for data sanitization to prevent the underlying backend from receiving potentially harmful inputs. TUIMigrate provides a tool for the automatic Transformation of TUIs to Web-based UIs. The TUIMigrate middleware acts as mediator, converting user interaction on the Web UI into commands for the TUI and the current state of the TUI into an XML representation which is then transformed into HTML by the Web frontend.

TUIMigrate is a very technology-specific approach that does not address phases prior to migration (S1). The target system is a Web System including a Web-based UI. However, this UI is a representation of a TUI, only half-satisfying S2. Risk management as per C1 is not addressed. Reuse is high due to the Encapsulation approach leaving legacy functionality unchanged and the automatic Transformation of the TUI to a Web-based UI maintaining the original user interaction. Therefore, TUIMigrate satisfies C3. Expertise requirements are low, resulting in fulfillment of C3, as the approach is limited to the technical perspective of transforming the UI through providing a transparent middleware. Integration into ongoing development processes required by C4 is not considered. Note that TUIMigrate combines Encapsulation – for the legacy backend – with automatic Transformation – for the UI; thus, we consider TUIMigrate overall as Transformation approach.

Table 3.23: TUIMigrate Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

M&S SW

As part of the Italian research project M&S SW (Methods and Tools for the Production of the Software, Formation, and Applications), Bodhuin et al. investigate the migration of COBOL Legacy Systems with character-based user interfaces towards MVC Web Applications (Bodhuin, Guardabascio, et al., 2002; Bodhuin, Guardabascio, et al., 2003; Bodhuin and Tortorella, 2004). The focus lies on wrapping the Legacy System and reimplementing the UI. The proposed incremental migration strategy (Bodhuin, Guardabascio, et al., 2002) consists of eight phases. The first two phases deal with COBOL-specific pre-processing of the legacy source to eliminate GOTO statements. They are followed by three Reverse Engineering phases which perform static analysis to identify the objects that represent fundamental concepts of the application domain, separate user interaction from application logic using program slicing techniques and abstract the data models from persistent objects. Automatic restructuring isolates user interaction and application logic into separate COBOL programs. The proposed toolkit creates wrappers and reimplements the GUI. Wrapped objects can be optionally reimplemented later. While the view and controller layer of MVC are generated as JSP Web Application, the model layer remains unchanged, supported by a gateway architecture mediating between the legacy and the new system. The proposed GUI Reimplementer tool supports the generation of basic HTML UIs through screen scraping (Merlo et al., 1995) of the character-based SCREEN SECTIONs in legacy COBOL programs. The approach was extended for non-decomposable systems using proxies

for user interface and database communication instead of re-structuring to MVC in (Bodhuin, Guardabascio, et al., 2003). Another extension towards grid technologies focuses on distribution aspects of the components of the target MVC architecture (Bodhuin and Tortorella, 2004). Introducing a directory service allowing to find legacy services to the target architecture, it can be seen as an early precursor to SOA migration that would gain significant attention in research about five years later.

M&S SW does not address phases prior to migration beyond the technical perspective of knowledge recovery and therefore not satisfies S1. The target system is a Web System including a Web-based UI. However, this UI is a representation of a TUI which only half-satisfies S2. C1, risk management, is not addressed. Reuse is high due to the Encapsulation of legacy functionality and automatic Transformation of the TUI to a Web-based UI maintaining the original user interaction, resulting in fulfillment of C2. Expertise requirements are low due to the narrow technical focus and near-complete automation, that completely satisfies C3. Integration into ongoing development (C4) is not considered.

Table 3.24: M&S SW Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

UWA/UWAT+

Distante et al. propose a Reengineering method for the migration of data-intensive legacy applications to ubiquitous Web Applications (Distante, Perrone, et al., 2002; Distante, Parveen, et al., 2004; Distante and Tilley, 2005; Distante, Canfora, et al., 2006; Distante, Tilley, and Canfora, 2006). The focus lies on the definition of a methodologi-

cal approach based on conceptual user-centered modeling by using the Ubiquitous Web Applications (UWA) framework and its extended transaction design model UWAT+ (Distante and Tilley, 2005). For brevity, we use UWA/UWAT+ as the short identifier for UWA/UWAT+ based Reengineering. A particular focus is on the interplay of content navigation and process execution, which is achieved through an extended conceptual model linking business process tasks to *web transactions* (Distante, Parveen, et al., 2004). The proposed Reengineering method comprises three phases: requirements elicitation, Reverse Engineering, and forward design. It starts with stakeholder analysis and goal-based requirements elicitation. The Reverse Engineering phase abstracts and formalizes knowledge from the Legacy System according to UWA metamodels. Finally, the forward design performs a model-based hypermedia re-design from the functional to the navigational paradigm, consisting of information, navigation, transaction and publishing design in UWA/UWAT+.

UWA/UWAT+-based Reengineering addresses phases prior to migration, but without consideration of communication aspects and therefore only half-satisfies S1. The target system is a Web Application including a Web-based UI, completely satisfying S2. Risk management (C1) is not addressed but knowledge recovery is present. Reuse is high both for legacy functionality and user interaction due to systematic Reverse Engineering of business processes and focus on the similarity of user interactions. Thus, C2 is satisfied. Expertise requirements are high due to the complexity of UWA/UWAT+ methodology and metamodels and lack of dedicated tools support, not satisfying C3. Integration into ongoing development for C4 is not considered.

Table 3.25: UWA/UWAT+ Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	○	○

CeLEST

The Canadian CEL⁵⁹ Legacy Enhancement Software Technologies (CeLEST) (Stroulia, El-Ramly, Iglinski, et al., 2003; Stroulia, El-Ramly, and Sorenson, 2002; Stroulia and Kapoor, 2002; El-Ramly et al., 2002; Stroulia, El-Ramly, L. Kong, et al., 1999; L. Kong et al., 1999) research project aims at the migration of legacy mainframe applications to the Web. It focuses on the migration of legacy terminal user interfaces by employing Reverse Engineering techniques to recover functionality in terms of user tasks, capturing these in interaction models and constructing new Web-based UIs. The CeLEST method allows to reverse engineer an executable service specification based on traces of users' interactions with the Legacy System. Similar to screen scraping approaches, the parts of the Legacy System are exposed (wrapped) to the new system; however, the executable model of legacy interface behavior is derived in a semi-automatic way. Communication between the Web-based frontend and the Legacy System is related via a translating proxy. The CeLEST process proposes five steps: 1) execution of the Legacy System in an emulator to collect interaction traces, 2) behavior modeling using state-transition-models, 3) task discovery through pattern analysis, 4) task modeling through information-exchange analysis and 5) Web GUI specification based on the identified functionalities. The creation of the state-transition model (2) employs semi-automatic clustering of similar UI snapshots to identify compound states using a custom feature set (Stroulia, El-Ramly, L. Kong, et al., 1999). A sequential pattern-mining

⁵⁹abbreviation for Celcorp, the main industrial sponsor

algorithm (El-Ramly et al., 2002) identifies task execution patterns based on recurring similar state-transition sequences (3), which are reviewed by an expert. Supported by manual annotation, inputs and outputs are identified, the command language is learned, and the task model is created manually (4). CelLEST automatically creates an XML specification of a form-based user interface that is executed in runtime interpreters translating to XHTML or WML.

S1 is not satisfied as CelLEST does not address phases prior to migration beyond knowledge recovery. The target system is a Web System including a Web-based UI; however, the UI is a representation of a TUI and thus S2 is only half-satisfied. Risk management (C1) is not addressed; knowledge recovery is present in terms of interaction modeling. Reuse is high both for legacy functionality and user interaction due to the proposed Encapsulation approach and explicit modeling of user interactions, satisfying C2 completely. Expertise requirements are high despite the simple inductive process, and extensive automation tools support (Stroulia, El-Ramly, and Sorenson, 2002), since manual task modeling and expert reviews of unsafe inferences are required. This inhibits fulfillment of C3. Integration into ongoing development as in C4 is not considered.

Table 3.26: CelLEST Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	○	○

DAS

B. Chen et al. (2016) propose a desktop application service (DAS) framework that aims at hosting a legacy Desktop Application on a Web server and displaying its GUI in a browser. The focus lies on desktop virtualization of applications instead of entire desktops through

standard Web protocols. To achieve this, DAS is implemented as a Web server module, providing URL-based access and instantiating the desktop applications. It functions as a mediator, forwarding user input and graphical output between the browser and the applications. DAS consists of a lifecycle manager handling instantiation and termination of desktop applications and a set of desktop UI-library-specific service agents that handle the input/output conversions. A service agent is proposed, which enables translation between GTK+ and HTML5.

DAS is a virtualization solution without explicit process model and belongs to the Migration & Transition phase of ReMiP. Thus phases prior to migration (S1) are not addressed. The target system is a Web System including a Web-based UI, however, this UI is an identical representation of the virtualized desktop UI and therefore does not support Web functionality such as bookmarking which only half-satisfies S2. C1 is not satisfied since risk management and knowledge recovery are not addressed. Reuse is high both for legacy functionality and user interaction due to the Encapsulation approach, therefore satisfying C2. Expertise requirements are low due to lack of a complex migration process and a tool-centric approach, resulting in fulfillment of C3. C4, the integration into ongoing development, is not considered.

Table 3.27: DAS Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
○	●	○	●	●	○

3.3 Analysis of Assessment Results

The Web Migration approaches have been assessed individually in section 3.2. Table 3.28 provides an overview of the assessment results.

While some commonalities with regard to the requirements can be identified, the migration approaches also show unique characteristics that cannot be captured on a per-group basis. Since software migration approaches are typically tailored to a specific environment (cf. *brownfield software engineering* in Hopkins and Jenkins, 2008), a high number of parameters determine a specific migration approach, e.g. Migration Type, Migration Discipline, Legacy Environment, Target Environment (Heil and Gaedke, 2017). In addition to the two orthogonal dimensions – target architecture and methods – as introduced in section 3.2, also cross-cutting characteristics impact the Web Migration approaches. The following analysis of the assessment of Web Migration approaches therefore reports findings relevant for the solution proposed in this thesis grouped by

- requirements,
- research areas and target architectures,
- overall methodologies, and
- cross-cutting characteristics.

Table 3.28: Evaluation Overview, target WA=Web Application, requirements
S1 Initial phase support, S2 Web Application Target, C1 Risk
management, C2 Reuse of legacy assets, C3 Expertise & Tool
Support, C4 Agile Development Process Integration

Approach	Target	Method	S1	S2	C1	C2	C3	C4
SMART	SOA	Other	○	○	●	○	●	○
SAPIENSA	SOA	Trans	○	○	○	○	○	○
serviciFi	SOA	ReEng	○	○	●	○	○	○

Approach	Target	Method	S1	S2	C1	C2	C3	C4
Marchetto2008	SOA	Encaps	○	●	○	●	●	○
SOAMIG	SOA	Trans	●	○	●	●	○	○
Gaps2Ws	SOA	Encaps	○	●	○	●	●	○
PRECISO	SOA	Encaps	○	●	○	○	●	○
AWS Migration	Cloud	Other	●	○	●	○	○	○
REMICS	Cloud	Trans	●	●	●	●	○	●
ARTIST	Cloud	Trans	●	●	●	●	○	○
CloudMIG	Cloud	Trans	○	●	○	●	○	○
IC4	Cloud	ReEng	●	●	●	●	○	○
AMS	Cloud	Encaps	○	●	○	●	●	○
NCHC	Cloud	Encaps	○	●	○	●	●	○
L2CMH	Cloud	Trans	●	●	●	●	○	○
MIGRARIA	WSE	Trans	○	●	○	●	○	○
MigrasOA	WSE	Trans	○	●	○	●	○	○
MELIS	WA	Encaps	●	●	●	●	○	○
TUIMigrate	WA	Trans	○	●	○	●	●	○
M&S SW	WA	Encaps	○	●	○	●	●	○
UWA/UWAT+	WA	ReEng	●	●	○	●	○	○
CelLEST	WA	Encaps	○	●	○	●	○	○
DAS	WA	Encaps	○	●	○	●	●	○

3.3.1 Results by Requirements

This section briefly describes common observations with regard to the requirements defined in section 2.3.

S1 Initial Phase Support. The majority of Web Migration approaches do not address the initial phases (cf. also Heil and Gaedke, 2017). Among those which do so, the focus is on the economic viability perspective (Khadka, Reijnders, et al., 2011; Khadka, 2016) in terms of

costs/risks/constraints (Razavian and Lago, 2012). Communication of migration benefits is widely overseen. Even comprehensive methodologies like ARTIST and AWS Migration – which comprise general consideration of benefits through analysis of customer needs using interviews, surveys, etc. – do not propose methods to provide concrete, tangible means of communicating migration benefits to support analysis of customer and ISV benefits prior to migration decision making.

S2 Web Application Target. All assessed approaches target a Web System, therefore belonging to the Web Migration scope defined in section 1.4. Thus the requirement achieves a high overall fulfillment ratio of about 0.6. The differences, however, are in the consideration of Web-based user interfaces. Only 6 out of 23 approaches explicitly address this migration aspect. Among them the approaches overall rated the highest, REMICS and UWA/UWAT+, provide UI designer tools and guidelines. Few approaches like TUIMigrate, DAS, and NCHC address user interface migration but do not target real Web-based user interfaces. Instead, they either provide character-based terminal user interfaces in the browser, or establish browser-based wrappers of graphical desktop user interfaces, thus not satisfying the requirement. The majority of approaches disregards migration to Web-based user interfaces entirely, focusing on backend aspects only.

C1 Risk Management. Risk management practices have not been observed frequently with only SMART and ARTIST comprising strategies that contribute to the business case and secure existing knowledge against loss during migration. IC4 has a dedicated focus on risk assessment through experimentation-oriented feasibility studies. Other approaches either do not address risk or are restricted to technical feasibility analysis. In contrast to academia, industrial practice often relies on interviewing stakeholders for knowledge elicitation, such as

locating functionality in legacy code while neglecting Reverse Engineering (Razavian, 2013; Razavian and Lago, 2012). However, stakeholders of long-running Legacy Systems, as in section 2.1, are no longer available, so the legacy source is often “the only source of domain knowledge” (Bodhuin, Guardabascio, et al., 2002).

C2 Reuse of Legacy Assets. While reuse of legacy artifacts for maintaining functionality is high, user interaction is disregarded. This lack of consideration of the impact on existing users is in contrast with the importance of a “similar look and feel” for industry (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, and Tortora, 2008; Distante, Perrone, et al., 2002). There are no concrete metrics or tools to support achieving this objective. Dedicated migration approaches for user interfaces are mainly focused on either character-based TUIs to the Web (M&S SW, TUIMigrate), where a nearly literal translation is simple and successful or on browser-based desktop virtualization wrappers (DAS, Gaps2Ws, AMS), both of which create results with limited suitability for the Web (Distante, Tilley, and Canfora, 2006).

C3 Expertise & Tool support. The feasibility of the assessed Web Migration approaches with existing staff lacking expertise in target technology, and software migration is limited. Many approaches require the application of complex migration techniques and tools and good knowledge of the target technology. This particularly holds for model-driven migration approaches implementing ADM standards like REMICS or ARTIST and approaches with a model-driven target architecture like UWA/UWAT+. While tool support exists for about 70% of the approaches, only half of the approaches satisfy the expertise requirements. Many of these approaches follow an Encapsulation strategy, making them easier to use at the cost of the shortcomings of wrappers as described in section 3.3.3.

C4 Agile Development Process Integration. Agile integration was only observed in one approach: REMICS. In REMICS, however, it is not directly described. Its agile extensions specify a mapping onto Scrum, which facilitates integration with an ongoing agile development process and environment. The same characteristics are found in Reengineering outside the Web Migration field, with approaches like PARFAIT (Cagnin et al., 2003) defining stand-alone agile Reengineering processes, in this case from procedural to object-oriented business resource management systems. Integration aspects for the most labor-expensive category, Reengineering, are not addressed in research. Due to the differences in the nature of any Software Modernization or Migration in comparison to Forward Engineering, integration is hard to achieve. However, even partial integration would benefit the applicability for ISVs with limited resources.

3.3.2 Results by Research Areas and Target Architectures

This section briefly summarizes commonalities of assessed approaches in groups defined by research area and target architecture.

SOA. The analyzed SOA migration approaches stress the importance of identifying the functionalities (i.e. services) of a legacy Web System. They are concerned with capturing functionality in technology-independent models and identifying the business processes which define and make use of it. The legacy Web System can be considered a *service repository* (Sosa-Sanchez, Clemente, Sanchez-Cabrera, et al., 2014). Exposing legacy code as services through the two dominant methods of Encapsulation or Transformation requires a good separation of *UI code* from *business or mission function code* (G. Lewis, Morris, Smith, et al., 2008). Otherwise, replacement is more cost-effective. SOA as target architectural style focuses more on the reuse of exposed functionality across several systems in a larger enterprise software environment than

on reuse of the functionality of single systems (G. Lewis, Morris, Smith, et al., 2008). The emphasis on backend functionality leads to dominantly medium ratings for S2 Web and C2 Reuse. Initial phases (S1 Init) are rarely considered. Reverse Engineering is neglected in the industry for SOA migration (Razavian and Lago, 2012). The consideration of business aspects results in higher use of risk management mechanisms (C1 Risk) compared to Cloud, WSE, and Web application target groups, employing feasibility studies, portfolio analysis, and pilots. C3 Exp results do not show similarities, C4 Agile is not addressed.

Cloud migration approaches show the highest support of phases prior to migration (S1 Initial) of all groups due to cloud-related additional planning activities such as resource allocation and decision making. However, only two approaches address business case support through the communication of benefits. The Cloud group also comprises the most comprehensive migration methodologies, including the results of two EU FP7 projects: REMICS and ARTIST. However, cloud migration research is often focused on migrating on-premise software of large companies; the ISV perspective is seldom represented (Fowley, Elango, et al., 2017). Compared to SOA's exclusive backend focus, Cloud migration approaches reach higher ratings for S2 Web due to consideration of UI aspects in some approaches that are not limited to desktop-virtualization variants. Risk management (C1 Risk) is addressed slightly more than in SOA migration, and expertise requirements (C3 Exp) are higher. C3 Reuse achieves medium ratings with only two approaches addressing user interaction reuse. With REMICS agile extensions, the Cloud group contains the only approach addressing integration into ongoing development (C4 Agile) to some degree.

WSE. The Web Systems Evolution approaches unanimously excel in S2 Web and C2 Reuse and fail in all other requirements. Since the

source environment is already a Web System that is modernized into a newer Web-based target architecture, WSE approaches achieve full rating for the target environment. Likewise, their Transformation approaches make complete reuse of existing assets to maintain both functionality and interaction. Requirements S1 Init, C1 Risk, C3 Exp and C4 Agile are not addressed.

Web Application. A poor support of phases before migration (S1 Initial) was observed in the mainly tool-focused Web Application migration approaches analyzed. Similarly, due to their technical focus, risk management (C3 Risk) is not present. They rate second highest with regard to the target environment (S2 Web), however not as high as expected due to the resulting Web Applications often being identical representations of Desktop Applications instead of Web Applications with real Web UIs. In terms of reuse of existing assets (C2 Reuse) they rate outstandingly high as the objective of these tool-based approaches is to transfer functionality and interaction into a Web-based environment. Risk (C1 Risk) is hardly addressed, and integration into agile development (C4 Agile) not at all. Mainly originating from the mid-2000s and with text-based source Legacy Systems, the migrated target systems have to be considered outdated in terms of technology (e.g. RMI) and user interaction (e.g. textual console). A certain neglect of the foundational Web Application migration field in favor of SOA and Cloud is evident (cf. also Heil and Gaedke, 2017).

3.3.3 Results by Overall Methodologies Used

This section provides a short overview of commonalities of Web Migration approaches in groups defined by overall methodology.

Encapsulation. Many Web Migration approaches are using wrapping approaches that create a hybrid Legacy System which does not miti-

gate legacy risks, since the fundamental characteristics of the Legacy System are not changed (Almonaies et al., 2010) and they lead to *architectural degradation*, where the “new system delivers lower quality than pre-existing system” (Razavian, Nguyen, et al., 2010), and the Legacy System must still be maintained in addition to the wrappers (Fuhr et al., 2013), typically increasing the technological heterogeneity. They propose a *Web Migration without modernization* and are therefore limited to the adaptive perspective, failing to address the required perspective dimension of Web Migration (ISO/IEEE, 2006). The resulting Web-enabled Legacy System does not fully achieve the Web Application target requirement S2 Web. Wrapping is only applicable if legacy functionality should remain completely unchanged. Otherwise, program understanding and re-structuring are required (Stroulia, El-Ramly, and Sorenson, 2002). Thus, Encapsulation approaches reach the highest levels of reuse of existing assets (C2 Reuse). As Encapsulation approaches often see migration from a system integration perspective, employing solution patterns from EAI (Enterprise Application Integration) like gateways, proxies (Karampaglis et al., 2014; Bodhuin, Guardabascio, et al., 2003), decomposability is the pre-requisite (Wagner, 2014; Khadka, 2016; Khadka, Saeidi, et al., 2013; Lucia, Francese, Scanniello, and Tortora, 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006). Software-wrapper solutions like DAS, Gaps2Ws, AMS are very similar and limited to a kind of application-specific desktop virtualization in the browser. Thus, initial phases (S1 Initial) and risk management (C3 Risk) are nearly not addressed. Due to the simplicity, however, expertise requirements (C3 Exp) are low.

Transformation-based Web Migration approaches have a high level of reuse for functionality, but user interaction is mostly not regarded (C2 Reuse). The automatic Transformations are very platform-specific and, therefore, have only limited applicability. The majority of Trans-

formation approaches follow the model-driven paradigm, extending the objective of reuse to the Transformations themselves, which implies high expertise requirements (C3 Exp) due to the complex model-driven technologies used. Due to their technical focus, support for initial phases (S1 Initial) and risk management (C4 Risk) is not widely observed in this group. Similar to Encapsulation approaches, decomposability is crucial (Khadka, 2016; Khadka, Saeidi, et al., 2013; Lucia, Francese, Scanniello, and Tortora, 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006). Transformation often requires additional prior Reengineering (Nasr et al., 2010). No characteristic commonalities for requirements S2 Web and C1 Risk are observed.

Reengineering. Both Encapsulation and Transformation approaches for Legacy Systems with low decomposability require re-structuring (Nasr et al., 2010; Aversano et al., 2001; Wagner, 2014; Khadka, 2016; Khadka, Saeidi, et al., 2013; Lucia, Francese, Scanniello, and Tortora, 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006). Their advantages in terms of reduced effort are quickly mitigated so that Reengineering becomes more viable. Reengineering approaches have the highest support for phases prior to migration (S1 Initial), and reuse of legacy assets is high (C2 Reuse) due to their objective of changing a Legacy System on different levels of abstraction that requires thorough planning and system understanding through implied Reverse Engineering. However, the complex architectural changes also demand high expertise and involve a significant amount of manual activities (C3 Exp). Integration into ongoing development (C4 Agile) is not considered, the results for S2 Web and C1 Risk show no group-specific commonalities.

3.3.4 Results by Cross-cutting characteristics

As seen in the above analysis, the analysis results in the two groupings share common characteristics within their groups. However, there also

cross-cutting observations that cannot be directly aligned with the groupings. Therefore, this section briefly summarizes common observations in the state of the art of Web Migration with regard to characteristics cross-cutting the groupings presented above. These characteristics mainly affect the expertise requirement C3 Exp and provide additional input to the formation of the solution in the following chapter.

Low Abstraction. Many approaches (e.g. MELIS, SoaMIG, M&S SW, Marchetto2008) operate at a low abstraction level close to code (cf. levels of abstraction in ADM and SOA-MF (Razavian and Lago, 2010)), “hindering reusability and maintainability of the obtained system” (Sosa-Sanchez, Clemente, Sanchez-Cabrera, et al., 2014) and focus on technical aspects of Encapsulation and Transformation. On the other hand, sophisticated methods like ARTIST, REMICS, UWA/UWAT+ consider business, organizational, and architectural abstractions but have high expertise requirements (C3 Exp).

Model-driven approaches. Half of the surveyed approaches, in particular Transformation approaches, are based on the model-driven paradigm, i.e. they provide formal definitions for metamodels and Transformation languages for instances of these metamodels (Fuhr et al., 2013). They employ model-driven technologies from OMG or the Eclipse Modeling Project (EMP)⁶⁰: the proposed metamodels are mainly UML profiles and MOF-compliant (OMG) or KM3-compliant (EMP). Transformation languages such as QVT (OMG) and ATL (EMP) are common. Especially OMG’s ADM family languages are widely used among model-driven migration approaches, as indicated in table 3.1. Most model-driven methodologies (e.g. ARTIST, REMICS, SoaMIG, MELIS) are focused on the Java platform for source and target system and are built around Eclipse modeling tools like MoDisco (Pérez-Castillo, Guzman, et al.,

⁶⁰<https://www.eclipse.org/modeling/> Retrieved: 6.12.2019

2011). The model-driven objective of achieving reuse of not only design artifacts but also the model Transformations introduces a high level of complexity, which results in high expertise requirements (C3 Exp) for both the migration and the target environment. In particular, it is assumed that the target system will be evolved in a model-driven way, which requires a fundamental change for ISVs previously not employing model-driven development.

SME-sized ISVs. The finding of our 2017 systematic mapping study (Heil and Gaedke, 2017), that the perspective of ISVs – in particular small and medium-sized ISVs – is hardly considered still holds. IC4 does so explicitly, but due to lack of concrete Reengineering methodology and tool support places high expertise requirements, lacks communication of migration benefits, systematic knowledge recovery, and reuse of interaction-related assets. Many approaches are only demonstrated with small, artificial sample applications (e.g. conference management (Sosa-Sánchez, Clemente, Sanchez-Cabrera, et al., 2014), Java Pet Store Demo (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012)) that do not represent all legacy characteristics, mainly representing legacy technology but lacking size, structural degradation, and Technical Debt. Likewise, characteristics of SME-sized ISVs (cf. section 2.1.1) are neglected.

Web Migration Methodologies. Comprehensive approaches like ARTIST, SAPIENSA, REMICS, and UWA/UWAT+ with full coverage according to ReMiP and popular approaches like SMART are specified as *methodologies*: they provide a set of principles, formalisms, methods, and tools (Wallmüller, 2001) for Web Migration. This structure provides an implicit mechanism for extension through integration with other methodologies. For instance, REMICS focuses on recovery, migration, and deployment. To address the initial phases of Web Migration, it integrates methods from the SMART

methodology. Package-oriented (Brodie and Stonebraker, 1995), “*feature-driven iterative migration*” (Menychtas, Konstanteli, et al., 2014) in increments is the dominant process model.

3.4 Shortcomings of Existing Approaches

The shortcomings in the current state of the art in Web Migration identified above make it difficult for SME-sized ISVs with legacy, non-Web, Desktop Applications, and large existing user bases to commence a Web Migration. Existing approaches do not sufficiently address their doubts about feasibility and desirability that make them hesitant to take the risks of a migration project.

The following three main gaps in Web Migration research summarize the shortcomings identified in several independent dimensions of grouping presented in section 3.3:

- G1** Lack of feasibility with limited resources and limited migration and target environment expertise
- G2** Lack of demonstration of desirability for decision making and risk management in initial phases
- G3** Lack of user interface migration and user interaction reuse

Figure 3.4 shows the relationship between the identified gaps and the requirements from section 2.3.

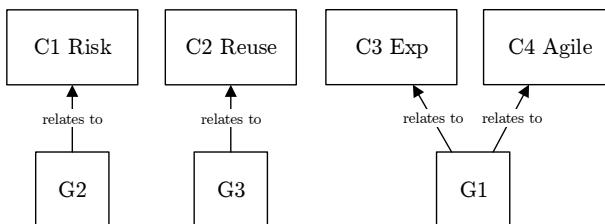


Figure 3.4: Concept Map of Relationships between Gaps and Requirements

G1 Lack of feasibility with limited resources and limited migration and target environment expertise

Existing Web Migration approaches have high requirements with regard to required resources and expertise. Integration into ongoing, agile development activities is widely overseen with all approaches designed as stand-alone processes requiring dedicated resources. Feasibility of the approaches with existing staff lacking expertise in target technology and software migration (C3 Exp) is only observed for Encapsulation approaches and Transformation for TUIs, that have severe drawbacks, as described in section 3.3.2. The most promising comprehensive approaches in terms of overall rating (i.e. REMICS, ARTIST, UWA/UWAT+) all have high expertise requirements.

G2 Lack of demonstration of desirability for decision making and risk management in initial phases

Existing Web Migration approaches fail to support decision making in initial phases and address risk. Their focus is on technical feasibility and financial viability. Demonstration of desirability of a Web-based version is not considered, leaving decision making without appropriate input specific for the software system to be migrated. Risk management is hardly addressed. Risk management techniques focus on technical feasibility studies. Existing knowledge is not systematically secured against loss during migration beyond stakeholder interviews. With stakeholders of long-running Legacy Systems often no longer available, this is not sufficient.

G3 Lack of user interface migration and user interaction reuse

Existing Web Migration approaches fail to address user interface aspects of Web Migration. The majority focuses on the backend and does not support migration towards a Web-based user interface. When regarded,

the user interfaces are not native Web user interfaces but terminal user interfaces, or wrappers of graphical desktop user interfaces in the browser. Reuse of user interaction and continuity of the user interface are neglected. There are no concrete metrics or tools to achieve a similar “similar look and feel” in spite of identified relevance for industry.

To conclude, there are many partial solutions in the Web Migration field, but even comprehensive methodologies like ARTIST, REMICS, UWA/UWAT+ exhibit shortcomings in requirements C1, C3 and C4, cf. table 3.28, and would benefit from a dedicated solution addressing these gaps.

3.5 Summary

This chapter presented relevant standards and a reference model for migration, and assessed existing Web Migration approaches according to the requirements elicited in chapter 2. A two-dimensional grouping scheme was introduced and a detailed analysis of observations was presented. This analysis has revealed the lack of a suitable solution combining support of initial phases and Web Application target architecture with suitability for SME-sized ISVs in terms of risk management, reuse of functionality and user interaction, available expertise and resources, and integration into ongoing agile development. These shortcomings were synthesized in three main gaps, demonstrating the relevance and need for the solution presented in the following chapter. The detailed findings have provided input for the creation of the solution concept.

Addressing Effort and Risk Concerns in Web Migration

4

This chapter outlines the AWSM (Agile Web Migration for SMEs) (Heil and Gaedke, 2016) approach providing a solution to address the shortcomings summarized in G1, G2, and G3 which were identified in the state of the art of Web Migration that keep ISVs, in particular SME-sized ISVs, from commencing a Web Migration. To devise a suitable solution addressing these gaps, section 4.1 identifies the main research objectives of this thesis. Section 4.2 derives the AWSM solution achieving these research objectives, consisting of the AWSM Methodology and Toolsuite.

4.1 Research Objectives

As identified in section 3.4, Web Migration research currently lacks in three areas: the feasibility with limited resources and limited migration and target environment expertise (G1), the demonstration of desirability for decision making and risk management in initial phases (G2), and the user interface migration and user interaction reuse (G3). These gaps correspond to the *doubts about feasibility* and *doubts about desirability* for ISVs with non-WebLegacy Systems and large existing user bases observed in the scenario in section 2.2.2. In order to devise a solution that addresses the gaps and supports ISVs to commence Web Migration, a set of research objectives that outline suitable key solution characteristics for the stakeholder-specific problems needs to be derived. To drive

ideation of solution ideas that represent research objectives, the *How-Might-We* method of HCD is applied. HMW questions were formulated in a collaborative *brainstorm* session. The initial HMW question was:

How might we address IVS's doubts about feasibility and desirability?

By considering the three main gaps in the state of the art of Web Migration in section 3.4 and the detailed problem in section 2.2.2, this question is further broken down into the HMW questions listed in table 4.1.

Table 4.1: How-Might-We-Questions for Ideation of Research Objectives

ID	HMW-Question: <i>How might we...</i>
HMW1	reduce the risk of loss of knowledge through Web Migration?
HMW2	demonstrate feasibility and advantages of a Web-based version of the Legacy System?
HMW3	reduce the risk of customers impact through Web Migration?
HMW4	address HMW1, HMW2 and HMW3 with limited resources and lack of Web Engineering expertise?

Questions HMW1 - HMW3 address the problems of feasibility and desirability doubts relating to G2 and G3 from a *functional* (ISO/IEEE, 2017b) perspective. HMW4 is a *non-functional* (ISO/IEEE, 2017b) cross-cutting constraint, representing the resource and expertise problems in G1 and therefore is integrated with all following research objectives. The overarching goal of this thesis is:

Overarching Research Goal OG

To provide methods, models, and tools that support ISVs with limited resources and lack of Web Engineering expertise to commence a Web Migration.

To achieve this goal, three specific research objectives are identified from the HMW questions in table 4.1 employing HCD-based opportunity ideation. These objectives are:

Research Objective RO1

To enable identification and management of existing knowledge in legacy source code with limited resources and lack of Web Engineering expertise.

Research Objective RO2

To enable demonstration of desirability and feasibility of a potential Web-based version of the Legacy System with limited resources and lack of Web Engineering expertise.

Research Objective RO3

To enable control of the impact of user interface changes through Web Migration on customers with limited resources and lack of Web Engineering expertise.

Each research objective addresses at least one of the gaps identified in section 3.4 as visualized in fig. 4.1 The solution specified in the following section addresses these three research objectives.

The solution ideas corresponding to the research objectives were communicated by *visual* representations (cf. paper prototypes in section 5.4.2)

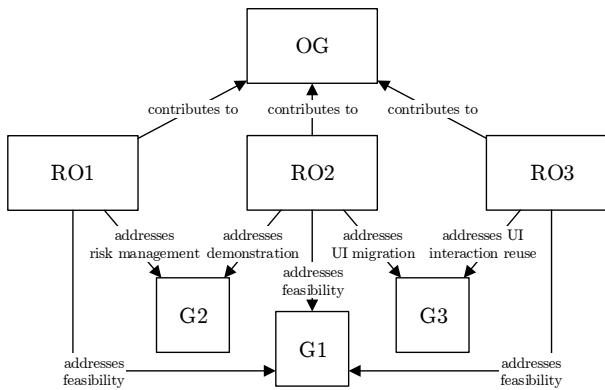


Figure 4.1: Relationships between Research Objectives and Gaps

and software *prototypes*. HCD *Feedback Integration and Iteration* was used to improve the solution ideas through stakeholder feedback gathered in a presentation and discussion session at the ISVs headquarters and continued throughout the entire research project.

4.2 Overview

Our solution to address RO1 to RO3 is *AWSM (Agile Web Migration for SMEs)* (Heil and Gaedke, 2016). AWSM addresses shortcomings of existing Web Migration approaches in initial phases of migration projects.

To achieve this, it provides solutions for each of the research objectives representing the three **AWSM Methods**:

- the AWSM Reverse Engineering Method for RO1,
- the AWSM Risk Management Method for RO2, and
- the AWSM Customer Impact Control Method for RO3.

AWSM is structured according to the constructive elements of software engineering for quality ensurance (Wallmüller, 2001)

into Methods, Tools, Principles, and Formalisms. Figure 4.2 shows the overall architecture of AWSM. The Methods, Tools, Principles, and Formalisms are grouped into two main parts: the AWSM Methodology, and the AWSM Toolsuite.

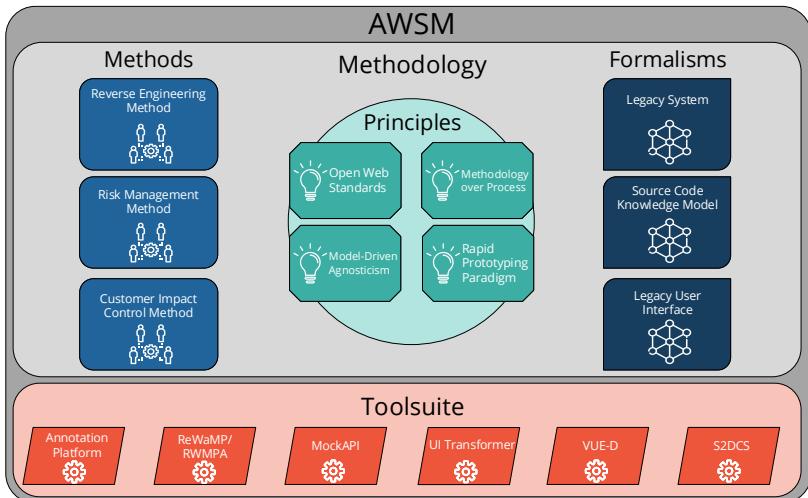


Figure 4.2: AWSM Solution Overview

The AWSM Methodology describes AWSM's Methods, Principles, and Formalisms. The Tools supporting the AWSM Methodology constitute the AWSM Toolsuite. AWSM does not impose a specific migration process. Thus, the AWSM Methodology focuses on principles, formalisms, and methods for Web Migration initiation and integration with existing comprehensive migration methods. AWSM can be combined with incremental Reengineering and Transformation processes, addressing the widely used package-oriented, feature-driven incremental migration process model characterized in section 3.3.4. For selecting or creating a suitable migration approach, AWSM provides a decision support system, as described in section 4.4.1.

The AWSM Methods are integrated into the selected migration approach, as shown in fig. 4.3. For consistent description, we apply the taxonomy from the ReMiP reference model (Harry M. Sneed et al., 2010b) throughout this thesis. The migration approach is divided into distinct sequential or parallel *migration phases*. These phases comprise at least one *migration activity* which employs one or more *migration methods*. The three methods of the AWM Methodology described in the following provide solutions to address the identified gaps in current Web Migration approaches through integration at the activity level. For each AWM Method, at least one *migration technique* is specified, detailing how to conduct a part of the method.

AWSM Tools support each technique and are the parts of the AWM Toolsuite. **AWSM Principles** are at the foundation of AWM. They are derived from the requirements and state of the art analysis results, and represent top-level design decisions of the AWM Methodology and Toolsuite. **AWSM Formalisms** form the conceptual basis for the Methods and Tools. They specify the underlying model and semantics for knowledge in legacy source code and the definition and algorithm for measuring the similarity between original and migrated user interfaces. The following four subsections describe the solutions provided by the AWM Methods, Tools, Principles, and Formalisms.

4.3 Methods

To address the research objectives, AWM provides three methods:

- AWM:RE – the AWM Reverse Engineering Method,
- AWM:RM – the AWM Risk Management Method,
- AWM:CI – the AWM Customer Impact Control Method.

These three methods provide techniques for solving the problems identified in section 3.4 in different disciplines and phases of Web Migration.

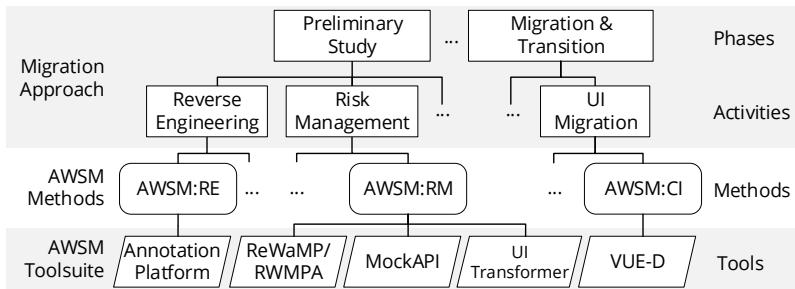


Figure 4.3: AWSM in Context, mapped to ReMiP Taxonomy

While the techniques of AWSM:RE and AWSM:RM belong to the initial phase, AWSM:CI techniques are conducted during Migration & Transition. The three methods cover several core and base disciplines each, but all are targeting the migration decision milestone. Table 4.2 provides an overview mapping of the AWSM Methods mapping them onto the migration phases and disciplines of the Reference Migration Process (Harry M. Sneed et al., 2010a) in order to facilitate integration with existing Web Migration approaches. The following three subsections outline the three AWSM Methods.

Table 4.2: Mapping of AWSM Methods to ReMiP

Method	Phase	Core Disciplines	Base Disciplines
AWSM:RE	Preliminary Study	Legacy Analysis, Requirements Analysis	Configuration & Change Management, Project Management, Migration Environment
AWSM:RM	Preliminary Study	Requirements Analysis, Target Design	Project Management, Staff Qualification

Method	Phase	Core Disciplines	Base Disciplines
AWSM:CI	Migration & Transition	Target Design, Implementation	Migration Environment

4.3.1 AWM Reverse Engineering Method

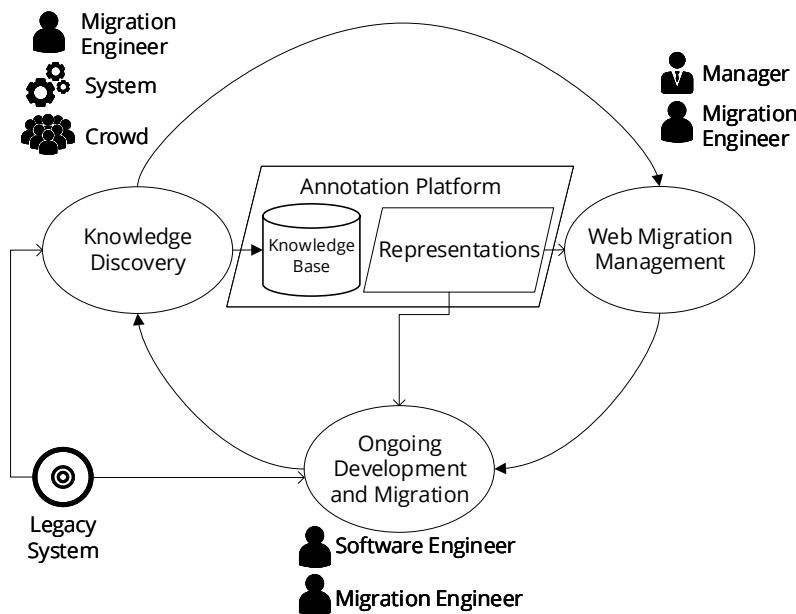


Figure 4.4: AWM:RE Method Overview

The AWM Reverse Engineering Method (AWM:RE) shown in fig. 4.4 provides a solution for RO1 by specifying techniques to identify and manage problem and solution domain knowledge in legacy source code. AWM:RE addresses the functional aspect of knowledge identification and management as well as the non-functional constraint of limited

resources and expertise of RO1. The method extracts knowledge from Legacy System artifacts based on the conceptual models for a Legacy System and knowledge in Legacy Systems introduced as formalisms in section 4.6.1. AWSM:RE solves RO1 through

- a process for incremental knowledge discovery from legacy codebases integrated with ongoing development
- a queryable knowledge representation supporting Web Migration processes based on Reengineering or Transformation at different degrees of model-driven adoption
- a model for crowdsourcing Reverse Engineering activities

Knowledge discovery in AWM:RE is solved through a Concept Assignment-based Reverse Engineering technique. It allows annotators to connect extracted knowledge to its distributed occurrences in the legacy codebase. AWM:RE specifies a unified process model for human, automatic and crowd-based annotators. The process model is incremental and integrated into ongoing development activities of ISVs. The annotation process supports identification of relevant components for migration, elicitation of architectural knowledge, and increasing decomposability of legacy code in the initial phase of Web Migration.

AWSM:RE represents reverse-engineered knowledge in a queryable, interoperable way using semantic Web technologies to provide a basis for subsequent Reengineering/Transformation approaches. Its Web Ontology Language (OWL)-Ontology for knowledge in legacy source code allows adaption of AWM:RE to different Web Migration methods at varying degrees of model-driven adoption. The ability to query the reverse-engineered knowledge in a technology-independent form ensures applicability with a wide range of existing Web Migration approaches and tools.

Unlike existing redocumentation and design recovery methods, AWSM:RE has a strong focus on applicability for ISVs with limited resources and limited Web Engineering expertise. To achieve this, AWSM:RE introduces the novel idea of *Crowdsourced Reverse Engineering* (Heil, Förster, et al., 2018) and demonstrates applicability by re-formulating the problem of Concept Assignment (Biggerstaff et al., 1994) as classification problem that can be solved using the *Crowdsourcing paradigm* (Howe, 2006). While Crowdsourcing has seen adoption in forward software engineering, AWSM is the first approach to transfer the paradigm to Reverse Engineering for Web Migration. To make use of the benefits of the crowd with regards to workforce and expertise, AWSM:RE addresses the challenges of controlled disclosure and quality control. AWSM:RE is described in detail in chapter 5.

4.3.2 AWSM Risk Management Method

The AWSM Risk Management Method (AWSM:RM) shown in fig. 4.5 provides a solution for RO2 by specifying techniques to demonstrate desirability and feasibility of a potential Web-based version of the Legacy System. AWSM:RM addresses the functional aspect of desirability and feasibility demonstration as well as the non-functional constraint of limited resources and expertise of RO2. The method transforms existing software artifacts defined in the conceptual model for a Legacy System specified in section 4.6 into a running Web-based prototype. AWSM:RM solves RO2 through

- a process for rapid, semi-automatic creation of web migration prototypes, supported by API Prototyping
- a technique for business logic reuse in web migration prototypes based on WebAssembly
- a technique for automatic Transformation of legacy user interface to Web-based user interface prototypes

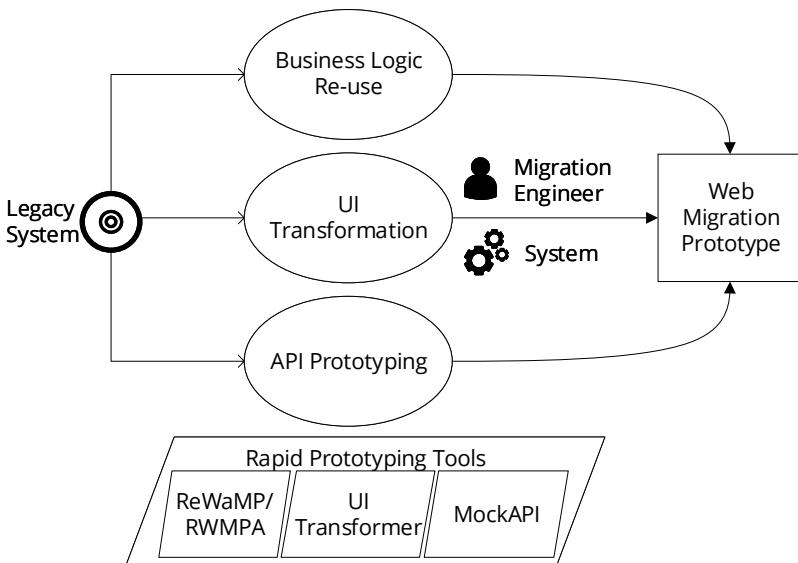


Figure 4.5: AWSM:RM Method Overview

Unlike existing risk management methods in the early stages of Web Migration, AWSM:RM allows to create a concrete and tangible contribution to the business case that serves as a means of communication for stakeholders to support decision making. To achieve this, AWSM introduces the novel idea of *Rapid Web Migration Prototyping* (Heil, Siegert, et al., 2018) by transferring the *Rapid Prototyping paradigm* (Gordon and Bieman, 1995) from Forward Engineering into the Web Migration domain. AWSM:RM defines a process for rapidly creating Web Migration prototypes based on the Legacy System artifacts.

To address the constraint of limited resources and Web Engineering expertise in RO2, AWSM:RE leverages the WebAssembly W3C standard (W3C, 2019) to allow reuse of existing business logic. In this way, the

semi-automated prototyping technique can be applied by ISV's existing staff with experience in the legacy technology. Additional guidance is provided to reduce the required Web Migration expertise.

For creating a Web-based user interface prototype, AWSM:RM allows to automatically transform legacy user interfaces based on an evolutionary optimization algorithm. It addresses the challenge of mapping pixel-based legacy desktop user interface layouts to responsive grid layouts commonly used in Web Applications. AWSM:RM is described in detail in chapter 6.

4.3.3 AWSM Customer Impact Control Method

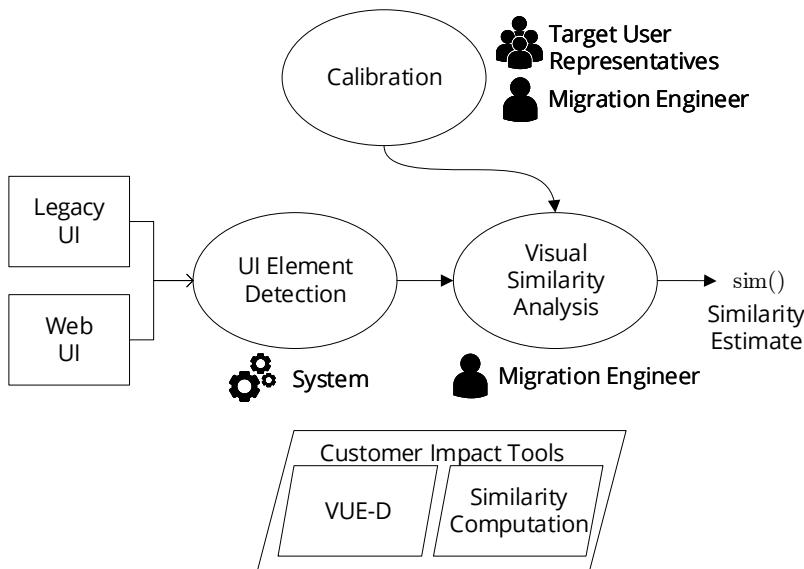


Figure 4.6: AWSM:CI Method Overview

The AWSM Customer Impact Control Method (AWSM:CI) shown in fig. 4.6 provides a solution for RO3 by specifying techniques to con-

trol the impact of Web Migration on the user bases of existing customers. AWSM:RM addresses the functional aspect of customer impact control as well as the non-functional constraint of limited resources and expertise of RO3. Based on the conceptual model of legacy user interfaces specified in section 4.6, AWSM:CI defines a visual similarity measure that allows determining and thus control the degree of visible changes introduced into the user interface through Web Migration. AWSM:CI solves RO3 through

- a model for analysis of visual similarity between legacy Desktop Application user interfaces and migrated Web-based user interfaces
- a process for calibrating the similarity model with a limited number of users
- a technique for automatically detecting user interface elements

Unlike existing Web Migration approaches, AWSM:CI explicitly addresses the concern of maintaining a similar look and feel between legacy and migrated system by defining a visual UI analysis model. This model allows to compute a measure of visual similarity based on automatically detectable visual aspects to resemble human perception of user interfaces.

To address the constraint of limited resources and Web Engineering expertise in RO3, AWSM:CI avoids extensive empirical data collection with large numbers of test subjects and user interface combinations. Instead, it specifies a process for calibrating the similarity model to the characteristics of the target user group with only a limited number of representatives and specifically designed test samples. In this way, the similarity measurement can be calibrated with limited effort in initial migration phases and then automatically applied for large numbers of user interfaces created throughout Web Migration.

The challenge lies in calculating object-dependent measures due to the heterogeneity of technologies and corresponding description formats for user interface elements. While existing automatic analysis methods for Web user interfaces rely on DOM⁶¹ analysis, the AWSM:CI similarity measure is applicable to both legacy and Web user interfaces (Heil, Bakaev, et al., 2016). To achieve this, AWSM:CI contributes to the novel field of research on visual analysis of user interfaces employing deep learning to automatically detect elements of the user interface, enabling calculation of derivative object-dependent measures. AWSM:CI is described in chapter 7.

4.4 Tools

The AWSM Toolsuite provides support tools for the methods of the AWM Methodology addressing the overarching goal and research objectives. It represents a Web-based Reverse Engineering and Web Migration management system to support management and Migration Engineer stakeholders, providing

- a decision support system for Web Migration strategy formation,
- a migration monitoring and management dashboard with software quality visualizations,
- a legacy knowledge repository with a query endpoint,
- a concept-assignment-based Reverse Engineering tool,
- a set of tools for crowd-based knowledge discovery,
- a business logic Transformation tool for Rapid Web Migration Prototyping,
- a tool for generation of RESTful APIs from legacy system screenshots,
- a tool for Transformation of legacy user interfaces to Web-based user interfaces,

⁶¹Document Object Model, cf. (W3C, 2015)

- a process guidance and orchestration tool for Rapid Web Migration Prototyping,
- a measurement tool for customer impact through Web Migration,
- a visual user interface element detector.

The following subsections provide an overview of these AWSM Tools. Several tools are grouped together by their relationship to each of the three research objectives and corresponding AWSM Methods.

4.4.1 AWSM Strategy Selection Decision Support System

The AWSM Strategy Selection Decision Support System (S2DCS) addresses the overarching goal OG by solving the problem of selecting a global migration strategy tailored to the situation of the ISV at the initiation of Web Migration. Within that Reengineering or Transformation strategy, the AWSM Methodology provides solutions addressing the identified gaps in current Web Migration approaches. The migration strategy determines the phases in fig. 4.3 and thus the outer context in which the AWSM Methods and Tools are applied. Selecting a suitable approach for the specific situation is a challenge for ISVs due to the wide range of approaches, as shown in section 3.2. The S2DCS tool is a decision support system based on the results of our systematic mapping study (Heil and Gaedke, 2017), which identified and evaluated 122 primary studies, comprising not only academic publications but also existing software tools. To address the constraints of OG, S2DCS represents a *faceted search* interface (Tunkelang, 2009) so that ISVs with limited resources and lack of Web Migration expertise are supported in formation of their migration strategy by being given easy access to a wide range of information that would have otherwise required extensive time, effort and expertise to accumulate.

Supporting non-technical Management stakeholders, as described in section 2.2.2, the faceted search is realized as a guided dialogue interaction, as shown in fig. 4.7. The S2DCS user answers a set of questions

The screenshot shows a web-based application titled "Entscheidungshilfesystem Softwaremigration S2DCS". At the top right, there is a "Logout" link. On the left, a welcome message "Willkommen Sebastian!" is displayed. Below the header, there is a navigation bar with three tabs: "Entscheidungsunterstützung" (highlighted in green), "Migrationsansätze", and "Migrationskriterien". A horizontal breadcrumb navigation bar below the tabs shows the current path: "Altsystem" > "Zielsystem" > "Migrationsphasen" > "Dynamische Kriterien" > "Ergebnismenge".
The main content area contains several sections:

- Abfrage zurücksetzen:** A link to reset the query.
- Geben Sie an, aus welchen Technologien sich das Altsystem zusammensetzt:** A question asking about the components of the legacy system.
 - UI:
 - TUI
 - Java-Swing
 - Java-AWT
 - HTML
 - GUI
 - Daten:
 - MySQL
 - Oracle Database
 - Microsoft SQL Server
 - PostgreSQL
 - AODB
 - Schnittstellen:
 - Shared Files
 - XML
 - DB
 - Sprache:
 - COBOL
 - Visual Basic
 - Java
 - C++
 - C
 - Fortran
 - Python
 - VBA
 - Architektur:
 - Monolithisch
 - Umgebung:
 - Mainframe
 - Keine Angabe zu dieser Frage
- Wie wichtig ist dieses Kriterium für Ihre Migrationssituation?** A question with a dropdown menu showing "Normal" and a "Weiter" button.

Figure 4.7: S2DCS Entry of Migration Situation View, First Step in Faceted Search: Legacy System Characteristics (German)

dynamically created from the available criteria and values in order to specify his Web Migration scenario, i.e. the as-is and to-be state and constraints. The answers form the search facets that define the query on the knowledge base. To construct the result set, candidate approaches are ranked according to their compliance with the query. The criteria

for selecting approaches are defined by target and source system characteristics (e.g. architectures, technologies), supported migration phases and dynamic criteria depending on the selection of the other criteria.

The user is shown a ranked list of approaches and tools matching his search criteria, as displayed in fig. C.1 and can review details and access information about authors, year of creation, a brief description, the main project or publication URL, complimentary linked resources such as reports and case studies, as well as available evidence of successful application, industrial relevance and tool support. The S2DCS is designed to allow the selection criteria, catalog data, and the dataset of its knowledge base to be extended through configuration to allow updating the Web Migration approaches.

4.4.2 AWSM Annotation Platform

The Annotation Platform implements knowledge discovery and management tools for the AWM Reverse Engineering Method solving RO1 and a support system for controlling Web Migration execution. To support the techniques of AWM:RE, Migration Engineer stakeholders can use the Annotation Platform to visualize software quality aspects of the legacy system code base and to conduct concept-assignment-based Reverse Engineering manually, using existing tools for automated Reverse Engineering or applying the novel AWM:RE crowdsourced Reverse Engineering approach. Figure 4.8 shows a screenshot of the dashboard view of the Annotation Platform, providing a starting page to access its functionality. The tools for crowd-based knowledge discovery implement AWM:RE techniques for the technical challenges of automatic creation and deployment of micro tasks for crowdworkers, controlling disclosure of legacy source code, and aggregation and quality control of crowdworker results. These functionalities represent solutions for the identification activity of RO1. The knowledge management activity of

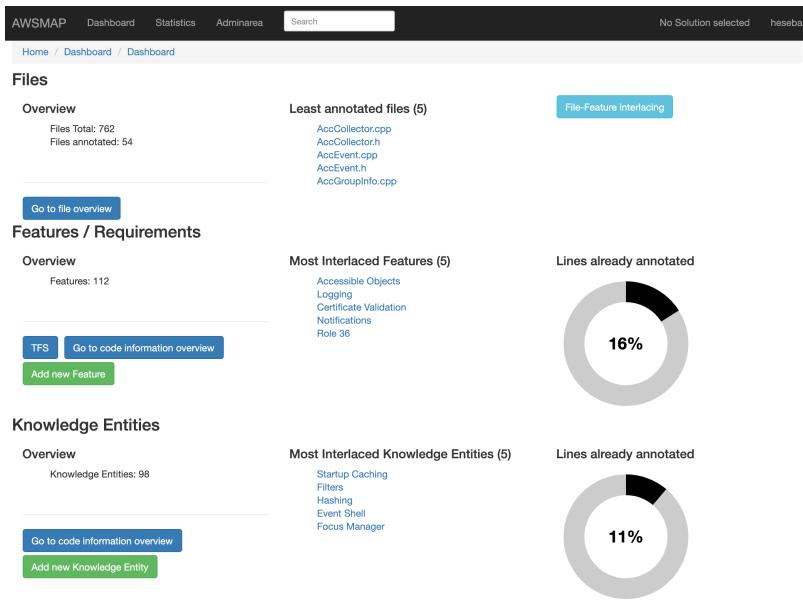


Figure 4.8: Annotation Platform Dashboard

RO1 is addressed by its Web-based navigable legacy knowledge repository for human stakeholders and its standards-based query endpoint for system actors and integration with other Web Migration tools. The Web Migration control part of the Annotation Platform provides a migration monitoring and management dashboard to allow management stakeholders to start and prioritize migration activities and monitor migration progress throughout the execution. To achieve good integration into ongoing development activities as per requirement C4 Agile, the AWSM Annotation Platform integrates with the existing tools landscape of ISVs. For management stakeholders, the Annotation Platform integrates with software project management platforms. This allows to feed migration-related tasks identified using the Annotation Platform into the

backlogs and planning of ongoing development. For Migration Engineer stakeholders, the Annotation Platform integrates with integrated development environments (Integrated Development Environments (IDEs)). This allows to conduct Reverse Engineering tasks for Web Migration during ongoing development activities, leveraging the existing mental models and program comprehension of developers applying changes to the legacy source code. The tools which constitute the AWSM Annotation Platform to address research objective RO1 are described in chapter 5 together with the AWSM:RE method which they support.

4.4.3 AWSM Rapid Web Migration Prototyping Tools

The ReWaMP/RWMPA, MockAPI and UI Transformer tools implement techniques of the AWSM Risk Management Method addressing RO2. They realize tool support for the AWSM's novel Rapid Web Migration Prototyping concept, allowing Migration Engineers to quickly create running Web-based prototypes of the Legacy System through automatic Transformation and process guidance.

ReWaMP leverages WebAssembly (WASM) to implement a reuse based Transformation of legacy business logic. This allows quick creation of Web Migration prototypes through reuse of parts of the legacy code base and making use of the expertise of existing staff in the legacy technology. The technical challenge addressed by ReWaMP lies in achieving a high degree of reuse of legacy code and limiting required adaptions in particular for dependency management. MockAPI supports rapid creation of RESTful API backends through annotation of screenshots of the legacy user interfaces. This allows automatic creation of a prototypical Web-based API for Web Migration prototypes without specific Web Engineering expertise, only requiring identification of the main domain entities represented in legacy user interfaces. The technical challenge addressed by MockAPI lies in Transformation from a light-weight domain-

specific annotation language into a running RESTful API endpoint. UI Transformer creates Web-based responsive grid layouts from pixel-based legacy user interfaces. This allows to automatically generate running Web-based frontend prototypes without Web Engineering expertise. The technical challenge addressed by UI Transformer lies in mapping between the two different layout paradigms of pixel-based and responsive grid while maintaining a sufficient level of visual similarity. The RWMPA tool guides Migration Engineers through the process of Rapid Web Migration Prototyping, providing descriptions and plugin-based extensible tool support for each necessary activity, orchestrating the tools and aggregating and forwarding the intermediate results. The technical challenge addressed by RWMPA lies in providing suitable guidance and tool support to enable Migration Engineers without extensive migration expertise to apply the AWSM:RM method and reduce the amount of manual interventions required. Figure 4.9 shows the view selection dialog of the RWMPA assistance system, allowing to choose the source material from existing legacy artifacts to start assisted prototyping.

Together, the ReWaMP/RWMPA, MockAPI and UI Transformer tools enable rapid creation of Web Migration prototypes for risk management Web Migration phases in support of the AWSM:RM method solving RO2. They represent the first concrete implementation of applying the Rapid Prototyping paradigm to the Web Migration domain for both frontend and backend. The AWSM Rapid Web Migration Prototyping Tools to address research objective RO2 are described in chapter 6 together with the AWSM:RM method which they support.

4.4.4 AWM Customer Impact Tools

The Customer Impact Tools realize techniques of the AWM Customer Impact Control Method addressing RO3. They provide tool support

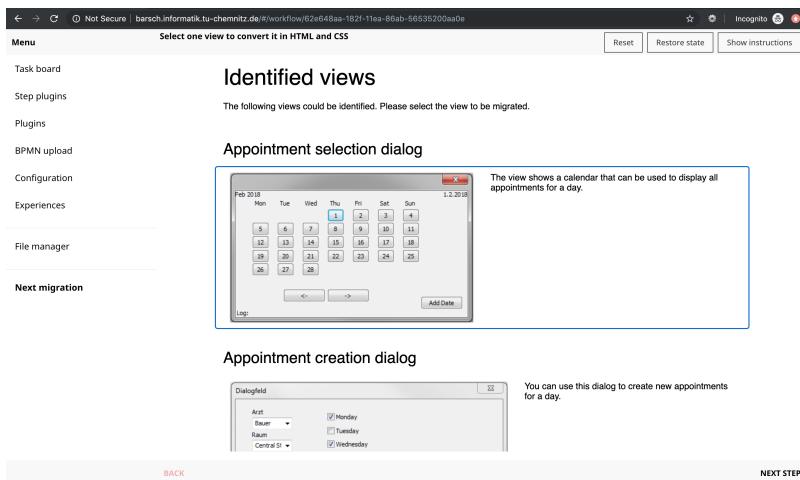


Figure 4.9: RWMPA View Selection

for quantifying the similarity between legacy non-Web user interfaces before, and Web-based user interfaces after migration to control the impact of visible layout changes through Web Migration.

To address the limited resources constraint of RO3, the Customer Impact Tools allow the computation of a similarity function. Instead of the high effort required for pair-wise empirical comparisons of user interfaces, this similarity function works as an approximation for perceived user interface similarity. It is based on a vector-space model of distances in several user interface similarity dimensions, each of which can be computed without manual intervention from objective measures on screenshots of user interfaces. This allows measuring the similarity between existing legacy user interfaces and newly created Web-based user interfaces using computable measures instead of time-consuming and expensive empirical experimentation, providing a basis for comparing alternatives and defining a step-wise transition

to the target layout with controlled impact in each step. Here, the technical challenge addressed by the VUE-D tool lies in calculating object-dependent measures such as variations in user interface element order across non-Web migration source and Web-based migration target system. VUE-D provides a solution for automatically detecting user interface elements in both legacy and Web-based user interfaces independent of the heterogeneous technologies and description formats. Figure 4.10 shows a Web user interface analyzed by VUE-D, combining shape detection (in red) with OCR (in green). Unlike existing approaches

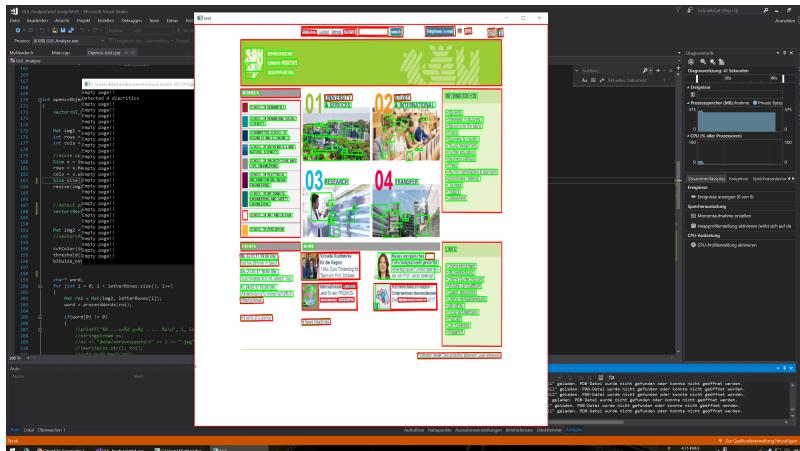


Figure 4.10: VUE-D UI Element Detection

which require specific textual representations of the user interface like DOM, it works on purely visual input data by applying computer vision techniques for locating areas of interest and machine learning techniques for classification of user interface element types. This allows to operate independent of specific legacy and Web-based user interface technologies and to restrict the input to visible characteristics that are

the basis for human perception. The AWSM Customer Impact Tools to address research objective RO3 are described in chapter 7 together with the AWSM:CI method which they support.

4.5 Principles

The AWSM Methodology and Toolsuite are built on four principles. These principles provide solutions to address the shortcomings in the state of the art of Web Migration identified in section 3.4 that are cross-cutting non-functional concerns for each of the three research objectives. Therefore, they drive design decisions of all methods and tools in the AWSM Methodology and Toolsuite. The following AWSM Principles formulate four top-level design decisions of AWSM and their rationale:

Principle P1 Open Web Standards

To address the problem of technology-specificity of Web Migration approaches limiting their applicability and inhibiting interoperability, this principle advocates using and extending open Web standards in the AWSM Methodology and Toolsuite. The use of standards is widely accepted in software engineering and Web Engineering. For Web Migration, using in particular W3C specifications and OMG migration standards allows interoperability with existing methods and tools and provides contributions that can easily be used and enhanced by future research (cf. OpenStand Principles⁶²). This especially applies to underlying data models, provided interfaces and technologies used in AWSM.

Principle P2 Methodology over Process

To address the problem of a wide range of existing Web Migration approaches with individual process models on the one hand, and their shortcomings with regard to Web Migration initiation for ISVs with limited resources and lack of Web Engineering expertise on the other

⁶²<https://open-stand.org/about-us/principles/> Retrieved: 6.12.2019

hand, AWSM aims at providing a *methodology* that addresses these shortcomings. This principle advocates an integration with existing Web Migration Reengineering and Transformation process models to prefer reuse over re-definition of a new Web Migration process model. Therefore, AWM Methods are designed for embedding within the context of other Web Migration approaches, as depicted in fig. 4.3.

Principle P3 Model-Driven Agnosticism

To address the limited applicability of Web Migration approaches due to the divide of the current Web Migration landscape on the use of Model-Driven Engineering (MDE), the AWM Methodology and Toolsuite are agnostic to Model-Driven adoption. This is required, since half of the approaches in section 3.2 employs MDE in some form, and half of the approaches does not. Likewise, MDE practices are employed in varying degrees in forward Web Engineering (Moreno et al., 2008). This principle advocates compatibility with Non-Model-Driven Web Migration approaches as well as with Web Migration approaches at different degrees of Model-Driven adoption. In particular Reverse Engineering, which forms the basis of Reengineering and Transformation, is laid out agnostic but adaptable to concrete Model-Driven or Non-Model-Driven methods in AWM to allow interoperability with a wide range of existing Web Migration approaches.

Principle P4 Rapid Prototyping Paradigm

To address the problem of a lack of demonstration of desirability for decision making in initial Web Migration phases, this principle advocates applying the Rapid Prototyping paradigm to the Web Migration domain. Prototyping is an established means of improving quality in forward software engineering (Wallmüller, 2001). The Rapid Prototyping paradigm (Gordon and Bieman, 1995) with its dedicated focus on quick and cheap creation of tangible means of communication of envisioned software for

all involved stakeholders (Alavi, 1984) is particularly common in the context of Agile Development (Abrahamsson et al., 2002) and Human-Centered Design (IDEO, 2015). Thus, the AWSM Methodology and Toolsuite favor the creation of prototypes as concrete, tangible means of communication to demonstrate desirability of a Web-based version of the migrated Legacy System over theoretical and general argumentation and narrowly focused technical feasibility studies.

4.6 Formalisms

The AWSM Methodology provides solutions for the three research objectives. These solutions are based on a common theoretical basis, the AWSM Formalisms specified in this section. The formalisms define basic *conceptual models* independent of their concrete implementation in the AWSM Toolsuite. AWSM defines three formalisms:

- Legacy System
- Source Code Knowledge Model
- Legacy User Interface

The Legacy System formalisms defines a Legacy System in terms of its *software artifacts* (Object Management Group, 2016a), i.e. of physical resources that are available as inputs for analysis and Transformation activities described in the methods of the AWSM methodology. The Source Code Knowledge Model formalism defines the knowledge contained in a Legacy System in terms of relevant *software assets* (Object Management Group, 2016a) that are represented through the artifacts but not necessarily existing in explicit forms. The Legacy User Interface formalism defines a uniform description of user interfaces and their conceptual components to support AWSM's focus on maintaining user interface consistency across legacy and target technologies.

As a design decision reasoned in principle P1, AWSM Formalisms extend OMG's Knowledge Discovery Meta-Model specification. On the one hand, this allows AWSM to reuse the semantics specified in KDM and avoid lengthy descriptions and re-definitions. On the other hand, KDM's technical focus on file-based knowledge representation is insufficient for the AWSM Methodology. Therefore, AWSM makes use of KDM's extension mechanism to effectively use only parts necessary (Object Management Group, 2016a) for the AWSM Methods and adds the required missing semantics. For brevity, when referring to KDM in the following description of formalisms, these references are to be considered as a reference to KDM version 1.4 of September 2016 (Object Management Group, 2016a).

4.6.1 Legacy System

The Legacy System formalism specifies the conceptual model of the central object of investigation of this thesis and is used to describe the source material of all AWSM Methods and Tools. KDM (Object Management Group, 2016a) provides a comprehensive *metamodel* for describing Legacy Systems. AWSM's conceptual model of a Legacy System is based on KDM concepts that provide the common foundation for the structure of Legacy Systems in the context of the AWSM Methodology.

Definition 4: Legacy System Formalism

In AWSM, a Legacy System \mathfrak{L} is a 6-tuple of a legacy codebase, a set of dependencies, a dependency matrix, a set of executables, a set of persistent data and a set of legacy user interfaces:

$$\mathfrak{L} = (B, \underline{Dep}, \underline{D}, E, D, U) \quad (4.1)$$

Legacy codebase B consists of a set of source code files (KDM: *SourceFile*), configuration files (KDM: *ConfigFile*) and other textual documents (KDM: *Document*).

Dependencies Dep represent external dependencies of the Legacy System (KDM: *LinkableFile*), in particular third-party libraries (KDM: *LibraryFile*).

Dependency matrix \underline{D} defined as $\underline{D} : \{1, \dots, |B|\} \times \{1, \dots, |Dep|\} \mapsto [0, 1]$ is a binary matrix with $\underline{D}_{i,j} = 1 \iff f_i \in B$ depends on $dep_j \in Dep$, else $\underline{D}_{i,j} = 0$.

Executables E are the software executables (KDM: *ExecutableFile*) of the Legacy System. In some cases, B or E can be $B = E = \emptyset$ (Binkley, 2007); however, as described in section 2.1.2, we assume availability of source code and executables in this thesis.

Persistent data D represents data stored or accessed by the Legacy System, such as databases or storage files (KDM: *FileResource*).

User interfaces U are the resources that form the user interface (KDM: *UIModel*) of the Legacy System, consisting of container resources (KDM: *UIResource*) like screens. The elements of U are not identical to their formal representation in B (e.g. XAML files, MFC resource files), but they are resulting from it and part of the executables in E from where their visual (i.e. pixel-based) representation can be retrieved. The legacy user interface is described in more detail in section 4.6.3.

4.6.2 Source Code Knowledge Model

To achieve RO1, the risk of losing knowledge through Web Migration needs to be addressed by extracting, capturing, and preserving valuable knowledge in the Legacy System. The Source Code Knowledge Model (SCKM) formalism is the part of the AWSM solution that allows to represent arbitrary knowledge in Legacy Systems.

The challenge addressed by the SCKM is the implicit nature of knowledge in legacy code bases: similar to *tacit knowledge* in organizations which is not expressed explicitly but guides human behavior (Nonaka, 2008), the knowledge in Legacy Systems is not explicitly documented but governs how they operate. Codification is required to make it explicit, which is achieved through *reverse engineering*. The knowledge reverse-engineered from the legacy source code through redocumentation and design recovery needs to be represented at different levels of abstraction and stored in a *knowledge base* to feed into subsequent Reengineering or Transformation.

OMG's KDM is insufficient for describing implicit knowledge. It is focused on redocumentation knowledge, i.e. equivalent representations of knowledge within the same abstraction level, the implementation level, describing the structure of the legacy codebase in terms of modules, classes, methods, files. KDM assumes knowledge to reside in dedicated files – data in data files, configuration in configuration files, etc. – whereas knowledge often appears mixed, due to the missing separation of concerns in Legacy Systems, within the source code.

Therefore, the SCKM allows the representation of design recovery knowledge, i.e. knowledge on higher levels of abstraction, free of the assumption of explicit representation while maintaining the connection to its occurrence in structural parts of the source code. The SCKM formalism specifies two conceptual models:

- a unified knowledge model independent of explicit physical representations,
- a model for linking knowledge to its occurrences within artifacts of the legacy code.

Unified knowledge model

In AWSM, an instance of knowledge k in the Legacy System \mathcal{L} is a tuple

$$k = (t, r) \quad (4.2)$$

of type $t \in T$ and a representation r . AWSM considers two basic varieties of knowledge in legacy source code:

- *Features* describe functionality of the Legacy System (What) and can be represented as user stories, scenarios, use cases etc.
- *Domain knowledge* is the knowledge supporting the implementation of features, describing parts of the problem and solution domain of the Legacy System (How).

Domain knowledge is divided into problem and solution domain knowledge (Marcus et al., 2004). In SCKM, problem domain knowledge comprises business processes and business rules. Solution domain knowledge comprises presentation, persistence, algorithms, configuration, deployment, and explanatory⁶³. The SCKM formalism exceeds the classical three-tier-architecture perspective considered in program decomposition providing a more detailed distinction of domain knowledge in the legacy source code. All SCKM knowledge types are shown as part of the full SCKM Formalism UML Model in fig. C.2.

These knowledge types can be represented by a variety of representations like UML class diagrams, BPMN diagrams, flow charts, SBVR⁶⁴ rules, other forms of models, informal or semi-formal natural

⁶³natural-language knowledge embedded in the code as comments

⁶⁴<https://www.omg.org/spec/SBVR/> Retrieved: 6.12.2019

language texts etc., depending on the intended use and can even have no representation ($r = 0$) since determination of the type of a particular piece of \mathcal{L} is knowledge.

Linking knowledge to legacy code

As knowledge is implicitly represented and distributed (e.g. partial classes) in the source code, it is crucial to codify the connection to its occurrences in the source, in particular considering the importance of decomposability and traceability for migration. AWSM models this connection as *annotation* $a \in K^* \times L^*$.

Definition 5: SCKM Annotation Formalism

An SCKM annotation a links a knowledge instance k to its location l in arbitrary parts within the legacy codebase B without changing any part of \mathcal{L} :

$$a = (k, l) \quad (4.3)$$

The annotation a represents the *intension* and *extension* (K. Chen and Václav Rajlich, 2010) of knowledge: it associates a piece of knowledge k with its location l in the legacy code base. While KDM relates knowledge to physical artifacts like files and structural elements like classes, the SCKM considers knowledge to be an independent asset that can be related to arbitrary parts within the elements of B . Figure 4.11 shows the SCKM Annotation Formalism.

Based on this model, we define the legacy code knowledge base required to address RO1 as follows:

Definition 6: Legacy Code Knowledge Base Formalism

A legacy code knowledge base \mathbb{K}_B consists of a set of knowledge instance K and a set of annotations A that result

from the application of a function $re : B^* \mapsto A^*$ that maps a legacy codebase B onto the set of annotations $A = \{a_1, a_2, \dots, a_n\}$: $re(B) = A$ that identify the set of knowledge instances $K \in K^*$ and their occurrences $L \in L^*$ in the source code:

$$\mathbb{K}_B = (K, A) \quad (4.4)$$

Figure C.2 shows the Knowledge Base as top-level element of the full SCKM Formalism.

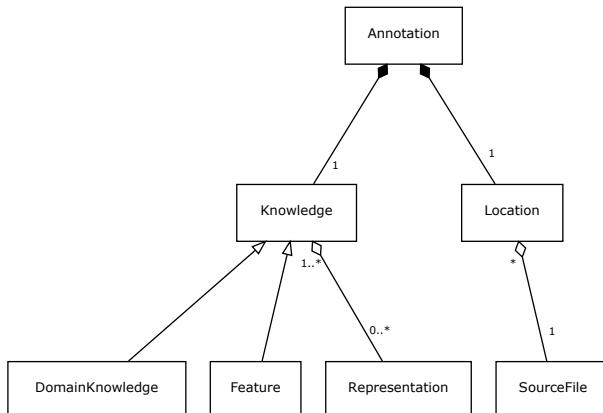


Figure 4.11: SCKM Annotation

4.6.3 Legacy User Interface

To achieve RO2 and RO3, AWSM addresses the lack of user interface migration, and user interaction reuse of existing Web Migration approaches, as identified in G3. This requires a formalism as common conceptual model for describing the legacy user interface and its migrated derivatives. User interfaces are defined in terms of Task, Behavior,

Thesaurus, Layout, and Material (Bakaev, Khvorostov, et al., 2017a). The AWSM conceptual model of legacy user interfaces focuses on the layout aspect, because of three reasons: 1. the layout necessarily undergoes the most visible change through Web Migration, whereas changes in e.g. Task or Thesaurus can be avoided, 2. it is the basic design artifact of the new Web UI, influencing other aspects like Behavior and 3. it has the most degrees of freedom during re-design, in contrast to e.g. Material which is governed by the platform.

AWSM defines a concrete legacy user interface $u \in U$ from the set of resources that form the user interface of the Legacy System as

$$u = (C_u, w_v, h_v), \text{ where } C_u \subseteq C_{\mathcal{L}} \text{ and } w_v, h_v \in \mathbb{N}_0 \quad (4.5)$$

It consists of C_u , the subset of UI controls of that particular user interface, and w_v and h_v , the viewport width and height of the user interface. In AWSM, all lengths like w_v and h_v and positions are represented as non-negative integer numbers in the unit of pixel. The physical size of a pixel may vary depending on the hardware. $C_{\mathcal{L}} \subset T_c \times \mathbb{N}_0^4 \times C_{\mathcal{L}}$ is the set of all UI controls of the Legacy System \mathcal{L} . A UI control $c \in C_{\mathcal{L}}$ is defined by its type $t(c)$, the rectangular bounding box $b(c)$ occupied by c within the surrounding container and its parent UI container $p(c)$.

UI controls can be nested inside other UI controls (e.g. a button inside a form inside a window). The nesting is represented via references to the parent: $p : C_{\mathcal{L}} \mapsto (C_{\mathcal{L}} \setminus \{c\}) \cup \{0\}$. If a UI control contains at least one other UI control, it is referred to as UI container. A UI control c cannot

be contained within itself. For root-level UI controls $p(c)$ is $p(c) = 0$. The resulting hierarchy of UI controls forms a tree-based representation of the UI, which is commonly used for further UI analysis.

The bounding box $b : C_{\mathfrak{L}} \mapsto \mathbb{N}_0^4$ is the minimal rectangular area completely enclosing c defined in terms of the horizontal coordinate x and the vertical coordinate y of the upper left corner of the rectangle and its width w and height h , in pixel respectively:

$$b(c) = (x, y, w, h) \in \mathbb{N}_0^4 \quad (4.6)$$

The coordinate origin is defined as position $x = 0, y = 0$ in the upper left corner of the root-level UI container. If a UI control c has a parent UI container $p(c) \neq 0$ then the coordinates of its bounding box $b(c) = (x, y, w, h)$ are relative to $p(c)$, i.e. the position $x = 0, y = 0$ does not refer to the top left corner of the root-level UI container, but to the top left corner of $p(c)$.

4.7 Summary

This chapter introduced the AWSM approach for Web Migration initiation by SME-sized ISV with legacy, non-Web, Desktop Application products, and a large existing user base. The basic idea of AWSM is to provide methods targeting the shortcomings of existing Web Migration approaches in addressing doubts about feasibility and desirability within a suitable migration strategy selection of which is supported by a guided Web Migration information system. The approach consists of methods, tools, principles, and formalisms to facilitate Web Migration initiation. The three methods specify several techniques for knowledge rediscovery, risk management, and customer impact control. The tools

of the AWSM Toolsuite provide implementations of the proposed techniques, facilitating application of the AWSM Methods and focusing on integration with migration approaches, ongoing development activities and environments. AWM's top-level design decisions reflected in four core principles promote open Web standards, integration in existing comprehensive approaches, model-driven agnosticism, and application of the Rapid Prototyping paradigm to Web Migration. The formalisms provide a common basis for the AWM Methods, their techniques and for the tools, and they include KDM-based conceptual models of Legacy Systems and knowledge in Legacy Systems, and of a model of legacy user interfaces. The following three chapters provide a detailed view of the methods and tools of the AWM Methodology and Toolsuite.

AWSM Reverse Engineering Method

5

This chapter addresses research objective RO1:

To enable identification and management of existing knowledge in legacy source code with limited resources and lack of Web Engineering expertise.

First, the current situation is briefly analyzed to identify requirements and review related work. Three research questions are derived from RO1, inspired by the analysis results. To address the research questions, three sections outline the three main contributions towards RO1 in the context of the AWM:RE method: a technique for Knowledge Rediscovery is described, the architecture and functionality of the Annotation Platform as support tool for AWM:RE is presented, and the novel concept of Crowdsourced Reverse Engineering is introduced as realization of the Knowledge Rediscovery technique with Crowd-based actors. Finally, the evaluation of the method, including detailed experimentation results, is reported.

5.1 Analysis

The following analysis briefly summarizes the situation of undocumented valuable knowledge in Legacy Systems and the challenges of recovering knowledge manually or through automation. Requirements are derived from this analysis and an overview of related work on the two key paradigms used for AWM:RE is given.

5.1.1 Situation and Challenges

Legacy Systems represent valuable problem and solution domain knowledge about business processes, rules, etc. (Aversano et al., 2001; Harry M. Sneed et al., 2010b; Wagner, 2014; Bodhuin, Guardabascio, et al., 2002; Ulrich, 2011) often resulting from years of requirements elicitation and experience. Thus, losing this valuable knowledge is a significant risk (Khadka, Batlajery, et al., 2014). Knowledge management is important for Web Migration (Razavian, Nguyen, et al., 2010; Razavian, 2013). Relevant and non-relevant parts of business logic must be distinguished (Ulrich, 2011). Knowing where knowledge is located in the code is essential because decomposability is a vital pre-requisite for migration (Lucia, Francese, Scanniello, and Tortora, 2008; Canfora, Cimitile, et al., 2000; Brodie and Stonebraker, 1995). It is not possible to migrate without understanding the domain (Masak, 2006).

The problem is the lack of legacy artifacts apart from the legacy source code, i.e. missing or poor documentation, models, requirements, etc. (Harry M. Sneed et al., 2010b; Warren, 2012; Batlajery et al., 2014; Lucia, Francese, Scanniello, and Tortora, 2008), inhibiting a systematic way to deal with changes (Sosa-Sanchez, Clemente, Sanchez-Cabrera, et al., 2014). The legacy source is often “the only source of domain knowledge” (Bodhuin, Guardabascio, et al., 2002) due to original developers not being available anymore. However, this knowledge is not explicitly documented. Therefore, extracting business rules and knowledge is among the main obstacles for modernization (Batlajery et al., 2014).

To achieve proper knowledge management, this extraction requires *codification* i.e. turning the knowledge implicitly represented by the source code explicit (Hansen et al., 1999). Re-gaining it is essential for

successful migration and requires *Reverse Engineering* techniques/tools (Harry M. Sneed et al., 2010b) when the legacy source is the only available representation of knowledge (IEEE Computer Society, 1998).

Ad-hoc migration approaches adopted in the industry do not consider systematic Reverse Engineering, focus on Forward Engineering and knowledge “remains tacit in stakeholders minds” (Razavian and Lago, 2012). However, the original stakeholders are not necessarily available for old Legacy Systems due to *erosion of soft knowledge* (Khadka, Batlajery, et al., 2014) and tacit knowledge requires “person-to-person knowledge transfer” (Razavian and Lago, 2012), which can be disadvantageous due to organizational resistance (Khadka, Batlajery, et al., 2014) hindering knowledge sharing. Existing re-documentation approaches focus on solution domain knowledge. OMG Standards KDM and ASTM focus mainly on the syntactic level, poorly addressing business process and rules recovery (Mohagheghi and Sæther, 2011).

Manual knowledge discovery is time-consuming and difficult (Khadka, Batlajery, et al., 2014; Batlajery et al., 2014) and cannot be easily integrated into ISV’s daily software development and maintenance. On the other hand, Reverse Engineering is very difficult to automate for large and complex information systems, potentially leading to low precision or recall (Canfora and Penta, 2007). This is in line with results from our own experiments employing machine learning technologies for knowledge discovery, as indicated in section 5.5. Thus, the challenge is to specify a suitable Reverse Engineering method for knowledge discovery that reduces effort compared to manual discovery, can be integrated with daily development, and provides better results than automated approaches.

5.1.2 Requirements

The following requirements specify RO1, based on the analysis presented above and the AWSM principles.

Efficiency Knowledge rediscovery should be supported to require fewer resources compared to manual rediscovery.

Effectiveness Knowledge rediscovery results quality should be similar to manual rediscovery results.

Expertise Knowledge rediscovery should be feasible with available expertise of the ISV's staff.

Integration Knowledge rediscovery should be integrated with ongoing development and maintenance activities of the ISV.

Knowledge Management Knowledge should be represented and made available for subsequent Reengineering or Transformation activities agnostic of specific model-driven or non-model-driven methods through open Web standards.

5.1.3 Related Work

This section introduces Concept Assignment and Crowdsourcing as key paradigms of AWSM:RE. It provides a minimal overview of related work in these fields and establishes the semantics of the key terminology used in the following description of AWSM:RE.

Concept Assignment is a Reverse Engineering paradigm that forms the basis of the AWSM Reverse Engineering method. It aims at deriving the “human-oriented expression of computational intent” (Biggerstaff et al., 1994) from source code. The *Concept Assignment Problem* is defined as “discovering these human-oriented concepts and assigning them to their realizations within a specific program or its context” (Biggerstaff

et al., 1994). This comprises a two-step process: 1) to identify which “entities and relations . . . are really important” and 2) to “assign them to the known” or newly discovered “domain concepts and relations” (Biggerstaff et al., 1993). Concepts are defined as follows:

Definition 7: Concept (V. Rajlich and Wilde, 2002)

Concepts are units of human knowledge that can be processed by the human mind (short-term memory) in one instance.

Concepts represent specific problem or solution domain knowledge (Marcus et al., 2004). Concepts are formalized as a triple of name, intension (meaning), extension (related parts of the Legacy System) (K. Chen and Václav Rajlich, 2010). In this thesis, we leverage the Concept formalism to describe knowledge in legacy source code as it allows to relate arbitrary problem and solution domain knowledge to its location in the legacy code base, enabling *traceability*. Thus, the AWSM Reverse Engineering method is designed as Concept Assignment method.

Concept Assignment is hard to automate and will never be completely automated since it relies heavily on *a priori* knowledge (Biggerstaff et al., 1993). Therefore, Concept Assignment is often performed manually. In spite of extensive Reverse Engineering research, knowledge extraction is still a major obstacle for software industry (cf. section 2.1 and (Khadka, Batlajery, et al., 2014; Batlajery et al., 2014)). Therefore, AWSM:RE considers the application of Crowdsourcing as an alternative to automation of Concept Assignment.

Crowdsourcing is a work organization paradigm representing an “online, distributed problem-solving and production model” (Brabham, 2008) to be applied to Concept Assignment in order to address the limited resources and expertise constraint of RO1. It is defined as:

Definition 8: Crowdsourcing (Howe, 2006)

The act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call.

Crowdsourcing in Software Engineering is an active field of research (K. Mao et al., 2017; Latoza and Hoek, 2016), aiming at a “new system-development model” (Kazman and H.-M. Chen, 2009). Latoza and Hoek (2016) identifies eight dimensions of *Crowdsourcing in software engineering* that allow defining specific Crowdsourcing models such as *peer production*, *competition* and *microtasking*, and these dimensions are used to identify a suitable approach for crowdsourcing Concept Assignment in AWSM:RE. Crowdsourcing was applied to software creation (Satzger et al., 2014; Nebeling, Leone, et al., 2012), UI design (Weidema et al., 2016; Nebeling, Speicher, et al., 2013), and testing (Stol and Fitzgerald, 2014), benefiting from parallel execution in terms of quality and speed. Quality control is a crucial challenge (Daniel et al., 2018; Allahbakhsh et al., 2013). There is an increasing interest in Crowdsourcing from the (forward) software engineering community (K. Mao et al., 2017).

Crowdsourcing in Reverse Engineering, in contrast, has not seen much interest, apart from malware classification in software binaries through a passive approach of repurposed Crowdsourcing (Saxe et al., 2014). None of the approaches in section 3.2 considers the application of Crowdsourcing, neither bespoke nor repurposed. However, Reverse Engineering can benefit from increased efficiency due to parallel work, access to a wider workforce of specialists (Latoza and Hoek, 2016), and reduced effort due to outsourcing and cost reduction (Stol and Fitzgerald, 2014). Since these potential benefits are well suited to address the requirements in section 5.1.2, section 5.5 shows simi-

larity of Concept Assignment to the established microtasking model in eight dimensions and describes the application of Crowdsourcing in Reverse Engineering by reformulating the problem of Concept Assignment as classification problem.

5.2 Research Questions

The analysis presented above raises three research questions:

AWSM:RE Research Question RQ1: How to identify problem and solution domain knowledge in Legacy Systems with limited resources and lack of Web Engineering expertise?

AWSM:RE Research Question RQ2: How to transfer the Crowdsourcing paradigm into a Reverse Engineering context?

AWSM:RE Research Question RQ3: How to manage problem and solution domain knowledge in Legacy Systems to make it usable for Web Migration processes based on Reengineering or Transformation at different degrees of model-driven adoption?

To design and evolve a solution for research objective RO1, this chapter addresses these three research questions in the following sections.

5.3 Knowledge Rediscovery

AWSM:RE Knowledge Rediscovery contributes to the identification of knowledge implicitly represented by the Legacy System specified in research objective RO1. It provides an answer to AWSM:RE RQ1, realizing the identification of knowledge by defining a Concept Assignment-based Reverse Engineering technique to identify valuable knowledge

in the Legacy System \mathcal{L} , which supports the identification of relevant components, elicitation of architectural knowledge and improving decomposability. The conceptual basis of the technique are the Legacy System and SCKM formalisms introduced in section 4.6. Concept Assignment takes the codebase B of Legacy System \mathcal{L} as input and is an instance of function $re : B^* \mapsto A^*$ described in the SCKM, which maps B onto a set of annotations A that constitute the legacy code knowledge base \mathbb{K}_B . The following subsections define the Knowledge Rediscovery in terms of three processes. The AWSM:RE Knowledge Rediscovery technique consists of three processes, each realizing a use case involving different roles, as shown in fig. 5.1: the Setup Process for initiation of the technique and environment, the Concept Assignment Process for knowledge extraction, and the Management and Usage Process for further use of the extracted knowledge.

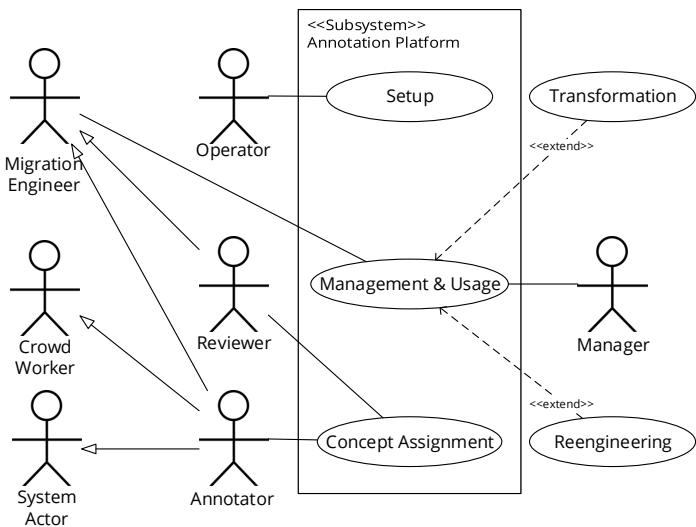


Figure 5.1: AWSM:RE Use Cases

The Annotator is the role realizing function re . This role can be assumed by a human actor i.e. a Migration Engineer of the ISV (cf. table B.1), by a system actor, i.e. a static or dynamic analysis approach as described in section 5.1.3 or by the crowd. The Annotation Platform is the subsystem of the AWSM Toolchain that enables the Annotator to create annotations by representing the codebase B and storing the created annotations in the Knowledge Base \mathbb{K}_B . The Reviewer is an optional human role required to verify the results of system or crowd Annotators. Reviewers are Migration Engineers of the ISV. Manager is a human role that oversees the extraction process (cf. table B.1) and manages the Web Migration using the information in the Annotation Platform. The Operator is a human role responsible for the operation of the toolchain and a part of the ISV's IT Operations or DevOps staff.

The processes corresponding to the three central use cases are described in the following. While the Setup Process needs to be performed once, to initiate the environment, the Concept Assignment and Management and Usage can be performed multiple times, in parallel and independent of each other.

5.3.1 Setup Process

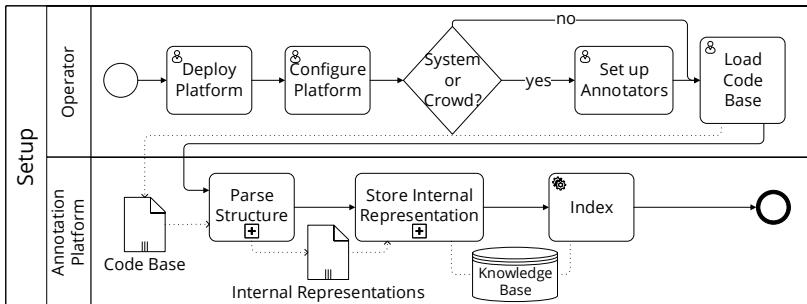


Figure 5.2: Setup Process

The Setup Process shown in fig. 5.2 needs to be executed once, to prepare the Environment for the Knowledge Rediscovery technique. The Operator sets up the Annotation Platform by deploying it and providing configuration for database connection, authentication, tool integrations etc. If realized as system actors, Annotators are set up. The Operator loads the codebase B in the Annotation platform, which, by parsing the structure of B , identifies the Legacy System's textual artifacts from B and D according to the SCKM, creates internal representations, and performs indexing on the contents.

As shown in fig. 5.3, the Annotator requests an artifact f via the suitable interface of the Annotation Platform, i.e. the user interface for human actors or the API for system actors. The user interface provides navigation of the information space of textual artifacts in B and D according to the parsed structure. It supports filtering based on the internal organization of the code, e.g. for different software packages, assemblies etc. It also supports searching based on full-text indexation of the textual contents. When selecting an artifact, the user interface displays the artifact with syntax highlighting and already existing assigned Concepts in the Knowledge Base to support the Annotator's understanding of the code. The Annotator views the artifact and performs the Reverse Engineering activities of *extraction* and *abstraction* realized as Concept Assignment: he identifies a relevant segment $s \in f$ of the artifact and assigns an existing concept (cf. Definition 7) to it or creates a new one. System actor Annotators interact with the Annotation platform through its API and automatically perform extraction and abstraction as defined by the static or dynamic analysis method they implement. The details of Crowd-based Annotators are described in section 5.5. Optionally, for system actors and Crowd-based Annotators, the Reviewer checks and confirms, declines, or corrects the Concept Assignments. The annotation process should produce at least one Concept Assignment as

code annotation a , regardless of the actor. The Knowledge Base adds the new information to its storage, which consists of a location l and a knowledge instance k as defined in the SCKM.

5.3.2 Concept Assignment Process

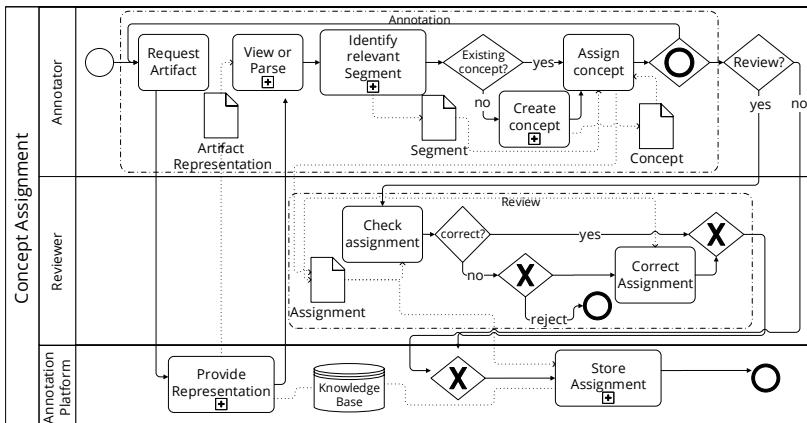


Figure 5.3: Concept Assignment Process

As outlined in requirement C4 Agile, integration of Web Migration into ongoing development is crucial to reduce additional efforts and address the lack of resources of ISVs. The Concept Assignment process can be applied in small increments whenever a part of the Legacy System needs to be changed for maintenance (Heil and Gaedke, 2016). Embedding Concept Assignment with ongoing development activities provides several benefits: The effort is lower compared to performing Concept Assignment as separate activity because the cognition effort of program comprehension for creating a *mental representation* of the source code (Pennington, 1987) in Forward Engineering is reused. Understanding of a specific segment of code gained for maintenance and development is codified during or right after the Forward Engineering

activity. Vice versa, availability of knowledge previously extracted can support the comprehension for Forward Engineering. The resulting continuity of Reverse Engineering improves the consistency of code and the knowledge base. To support Migration Engineer stakeholders performing this Concept Assignment embedded into ongoing development, integration into editors/IDEs is required as provided by the Annotation Platform described in section 5.4.2.

5.3.3 Management and Usage Process

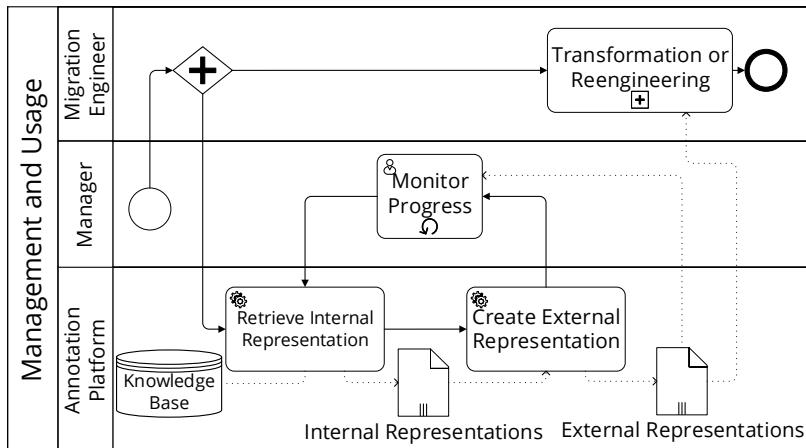


Figure 5.4: Management and Usage Process

As shown in fig. 5.4, the Platform supports the Manager role to monitor the Concept Assignment by providing a suitable external representation – a user interface with statistics and visualizations of the progress and to understand the as-is business process and rules portfolio of the legacy application. It also supports subsequent Transformation or Reengineering methods at different degrees of human involvement and model-driven adoption (cf. P2 and P3) by representing the Knowledge Base in a human-understandable form via the user

interface and an API based on open Web standards (cf. P1). This requires a generic, extensible, and queryable knowledge representation model, which is described in section 5.4.3.

Integration with ongoing development activities (requirement C4) of the Management and Usage process is achieved by associating a set of annotations $A = \{a_1, a_2, \dots, a_n\}$ with forward development process artifacts, i.e. a set of stories $S = \{s_1, s_2, \dots, s_n\}$ in a backlog. In this way, they form a *migration package* that feeds a *migration backlog* (Heil and Gaedke, 2016) from which stories can be integrated into the planning of iterations, e.g. into sprint backlogs, achieving integration with agile planning at artifacts level. To support management stakeholders, integration into software project management platforms is required as described as part of the Annotation platform in Section 5.4.2.

5.4 Annotation Platform

The AWSM Annotation Platform (AWSMAP) (Heil and Gaedke, 2016) Reverse Engineering tool contributes to the identification and management of knowledge specified in research objective RO1 by supporting the AWSM:RE Knowledge Rediscovery technique and managing the extracted knowledge. It answers AWSM:RE RQ3 for both system and human users and enables Concept Assignment for the three types of Annotators introduced above and implements the Knowledge Base \mathbb{K}_B in Definition 6. Figure 5.5 shows its architecture, which was prototypically implemented using C# .NET 4.5 MVC 5.2.

AWSMAP provides two interfaces over the Knowledge Base: a user interface for supporting human users in the different roles and processes defined in section 5.3, and an API for supporting system users in annotation and subsequent Transformation or Reengineering. The

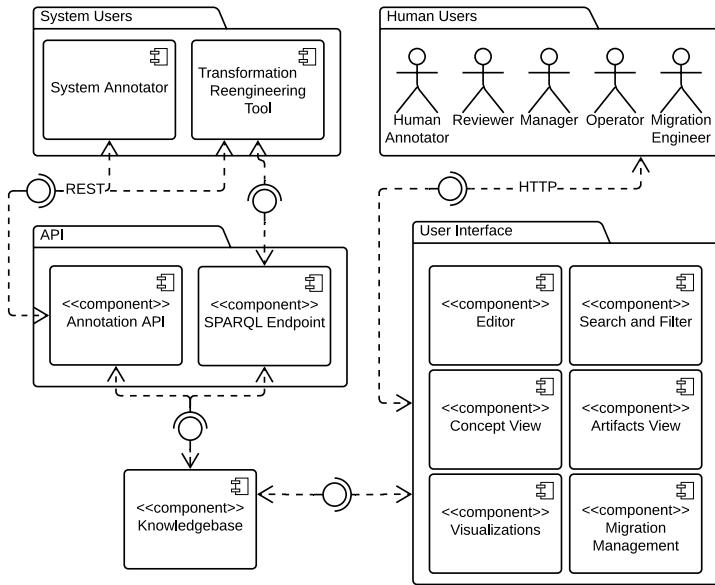


Figure 5.5: AWSMAP Architecture

functionality provided to human users via the user interface is described in section 5.4.1, its integration with the development environment of ISVs in section 5.4.2, and the internal and external knowledge representations for system users via the API is described in section 5.4.3.

5.4.1 User Functionality

The Web-based user interface supports Migration Engineers to perform Concept Assignment and manage domain knowledge, answering AWSM:RE RQ3 for human users. It displays the structure of the legacy codebase B , allowing to filter by KDM Module, i.e. Visual Studio Solution, and provides a view for each artifact $f \in B$.

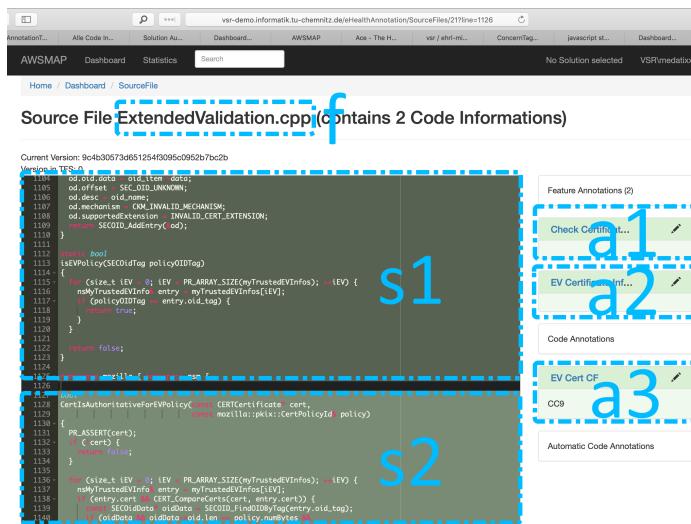


Figure 5.6: Annotation Platform Editor Screenshot

AWSMAP displays artifacts with syntax highlighting ace.js⁶⁵, as shown in fig. 5.6. The content of the artifact *f*, e.g. the source code, is displayed in the center, existing annotations *a*₁, *a*₂, *a*₃) are displayed as overlays and are listed on the right side of the view. By clicking on an annotation in the list, the content view scrolls to the corresponding location in the artifact. New annotations can be created by selecting a segment of code *s* in the content view, like *s*₁, *s*₂ shown in the figure, and selecting an existing or creating a new Concept. The representation of the assigned Concept can be viewed by following the link in the annotation. The Concept View that is opened is shown in fig. 5.7. It shows a knowledge instance *k* = (*t*, *r*), displaying the Concept's name, type *t*, and description. Additionally, attachments can be uploaded, e.g. UML or BPMN diagrams or other artifacts specific for subsequent processing (cf. P2 and

⁶⁵<https://ace.c9.io/> Retrieved: 6.12.2019

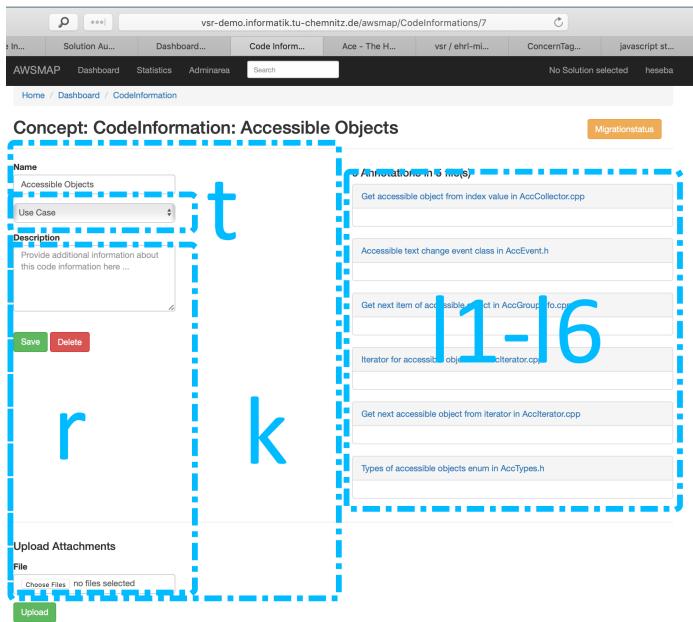
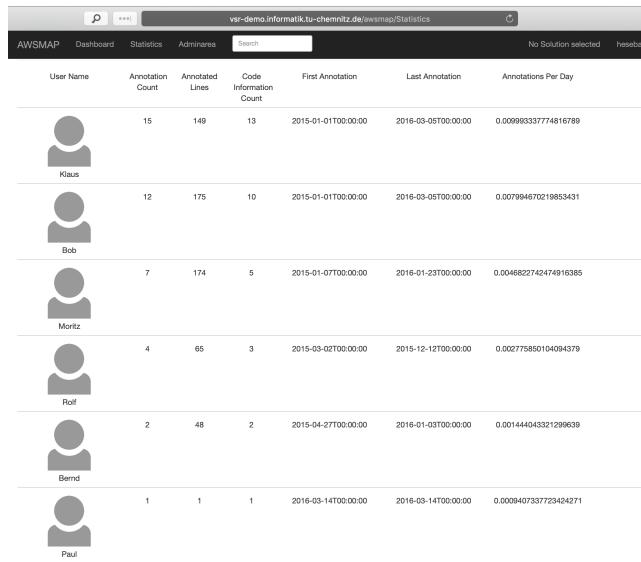


Figure 5.7: Annotation Platform Concept View

P3). Locations l_1-l_6 of the Concept in B are listed on the right side with the links leading to the corresponding segment s in artifact f . One of the core benefits of AWSMAP is referencability of all artifacts and knowledge through individual URLs, as seen in the address bar in fig. 5.7. This implements hypertext-based navigation of the legacy information hyperspace and fosters integration by making knowledge linkable in internal email or chat communications, intranet wikis, blogs, etc.

For Management stakeholders, AWSMAP provides a UI with different monitoring statistics and visualizations of Concept structure. Figure 5.8 shows Annotator statistics, providing an overview of the annotation progress and individual Annotator performance.



The screenshot shows a web-based annotation platform interface titled "AWSMAP Statistics". The top navigation bar includes links for "AWSMAP", "Dashboard", "Statistics", "Administrators", and a search bar. Below the navigation is a table with the following columns: User Name, Annotation Count, Annotated Lines, Code Information Count, First Annotation, Last Annotation, and Annotations Per Day.

User Name	Annotation Count	Annotated Lines	Code Information Count	First Annotation	Last Annotation	Annotations Per Day
Klaus	15	149	13	2015-01-01T00:00:00	2016-03-05T00:00:00	0.0099337774816789
Bob	12	175	10	2015-01-01T00:00:00	2016-03-05T00:00:00	0.007994670219853431
Moritz	7	174	5	2015-01-07T00:00:00	2016-01-23T00:00:00	0.004682274274916385
Rolf	4	65	3	2015-03-02T00:00:00	2015-12-12T00:00:00	0.002775850104094379
Bernd	2	48	2	2015-04-27T00:00:00	2016-01-03T00:00:00	0.00144043321299639
Paul	1	1	1	2016-03-14T00:00:00	2016-03-14T00:00:00	0.0009407337723424271

Figure 5.8: Annotation Platform Statistics

Based on the existing annotations, AWSMAP provides software quality visualizations allowing to assess structural problems of the legacy code that need special consideration before or during Web Migration. The focus of the visualizations is on interlacing between artifacts, features, and knowledge instances in different combinations. Figure 5.9 shows a Sankey Diagram of artifact-feature interlacing. artifacts f are displayed on the left side, features $k = (t, r)$ with $t \in \text{userstory, usecase, ...}$ on the right side. The connections between them represent occurrence of features in the artifacts, the width of the connection increases with the number of corresponding annotations. The more chaotic, i.e. the more entangled the connecting lines are, the higher the interlacing, indicating a lack of separation of concerns. Individual connections can be highlighted,

and the artifact and feature names are links leading to the corresponding location in the source code for further inspection. Other visualizations implemented in AWSMAP are Force Graphs and Chord diagrams.

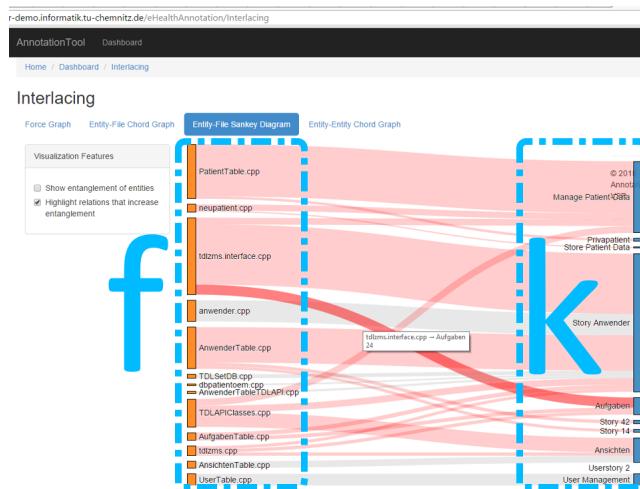


Figure 5.9: Annotation Platform Visualizations

5.4.2 Integration in ISV environments

This section addresses integration of AWSMAP into core parts of the environment of ISVs: Integrated Development Environments, Version Control Systems, Project Management Platforms, and Authentication and Authorization Systems. Figure 5.10 provides an overview of the integration architecture of AWSMAP. The following paragraphs provide one example for each of these integrations, which was prototypically implemented in AWSMAP for the specific environment of the scenario stakeholder, as described in section 2.1.

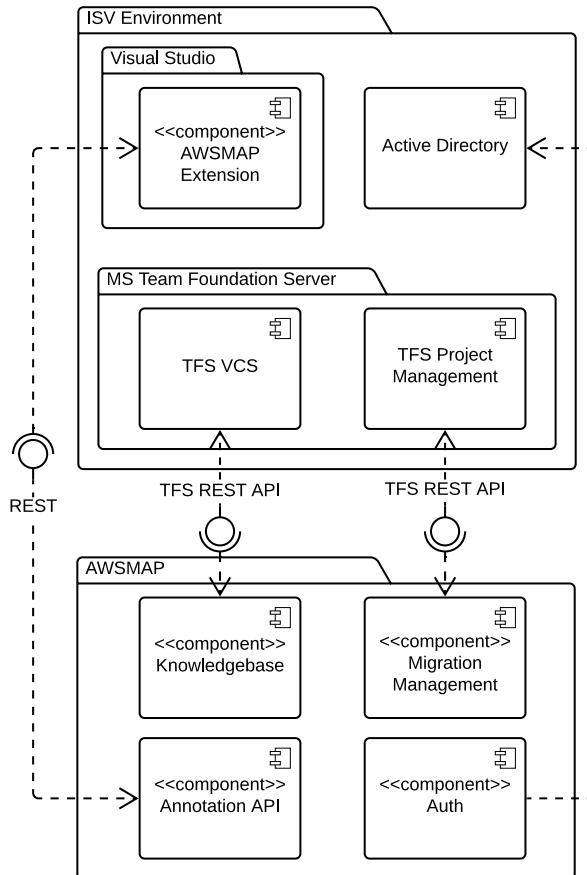


Figure 5.10: AWSMAP Integration Architecture

Integrated Development Environment Integration. Software engineers of ISVs create and manage artifacts of a software system using an environment of development tools. Typically *Integrated Development Environments (IDEs)* are used for these activities. AWSMAP supports integration into IDEs via its RESTful API in order to facilitate the inte-

gration of AWSM:RE Knowledge Rediscovery into ongoing development as specified in section 5.3. The implementation in the context of section 2.1 is realized as Visual Studio Extension, presented in fig. 5.11. The screenshot shows creation of a new annotation on an artifact f ,

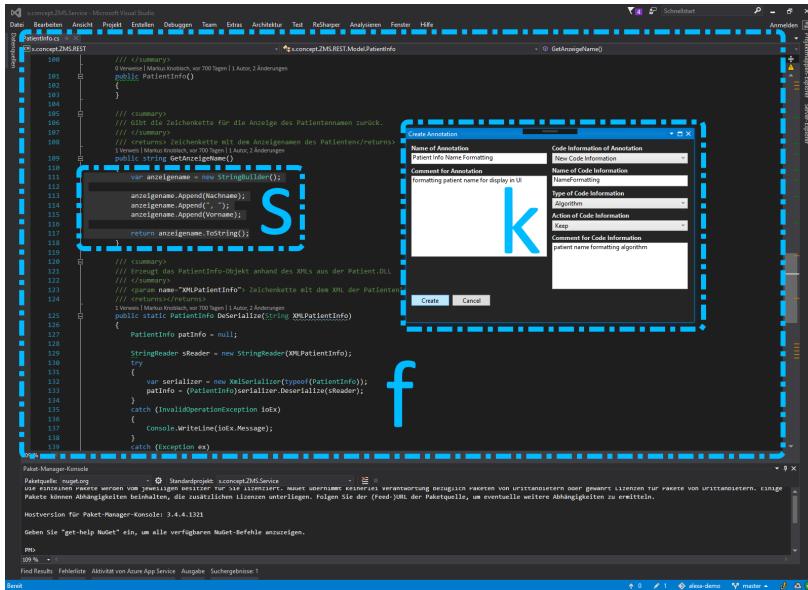


Figure 5.11: AWSMAP IDE Integration

highlighting a segment s and entering information about knowledge instance k . Paper prototypes and interactive MS PowerPoint-based wireframes were used for collaborative design (cf. Co-Creation (IDEO, 2015)) of the IDE integration with ISV Software Engineers. Figure 5.11 shows the creation of a new annotation in the Visual Studio Extension. A part of the source code can be selected in the IDE's source code editor, by right-clicking and selecting “Create Annotation” in the menu, the

dialog displayed on the right side opens. It captures the same data as the corresponding view in the Web user interface of AWSMAP and synchronizes annotations via the RESTful API.

Version Control Integration. For typical ISVs, the artifacts of the Legacy System \mathcal{L} are under version control using a *Version Control System* (*VCS*). AWSMAP supports VCS-integration by implementing the “load codebase” step in fig. 5.2 as import from an existing VCS repository. In the context of section 2.1, VCS integration was implemented using the Microsoft Team Foundation Server (TFS) REST API⁶⁶. The Operator specifies the TFS instance configuration and repository URL in the AWSMAP web.config in the “configure toolchain” step of fig. 5.2 and can then trigger import and parsing of the codebase from TFS VCS via the Admin Dashboard.

Project Management Integration. For Management stakeholders, AWSMAP integrates with TFS *project management* capabilities via the TFS REST API. Integration with ongoing agile development is achieved in the context of the TFS Scrum Process Template. The migration packages described in section 5.3.3 are represented as work items of type feature, its contents as backlog items with parent relationships to the migration package. URLs pointing to the corresponding entity in AWSMAP are added to the descriptions, and URLs to the entity in the TFS Web UI are added to AWSMAP entities to allow switch between the two contexts. Figure 5.12 shows a screenshot of the definition of a migration package in AWSMAP.

Authentication and Authorization Integration. The last integration aspect is *authentication and authorization* of users. ISVs typically

⁶⁶<https://docs.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-5.0> Retrieved: 6.12.2019

The screenshot shows a web-based application interface for managing features in TFS. At the top, there's a header bar with the URL 'arola.informatik.tu-chemnitz.de/eHealthAnnotation/TFS/TFS'. Below the header, a navigation menu includes 'AnnotationTool', 'Dashboard', and 'Adminarea'. A breadcrumb trail shows 'Home / Dashboard / TFS'. The main content area is titled 'TFS' and contains a form for adding features. On the left, a section titled 'Add Features to TFS' has a sub-section 'Name of Package:' with a text input field containing 'Migration Milestone 1'. On the right, a section titled 'Features in TFS' has a red button labeled 'Remove selected Features from TFS'. At the bottom of the form, there's a green button labeled 'Add selected Features to TFS'. The footer of the page includes the copyright notice '© 2015 - AnnotationTool - VSR'.

Figure 5.12: TFS Integration: Migration Packages

operate an Identity Provider to restrict access to their company-internal resources. In the context of section 2.1, *Active Directory (AD)*⁶⁷ integrates with AD for federated authentication, allowing users to login with their domain accounts.

5.4.3 Queryable Legacy Knowledge Base Representation

This section addresses the system actor perspective of AWSM:RE RQ3, the management of extracted problem and solution domain knowledge, and the provision of this knowledge for further usage in Web Migration processes based on Reengineering or Transformation at different degrees of model-driven adoption. The challenge is to provide an external representation of the Knowledge Base, as specified in fig. 5.4, which is suitable for various different Web Migration approaches. Thus, the next two subsections describe the representation of knowledge in the Knowledge Base as ontology and the support for querying.

⁶⁷<https://docs.microsoft.com/de-de/windows-server/identity/active-directory-federation-services> is used, thus AWSMAP Retrieved: 6.12.2019

SCKM Ontology

Migration Engineers require knowledge representations depending on the specific Transformation or Reengineering process. The variety of knowledge and models is high, since AWSM:RE addresses knowledge at PSM or CIM level of abstraction as resulting from design recovery and is designed for integration with existing Web Migration approaches at different degrees of model-driven adoption (principles P2 and P3). This requires a generic and extensible model allowing to associate arbitrary knowledge models with legacy codebases.

Thus, AWSM:RE implements the SCKM as *Ontology of knowledge in Legacy Systems* using the *Web Ontology Language (OWL)*⁶⁸ (Heil and Gaedke, 2016). This allows for a generic knowledge representation that can be extended and integrated with knowledge from external Linked Open Data (LOD) sources, semantics, and reasoning, using SPARQL⁶⁹ as powerful and model-independent query language and enables access to the rich open Web standards-based environment and existing infrastructure in the context of the *Semantic Web*. The SCKM Ontology was designed based on the SCKM using Protégé⁷⁰. Figure 5.13 shows the main OWL classes of the ontology.

The root Annotation class is equivalent to an SCKM annotation as defined in Definition 5. It is modeled using the W3C Web Annotation Data Model (OA) (W3C, 2017b), to allow for portability and sharing across tools. An SCKM annotation $a = (k, l)$ is modeled as W3C OA annotation with the object properties oa:hasBody pointing to the knowledge payload k and oa:hasTarget to location l . Valid bodies are Feature and DomainKnowledge (Heil and Gaedke, 2016), which can

⁶⁸<https://www.w3.org/TR/owl-overview/> Retrieved: 6.12.2019

⁶⁹<https://www.w3.org/TR/sparql11-overview/> Retrieved: 6.12.2019

⁷⁰<https://protege.stanford.edu/> Retrieved: 6.12.2019

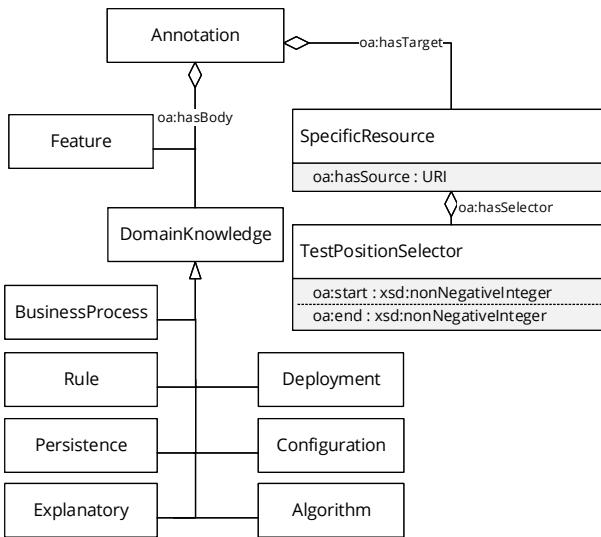


Figure 5.13: OWL SCKM Ontology Classes

be of any of the SCKM problem and solution domain types defined in section 4.6.2. To model the target $l = (s, f)$, an **oa:SpecificResource** is used, that references the legacy artifact f (a KDM SourceFile) as URI via **oa:hasSource**. Segment $s = (\alpha, \omega)$ (a KDM SourceRegion) is represented as **oa:TextPositionSelector** via the **oa:hasSelector** property. The start α and end ω of the segment are 0-based index numbers in the character stream of f represented as **oa:start** and **oa:end**. The complete SCKM Ontology can be seen in appendix D.1.

Querying Support

To allow filtering the information in the potentially large Knowledge Base according to information requirements of the Transformation or Reengineering approaches in use, a key factor is the ability to query the knowledge base. While there is a variety of querying languages

for model-driven approaches, e.g. QVT introduced in section 3.1.1, these are specific for particular model-driven Web Migration approaches and technologies. Following the design decision of P1, the representation and the query language should instead be based on open Web standards. Therefore, the Knowledge Base \mathbb{K}_B can be queried using the SPARQL Endpoint shown in fig. 5.5 to retrieve a set of Resource Description Framework (RDF)⁷¹ triples according to the ontology introduced above, serialized in one of the RDF notations (e.g. N3, Turtle, XML, etc.) available. This allows for interoperability based on open Web standards and supports querying via SPARQL. To support complex use cases, the AWSMAP SPARQL Endpoint supports reasoning. This can be demonstrated with the following example.

When querying Knowledge Bases representing knowledge instances in legacy codebases, queries can become unnecessarily complex due to encoding location-related semantics in the query. Querying all knowledge within a certain area s_a in an artifact f , the semantics of containment need to be expressed in FILTER statements like:

Listing 5.1: Partial SPARQL Containment Query

```
FILTER(  
  ( ?end > ?astart && ?end <= ?aend )  
  ||  
  ( ?start >= ?astart && ?start < ?aend )  
).
```

This definition of 1-dimensional containment is part of the domain modeled by the OWL ontology and should not need to be encoded, neither in queries nor explicitly in the data. Similar to GeoSPARQL⁷², a query should simply state:

⁷¹<https://www.w3.org/RDF/> Retrieved: 6.12.2019

⁷²<http://www.geosparql.org/> Retrieved: 6.12.2019

Listing 5.2: Partial SPARQL Containment Query using sckm:within Property

```
?annotation sckm:within ?area
```

Likewise, complex relationships like knowledge-influences-feature should be interfaced as simple triple patterns. The AWSMAP Querying Support realizes this, by extending the Ontology with additional rules using the *Semantic Web Rule Language (SWRL)*⁷³. SWRL allows defining inference rules based on which additional triples are asserted into the knowledge graph. The following shows an example of a SWRL rule in the Ontology defining the influences-relationship. The complete list of SWRL rules for Querying Support can be seen in appendix D.2.

Listing 5.3: SWRL Rules for sckm:influences

```
oa:Annotation(?AF) ^ oa:Annotation(?AR) ^ sckm:Feature(?F)
    ↳ ^ sckm:Knowledge(?R) ^ oa:body(?AF, ?F) ^ oa:body
    ↳ (?AR, ?R) ^ oa:SpecificResource(?SR) ^ oa:
    ↳ SpecificResource(?SF) ^ oa:target(?AF, ?SF) ^ oa:
    ↳ target(?AR, ?SR) ^ oa:source(?SF, ?S) ^ oa:source
    ↳ (?SR, ?S) ^ oa:TextPositionSelector(?TR) ^ oa:
    ↳ TextPositionSelector(?TF) ^ oa:selector(?SF, ?TF) ^
    ↳ oa:selector(?SR, ?TR) ^ oa:start(?TR, ?sr) ^ oa:
    ↳ start(?TF, ?sf) ^ oa:end(?TR, ?er) ^ oa:end(?TF, ?
    ↳ ef) ^ swrlb:greaterThanOrEqual(?er, ?sf) ^ swrlb:
    ↳ lessThan(?sr, ?sf) ^ swrlb:greaterThanOrEqual(?ef,
    ↳ ?er) => sckm:influences(?R, ?F)
```

5.5 Crowdsourced Reverse Engineering

To answer AWSM:RE RQ2 in section 5.2, this section specifies the Concept Assignment process in fig. 5.3 for Crowd Annotators (Heil, Förster,

⁷³<https://www.w3.org/Submission/SWRL/> Retrieved: 6.12.2019

et al., 2018; Heil, Siegert, et al., 2019). As outlined in section 5.3, the Annotator role can be impersonated by ISV staff, an automated system, or the crowd. To address the limited resources constraint in RO1, automation or Crowdsourcing are suitable as both shift efforts away from ISV staff. While automation has been the focus of Reverse Engineering research, the difficulty of automation of Reverse Engineering and Concept Assignment in particular often leads to low precision or recall (Canfora and Penta, 2007) for complex information systems and value-added knowledge discovery of Concepts requires human experts.

This notion is in line with our initial experimentation in automating Concept Assignment using supervised machine learning techniques. We assessed various feature representation and classification method combinations on 714 manually classified code segments. Passive (repurposed) Crowdsourcing was used for training set creation based on StackOverflow posts. The best performing combinations were Latent Semantic Indexing (LSI) with 200 features combined with Multi-class Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel (Recall 0.57, Precision 0.74, F1 0.64) and Multi-class Rocchio classifier (Joachims, 1997) (R 0.55, P 0.52, F1 0.53), and Information Gain with 200 features combined with Multi-class Naive Bayes with a multivariate Bernoulli model (McCallum and Nigam, 1998) (R 0.75, P 0.50, F1 0.60).

Thus, AWSM:RE focuses on the crowd as alternative Annotator by introducing *Crowdsourced Reverse Engineering* (CSRE) (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019). The *Concept Assignment Problem* (Biggerstaff et al., 1993) can be reformulated as *Classification Problem* (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019): the two-step process of identification of relevant entities and relations and assignment to known domain Concepts becomes identification of relevant code

segments s and selecting a class $c \in \hat{C}$ from a given set of classes. After bootstrapping, through human or system Annotators, the reformulated classification problem can be solved using Crowdsourcing.

We characterize the AWSM:RE Concept Assignment process for Crowd Annotators, as shown in fig. 5.14 in the eight foundational and orthogonal dimensions of Crowdsourcing for software engineering: crowd size, task length, expertise demands, locus of control, incentives, task interdependence, task context, and replication (Latoza and Hoek, 2016).

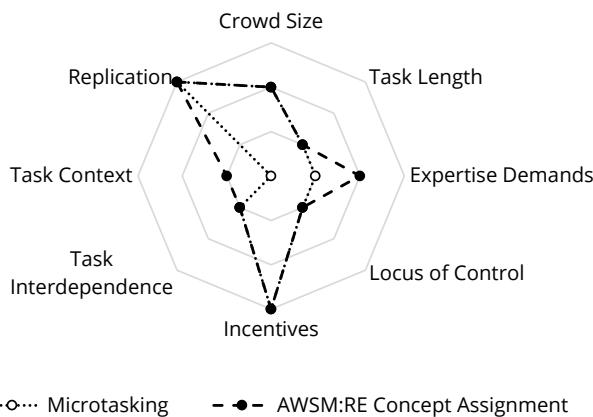


Figure 5.14: Crowd-based Concept Assignment compared to Microtasking
(Heil, Förster, et al., 2018)

The resulting localization in the Crowdsourcing space places AWSM:RE very close to the established and successful *microtasking* (Latoza and Hoek, 2016) or *marketplace* (Daniel et al., 2018) model. Only two out of eight dimensions do not match exactly: expertise demand and task context of AWSM:RE Concept Assignment is higher compared to the low expertise demand and independence of context for typical microtasking tasks. Due to this high similarity, it is likely that microtasking can be similarly successful for the small, independent, and replicatable

classification tasks as for other microtasking applications, benefiting from high worker numbers and parallel execution of tasks. The potential key benefit of reduced time to market requires two characteristics: work can be broken down into short tasks, and each task must be self-contained with minimal coordination demands (Latoza and Hoek, 2016). AWSM:RE meets both of these characteristics.

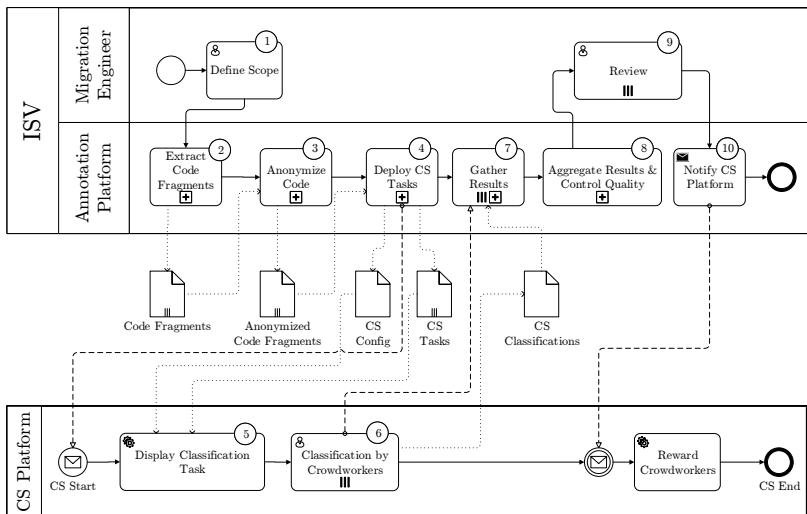


Figure 5.15: Crowdsourcing-based classification process
(adapted from Heil, Siegert, et al., 2019)

For the CSRE extension (Heil, Siegert, et al., 2019) of AWSM:RE Concept Assignment, a suitable *CS Platform* is required, representing a Crowdsourcing marketplace to post an open classification call to a crowd. The adapted process for crowd annotators in fig. 5.15 addresses three main *challenges of the application of Crowdsourcing in Reverse Engineering* (Heil, Siegert, et al., 2019):

1. Automatic Extraction & Creation of Crowdsourcing Tasks from *B*

2. Balancing Controlled Disclosure of Proprietary Source Code with Readability
3. Aggregation of Results and Quality Control

The following subsections outline solutions to address these challenges by detailing the collapsed subprocesses in fig. 5.15. The general process is as follows: the Migration Engineer (1) defines the scope of code to be classified using the filtering capabilities of the Annotation platform described above. The annotation platform automatically extracts code fragments for classification (2), these are pre-processed to achieve the intended anonymization properties (3), and the annotation platform deploys classification tasks in the CS Platform (4). CS Configuration data is passed to the CS Platform to set-up microtasks. It includes a brief description of the Concept Assignment classification task, a URL pointing to the external representation for crowdworkers, the crowdworker selection criteria, and the reward configuration. Suitable crowdworkers are presented a textual description of the available categories for classification (5), according to the ontology in section 5.4.3. The interactive external representation of the code fragment to be classified allows the crowdworker to enter his classification (6). Results are gathered per crowdworker (7), and the classification results are aggregated across the different crowdworkers, and quality control measures are applied (8). The pre-filtered results are forwarded to the review subprocess in fig. 5.3 and (10) the annotation platform notifies the CS platform to reward the participating crowd workers according to the reward policy (9).

This process was implemented as an extension of AWSMAP for experimentation with CSRE Concept Assignment on the bespoke Crowdsourcing platform *microWorkers*⁷⁴ (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019). CSRE Management facilities were added for defining the

⁷⁴<https://microworkers.com/> Retrieved: 6.12.2019

scope, configuring, and starting CSRE projects. Crowdworker Views (cf. fig. 5.16) were added as external representation for crowd Annotators, showing the information required for the microtasks, and handling token-based authentication of crowdworkers. Result Analysis Views were added for judging the progress and outcome of CSRE, as in fig. 5.17 and fig. 5.18. The following three subsections describe solutions for the three main challenges of CSRE introduced above.

5.5.1 Automatic Extraction and Creation of Crowdsourcing Tasks

AWSM:RE Concept Assignment was reformulated as classification problem, and the similarity with microtasking was demonstrated. To apply the microtasking model, self-contained, simple, repetitive, short microtasks are required by automatically dividing *B* into *code fragments* for classification through static analysis. Three *classification task extraction properties* are required: Automation, Legacy Language Support, and Completeness of References (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019). *Automation* means that no additional user interaction must be required. This can be achieved by documentation tools, syntactic analysis tools, or syntax highlighters. *Legacy Language Support* is important since static analysis is language-specific. Relevant commonly used⁷⁵ legacy languages, in particular C, C++, Java should be supported. Crowdworkers require sufficient information for classification: control and data flow need to be available. Thus, *Completeness of References* for the code referenced in a code fragment is required. A comparative feasibility study with students showed that documentation tools are the best-suited alternative since production-grade implementations exist for most programming languages, they keep track of referenced parts of the source

⁷⁵cf. <https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages> Retrieved: 6.12.2019

code, and they work entirely automatic. The implementation of step 2 in fig. 5.15 runs Doxygen⁷⁶ on B and parses the generated documentation artifacts to identify relevant code fragments and referenced code.

5.5.2 Balancing Controlled Disclosure with Readability

The open classification call to a potentially large and unknown group of workers means publishing task contents, bearing the risk of uncontrolled use of the code fragments, e.g. through competitors of the ISV. Proper source code anonymization is required to prevent unintended use but must be balanced with the readability of the source code, allowing the crowd to achieve the classification. This balance is reflected in the following three *anonymization properties*; a suitable approach must: *prevent identification of software provider, software product and application domain, maintain control flow and all information relevant for classification, and avoid negative impact on readability of the source code* (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019). Common code obfuscation techniques are not sufficient as they target readability, but an adapted form of *identifier renaming* is required, extended to all parts of code that contain *identification information* residing in three loci: identifiers, strings, and comments. *Classification information*, that represents relationships between entities in the code should be kept.

The CSRE Anonymization Algorithm (Heil, Siegert, et al., 2019) in algorithm 5.1 implements step 3 in fig. 5.15 and takes a PSM resulting from the static analysis of the task extraction step and a list of identifiers and automatically generates a replacement mapping for the different types of identifiers. The mapping represents simple relationships like generalization and class-instance. We assessed the readability of the anonymized code through a brief experimental validation (Heil,

⁷⁶<https://www.doxygen.nl/> Retrieved: 6.12.2019

Algorithm 5.1: CSRE Anonymization Algorithm

Input: Source Code $f \in B$, Platform Specific Model PSM ,
Identifier List I

Output: Anonymized Source Code

1 $m(i) =$
$$\begin{cases} \text{"instance_of_"} + m(c) & \text{if } i \text{ instance of } c \\ \text{genericName}(i) + \text{"_extends_"} + m(s) & \text{if } i \text{ subclass of } s \\ \text{genericName}(i) & \text{else} \end{cases}$$

2 replace Strings in f by "String"
3 remove comments from f
4 replace all identifiers $i \in I$ in f with $m(i)$
5 **return** f

Siegert, et al., 2019) with employees of an SME-sized ISV who rated the readability of anonymized code fragments on a five-level Likert scale (agreement 1-5 for "The code is easy to read"). Code obfuscation ranked near-unreadable (0.7), CSRE Anonymization (3.7) performed slightly better than the naive approach using dictionary replacements (3.2).

5.5.3 Aggregation of Results and Quality Control

Aggregating potentially contradicting results from unknown crowdworkers and ensuring quality is a challenge to be addressed to justify the ISV's investment. Originating in different experience levels of the crowdworkers and fake answers may lead to poor classification precision. AWSM:RE combines several quality-control design-time (Worker selection, Effective task preparation) and quality control run-time approaches (*Ground truth, Majority consensus*) (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019) to enable the crowdsourcing of Concept Assignment, following guidelines for quality control (Daniel et al., 2018; Allahbakhsh et al., 2013). The quality control measures below constitute the *Quality control and results aggregation properties* of CSRE.

CSRE Concept Assignment uses *reputation-based worker selection* based on previous crowdworker ratings. For our experiments on microWorkers.com, we restricted participation to the “best workers” group. *Effective task preparation* comprises clear and unambiguous task description and *defensive design*. The crowd worker View in fig. 5.16 provides code fragments and references with syntax highlighting, available Concepts to assign and requires min. 50 characters of justification per classification. This slows down fake contributions and enables filtering and analysis.

The CSRE *compensation policy* is a combination of monetary and non-monetary rewards: quality contributions receive 0.30 USD and a positive rating. A *Ground Truth* approach is used as runtime quality control: classification tasks with known solution are added to allow calculation of the individual user score $S(w_i) \in [0, 1]$ for each crowd worker $w_i \in W$ by comparing amounts of correct $C_{w_i}^+$ and incorrect $C_{w_i}^-$ classifications as in eq. (5.1). This score can be used as weight factor in results aggregation.

$$S(w_i) = \frac{|C_{w_i}^+|}{|C_{w_i}^+| + |C_{w_i}^-|} \quad (5.1)$$

CSRE uses *Majority/Group Consensus* for aggregating results from different crowdworkers. For each code fragment, classifications $C \subset W \times A$ are tuples of a worker $w_i \in W$ and an annotation by that worker, expressed as class $c_k \sim a, a \in A$. This creates a voting distribution $V : A \mapsto [0, 1]$ for each possible class c_k as in eq. (5.2).

$$V(c_k) = \frac{\sum_{(w,c) \in C | c=c_k} S(w)}{\sum_{(w,c) \in C} S(w)} \quad (5.2)$$

Sourcecode:

```

public HttpResponseMessage Get(string id)
{
    var result = repository.FindById(id);
    if (result == null)
        return Request.CreateResponse(HttpStatusCode.NotFound);

    return Request.CreateResponse(HttpStatusCode.OK, result);
}

```

References: [Click here to show/hide References](#)

Categories: [Business Process](#)
[Algorithm](#)
Persistence & Data Handling
[User Interface/Interaction](#)
[Explanatory](#)
[Rules](#)
[Configuration](#)
[Deployment](#)

Explanation: Persistence & Data Handling - Reads data from `FindByid` method of `repository` object to decide the status code to be returned.

Save

[Click here to show/hide Categories](#)

Figure 5.16: Crowd Worker View

The aggregated result, the consensual classification c^* is then calculated from majority consensus as in eq. (5.3).

$$c^* = \arg \max_{c \in A} V(c) \quad (5.3)$$

For cases where no clear majority can be found, an overview of result distributions (fig. 5.17) and a display of crowdworker rationales (fig. 5.18) was implemented.

5.6 Evaluation

This section evaluates AWSM:RE regarding three aspects. Section 5.6.1 evaluates AWSM:RE against the requirements in section 5.1.2. Effectiveness and Efficiency are further assessed in detail through experimentation in section 5.6.2. Section 5.6.3 revisits research objective RO1 and the AWSM:RE research questions in the light of the evaluation results.

Statistik		
Kategorie	Prozent	Umwandeln
Rules	47.06	Umwandeln
Persistence & Data Handling	17.65	Umwandeln
Explanatory	11.76	Umwandeln
Deployment	11.76	Umwandeln
Algorithm	5.88	Umwandeln
User Interface/Interaction	5.88	Umwandeln

Statistik inkl. Testfragen	
Kategorie	Prozent
Rules	42.35
Persistence & Data Handling	25.98
Explanatory	10.85
User Interface/Interaction	9.96
Deployment	8.9
Algorithm	1.96

Figure 5.17: CSRE Statistics

5.6.1 Assessment of Requirements

This section reports on satisfaction of the requirements for AWSM:RE.

Effectiveness. The effectiveness of AWSM:RE for knowledge rediscovery is achieved through specification of a Concept-Assignment-based Reverse Engineering technique, which systematically discovers, assigns, and manages knowledge in the Legacy System. A detailed evaluation of results quality is presented in section 5.6.2.

Efficiency. The efficiency of AWSM:RE is achieved through reformulation of Concept Assignment for crowd annotators. This reduces the resource demand for the ISV by moving the work from ISV staff to the crowd. A detailed evaluation showing the results achievable with very limited financial resources is presented in section 5.6.2.

Expertise. The expertise requirement is addressed in AWSM:RE by reusing the results of the cognitive process of program comprehension for Forward Engineering and by automating decomposition of the Reverse Engineering activity of Concept Assignment into microtasks that are solved leveraging crowd expertise. While Web Migration activities always pose some expertise demands for ISV staff in new fields, AWM:RE conditionally meets the expertise requirement as it is a method that is feasible with available staff assuming a general understanding of software engineering activities and learning capability.

Integration. Integration is one of the core design maxims of AWM:RE. Section 5.3 has presented the integration of AWM:RE processes into ongoing development and maintenance activities following the continuous Reverse Engineering paradigm and section 5.4.2 presented the implementation of the integration aspects in the AWM Toolsuite through connection of AWSMAP with core components of ISVs' development environment. Thus, AWM:RE meets the integration requirement.

Knowledge Management. The knowledge discovered through AWM:RE is represented based on open Web standards like OWL, RDF and W3Coa as described in section 5.4.3 and is available for integration with different model-driven or non-model-driven methods through its external representation for subsequent human use via the Web-based user interface and for system use via the queryable interface based on SPARQL and Querying Support as described in section 5.4.3. Thus, AWM:RE meets the knowledge management requirement.

5.6.2 Experimental Evaluation of CSRE

As the idea of CSRE, i.e. application of the Crowdsourcing paradigm to Reverse Engineering, is novel, we conducted evaluation experiments (Heil, Siegert, et al., 2019). The results provide a basis for assessing

efficiency and effectiveness of AWSM:RE for crowd Annotators. Furthermore, the analysis aimed at exploring the suitability of crowdworkers for CSRE and exploring crowdworker behaviors.

Setup. The automatic task extraction described above was used to pre-process the codebase of BlogEngine.NET⁷⁷, from which 10 code fragments were randomly selected. Length distribution of the fragments was between 7 and 57 LOC, average 25.4 LOC. The set of classes \bar{C} was formed from the 8 domain knowledge types of the SCKM ontology fig. 5.13 as Concepts to be assigned to the fragments. All 10 fragments were manually classified so that the *test dataset* consisted of 10 classified code fragments which formed the ground truth as baseline for assessing crowdworker results. Experiments were run on the crowd-extended AWSMAP with microWorkers as Crowdsourcing platform, restricted to the “best workers” group.

Procedure. A classification campaign was run for 14 days between March 14th and March 28th 2017 with 0.30 USD – platform average at this time – as financial reward per 3 classifications. The crowdworkers were presented a detailed explanation of the classification task and the available classifications and then used the Crowdworker View to perform the classifications. The integration mechanism of microWorkers for external Web Applications is an IFrame. 3 code fragments were randomly selected from the pool of 10 fragments available and classified by the crowdworkers, who could repeat the classification until less than 3 fragments not classified by them were available. To observe the crowdworkers’ behavior, focus and blur events in the Crowdworker View were used to track the time spent.

⁷⁷<http://www.dotnetblogengine.net/> Retrieved: 6.12.2019

Table 5.1: CSRE Experimental Results (Heil, Siegert, et al., 2019)

	Categories								
CF	1	2	3	4	5	6	7	8	Consensus
A	1	14	4	1	1	1	1	1	Algorithm
B	0	1	3	0	0	12	0	0	Rule
C	3	0	4	1	0	1	0	0	Persistence
D	2	5	6	3	0	5	2	0	Persistence
E	4	2	1	9	2	0	3	1	UIX
F	0	0	3	0	1	6	7	2	Config
G	3	1	1	2	2	6	0	0	Rule
H	0	2	12	2	4	3	2	2	Persistence
I	0	1	3	1	2	8	0	2	Rule
J	2	2	4	0	1	2	1	4	Persistence/Deployment

Experimental results and descriptive statistics. 34 unique crowdworkers participated in the experiment, contributing 187 classifications on our test dataset. The results are shown in table 5.1. On average, 16 crowdworkers created 18.7 classifications per code fragment. CF are the ten code fragments, Consensus indicates the result of the consensus voting. The numbers of classifications per category are stated in the categories cells. Bold values are the maxima, which are the basis for the majority consensus. Grey background marks the correct classification of the code fragment. Descriptive statistics of the results are shown in table 5.2, where $|C|$ is the number of classifications and $|W|$ the number of crowdworkers. Note that the crowdworker and classification

numbers differ due to multi-selections. The code fragment length in LOC is stated in l , Σt reports the overall time spent, \bar{t} is the average time. Times are reported in seconds. The error rate f_e (cf. eq. (5.4))

$$f_e = \frac{|C^-|}{|C|} \quad (5.4)$$

is the ratio between false classifications C^- and all crowdworker classifications C of a code fragment. In order to investigate indicators of (dis-)agreement between individual crowdworkers for quality control, table 5.2 includes Entropy E (cf. eq. (5.5))

$$E = - \sum_{I=1}^k f_i \lg f_i \quad (5.5)$$

and normalised Herfindahl dispersion measure H^* (cf. eq. (5.6))

$$H^* = \frac{k}{k-1} \left(1 - \sum_{i=1}^k f_i^2 \right) \quad (5.6)$$

from the relative frequencies f_i of the classifications in the $k = 8$ classes. E And H^* both indicate disorder/dispersion among the crowdworkers: unanimous classifications yield $E = H^* = 0$. The higher the disagreement, the more different classifications, the closer E And H^* get to 1. Thus, they are used as indicators of classification certainty in the following analysis. Figure 5.18 shows a subset of the rationales provided by crowdworkers to justify their classifications.

Table 5.2: CSRE Experiment Descriptive Statistics (Heil, Siegert, et al., 2019)

CF	$ C $	$ W $	l	Σt	\bar{t}	f_e	E	H^*
A	24	19	18	2822	122	0.4167	0.6113	0.6906
B	16	16	20	2531	158	0.25	0.3053	0.4427
C	10	10	40	1128	112	0.6	0.6160	0.8
D	23	21	8	3033	131	0.7391	0.7402	0.8948
E	22	18	7	2580	117	0.5909	0.7228	0.8448
F	19	15	28	2857	150	0.6316	0.6146	0.8064
G	15	13	57	3225	215	0.8667	0.6891	0.8395
H	25	21	24	5249	209	0.52	0.6541	0.7893
I	17	13	40	1917	112	1	0.6504	0.7920
J	16	14	12	3393	212	0.9375	0.7902	0.9115

Quality and classification certainty analysis. An average error rate of 0.655 seems high at first. However, through majority consensus 7 of 10 code fragments were classified correctly. The minimum error rate was .25 on fragment B and the maximum 1 for fragment I. Provided a small expertise variation of the participating crowd workers, this indicates differences in the difficulty (fragment I was one of the longest) and the understanding of the categories. Rule was the Concept most frequently assigned (23.5%), Persistence & Data Handling (21.9%) second, was Deployment the least selected (5.3%). No majorities were achieved for the Business Process and Explanatory Concepts, indicating that they might not be described clear enough for the crowdworkers. All other categories were correctly classified by the respective majorities.

Averages of entropy and Herfindahl dispersion measure are $\bar{E} = 0.639$ and $\bar{H}^* = 0.757$, their minima co-occur with minimal error rate, their maxima with the second-highest f_e . Figure 5.19 shows E and H^* plotted over f_e and their fit to a squared regression. There is a significant

Crowdworker	Erklärung	Benötigte Sekunden
4f6ebfbef	As said in description of "Rules", this source code is setting and applying related rules for the "list category". So, it comes in the "rules" category. → Rules	186
57a169c1	It is used for presenting data into category after sorting according to ID. → Explanatory	67
e858273d	This category describes source code, in which general rules or rules of a specific domain are used to check or decide anything. → Rules	49
40809b34	I choose this category because of two things I noticed inside this database and first is the parent word that can explain and show the meaning of rules because when you say parent it explains many things as advice and how to live and that what means rules to process and the second one is guide and everyone knows the parent guides and explains how to deal with things and also that too specific to rules → Rules	198
0fa041fb2	This method is used to create a XML file. If the file is not available than the XML file is created. List of Categories with XML files. → Deployment	84
81a80188	This code contains deployment commands and the server is hosted for the same purpose. → Deployment	37
a8ae5daa	because it defines rules of listing the entire method. → Rules	126
1e386615	the code contains rules to check if statements are true or false → Rules	65
e0aefc92	Since the methods checks if an object is not null throughout the code it belongs to Rules category. Since it processes an xml file to create a list of Category objects, which is a data structure. It belongs to Persistence & Data Handling category → Persistence & Data Handling → Rules	323
d76dbbe0	Persistence & Data Handling - the code parses an XML file and creates a list of Category objects. Rules - checks if the file with the given name exists → Persistence & Data Handling → Rules	183
8ba95609	Algorithm is a way to understand the logical facts. It helps to make program on C# or any other languages. In this process programming becomes very easy. Explanatory is also useful to understand why the particular source code is used → Algorithm → Explanatory → Rules	14
71535d20	method, that fills a field of categories with content, so the user can interact afterwards → User Interface/Interaction	42
dba271d8	The provided function is used to save the blog details such as category and description related to it. A new category is created and stored in the categories.xml file located on server. The updated categories list is then returned as a list. → Persistence & Data Handling	9
Durchschnittszeit		106

Figure 5.18: CSRE Results View in Annotation Platform, showing classification justifications provided by crowd workers during the experiment and required classification times

($\alpha = 0.05$) positive correlation (Spearman's $\rho = 0.649$, $p = 0.049$) between f_e and H^* and between f_e and E ($\rho = 0.6$, $p = 0.073$), significant

at $\alpha = 0.1$. Under the peaks at $f_e = 0.8$), that means that E and H^* can be used as classification thresholds, i.e. the more crowd workers voting a Concept, the less likely it is a wrong classification. No clear majorities for wrong classifications were observed, supporting the fundamental Crowdsourcing principle “*wisdom of the masses*” and the majority consensus assumption, that majorities are indicative of correct answers.

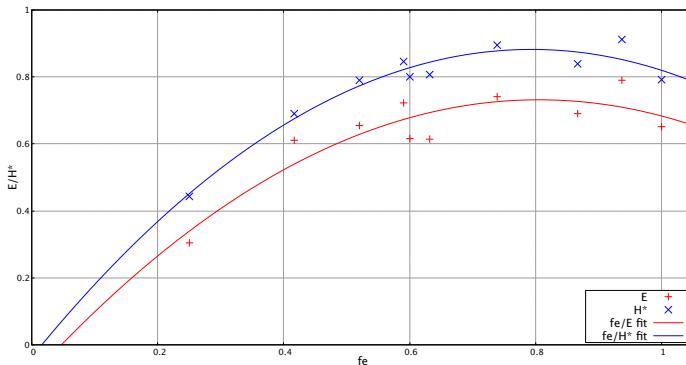


Figure 5.19: Error Rate and Entropy/normalized Herfindahl measure

No correlation between length of code fragments and classification time was found, indicating the influence of other variables such as different levels of difficulty/clarity of the classification. Considering correctness, correct classifications showed fewer time outliers⁷⁸ than wrong classifications. A likely interpretation is that longer classification time and explanations are indicative of uncertainty, leading to wrong classifications in most observed cases. Most outliers in fig. 5.21c belong to *Persistence & Data Handling*, the second most frequently voted, and indicating that the formulation of this Concept is not precise enough.

⁷⁸we use $Q3 + 1.5 \text{ IQR}$ as outlier threshold

Crowdworker behavior analysis. Crowdworker behavior was analyzed through consideration of the distributions of time and explanation length, as well as derived length-time ratio and average explanation similarity per worker, as shown in fig. 5.20. Time median was $\tilde{t} = 117.5$ s, but the observed times varied widely. The interquartile range was at $IQR = 143.25$ s, while upper outliers reached near half an hour (1606s). The relatively low times (Q1 at 65s, min time 6s) show the microtask nature but also point to fake contributions as described below. However, longer times do not imply better accuracy: all but one time outlier in fig. 5.20a belonged to C^- . The results of one particular crowdworker showed suspiciously identical time measurements (3x14s, 3x33s, 3x515s) and identical explanations in all three groups, most likely resulting from the use of a record-and-replay script.

Explanation lengths in fig. 5.20b range from 51 to 1017 characters (median $\tilde{d}=119$) and are relatively close ($IQR=100.5$) to the minimal threshold of 50 chars of explanation. Most of the crowdworkers wrote rather short explanations in short time. This can be seen by considering the lower quartile of 76.25 characters and the outliers for time and length only appearing above the third quartile. Figure 5.21 shows the values for explanation length and time in relation, with correct classifications in green and wrong in red.

Analysis of fake contributions. Cheating is an issue for the quality of Crowdsourcing (Daniel et al., 2018; Allahbakhsh et al., 2013). Observations in time and explanation length distributions revealed that some crowdworkers tried to gain the reward quickly through fake contributions. To identify very fast classifications, length-time ratio⁷⁹ was calculated. As shown in fig. 5.20c, within a range of 0.13 to 61.2 cps, the majority of values is distributed very closely ($IQR 0.96$ cps) around

⁷⁹measured in characters per second (cps)

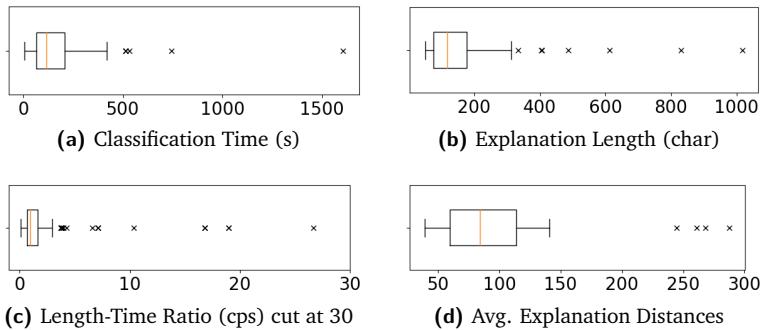


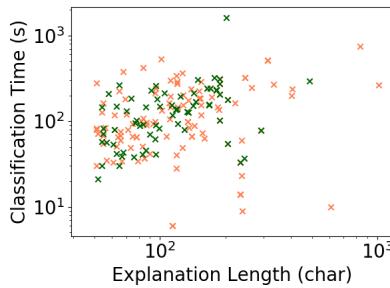
Figure 5.20: Classification Time and Explanation Length: 1-dimensional Distributions (Heil, Siegert, et al., 2019)

a median of 0.94 cps. Analysis of the 20 outliers showed that they are likely to have copied texts. These copies were most often copies of own previous explanations and in some cases from the task description. As the measured time does not only include time for typing the rationales in the explanation field, but also source reading, understanding, deciding and selecting the classification, very high length-time ratios were only reachable if little time was spent for thorough consideration, as evident by only 4 of 20 outliers belonging to C^+ . In any case, speeds are not likely to exceed the upper level of 16 cps measured at competitions⁸⁰. However, 7 crowdworkers exceeded this level. Manual analysis of explanations provided by the outliers determined the fastest worker not to copy text and to classify correctly at approx. 6.6 cps. The upper quartile at 1.63 cps, the vast majority of crowdworkers produced results in reasonable time.

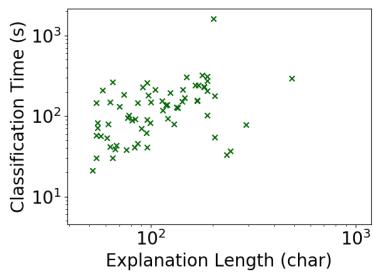
To further identify workers completing the tasks very quickly through copying, average similarity of explanations per user was calculated

⁸⁰cf. <http://www.intersteno.org/> Retrieved: 6.12.2019

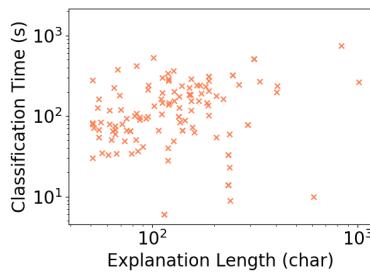
using pairwise Levenshtein distance. The averages range from 39 to 287 and are concentrated (IQR=54.5) around a median of 83.8 (cf. fig. 5.20d). Identical copies (distance 0) were recorded from 10 of 34 (29.4%) workers, but the minimum Levenshtein average of 39 indicates that crowd workers did not exclusively copy. Normalized with \bar{l} , two workers had even less than 50% relative changes, copying most explanations with only minor adaptions. Manual inspection of responses showed that 3 of 34 exclusively put code in their explanations, indicating a wrong understanding of the task.



(a) correct and wrong



(b) correct only



(c) wrong only

Figure 5.21: Classification Time and Explanation Length (log scaled) (Heil, Siegert, et al., 2019)

High crowdworker commitment. In contrast to the negative cases analyzed in detail above, also very thorough workers have been observed: Fragment J was split-voted as Persistence/Deployment assignment. The explanations argued that J is related to persistence because the fragment is part of a class related to persistence. This observation was fascinating, because our dataset did not include the code of the surrounding class. Thus, several crowd workers looked up the sample on the Web and also read the surrounding parts from an external source in order to classify. This level of active commitment and investment of time by the crowdworkers to complete their task was positively surprising.

Threats to Validity. The experiments conducted in the context of this thesis all comprise an analysis of threats to validity. This analysis is structured according to the three different types of validity: construct validity, internal validity, and external validity. *Construct validity* is the validity of the experimental design, determining whether the measured indicators are appropriate for the intended construct as abstraction of the latent variables. *Internal validity* is the validity of cause-effect relationships observed in an experiment established by ruling out alternative explanations as originating from systematic error. *External validity* is the validity of generalization based on the results of an experiment. (Creswell, 2014)

Construct validity of this experiment is threatened by the explanations for the available classes in \bar{C} , which may have lead to differences in the understanding across crowd workers. However, these differences are also expected when operationalizing the CSRE method and are addressed by the multiple redundancy of classification results. Another threat to construct validity is the relatively hidden possibility to assign several classes. As only very few crowdworkers used this functionality, their impact on the results is limited. The third threat to construct

validity are random classifications. These were, however, expected and addressed both through the quality control measures and our detailed analysis of fake contributions showing, that the vast majority of contributions was appropriate and that the quality control measures are effective in mitigating the effects of fake contributions. The measured error rates are valid indicators of the desired classification quality required for ISV usage, as higher errors lead to more effort for checking the results, lowering the advantage of employing Crowdsourcing.

Internal validity of this experiment is threatened through potential subjective biases. This is addressed through a purely tool-based interaction with the test subjects, i.e. the crowdworkers. There was no interaction between the researchers and the test subjects; all test subjects used the exact same experimental setup, which was unchanged throughout the entire experimental timespan. There were no hints towards the correct or expected classification result in the user interface, and the test objects, i.e. the code fragments, were randomly assigned to the test subjects, avoiding the bias of improved results through learning over time on specific code fragments. The 14 days experimentation timespan reduces the test subject selection bias, allowing persons of different time-availability, e.g. weekend-only vs. weekdays, business hours vs. evening, to participate.

External validity of the experiment is threatened through limitations in the generalizability of results. The main factor is the test subjects who conducted the experiment. As described above, the two weeks experimentation period reduced test subject selection bias, so that the 34 participating crowdworkers from the microWorkers Crowdsourcing platform can be considered sufficiently representative of the quality characteristics for crowdworkers on this general-purpose Crowdsourcing platform. While the results cannot be generalized to arbitrary plat-

forms, we would expect an even better quality on a dedicated software development platform such as topcoder⁸¹. The test objects, i.e. the code fragments, limit generalisability of results since they are selected from one specific technology and platform, C# and Microsoft .NET. However, we expect results for other technologies and platforms used in Legacy Systems such as Java Swing, C/C++ MFC to be at least similar due to higher popularity⁸² and therefore greater availability of crowdworkers experienced in this technological base.

Conclusion of CSRE Experimentation. In spite of the cases of low quality and fake contributions reported above, the CSRE quality control measures proved robust enough to yield 70% overall correctness. The experiment has shown that the expertise level of the best crowd workers group on Crowdsourcing platform microWorkers in combination with CSRE quality control is sufficient to perform the Reverse Engineering classification activity and produce decent results. The overall correctness of 70% is a good result similar to what can be achieved by a single expert performing the same task. However, with less than 20 USD expenses for classifying the ten code fragments, Crowdsourcing is a significantly more cost-effective solution. The results indicate that Crowdsourcing can be applied to perform AWSM:RE Concept Assignment reformulated as classification problem when it is automatically broken down into microtasks, and the process is guided by CSRE quality control methods. CSRE Concept Assignment defines a novel manual knowledge rediscovery approach reducing the resource demand for ISVs through Crowdsourcing. Thus, the requirements effectiveness and efficiency requirements from section 5.1.2 are met.

⁸¹<https://www.topcoder.com> Retrieved: 6.12.2019

⁸²cf. TIOBE Index <https://www.tiobe.com/tiobe-index/> Retrieved: 6.12.2019

5.6.3 Research Results

The research objective addressed by this chapter, RO1, to enable identification and management of existing knowledge in legacy source code with limited resources and lack of Web Engineering expertise, is achieved. The effectiveness and efficiency of the AWSM:RE Method for knowledge discovery have been demonstrated and detailed through experimentation, its feasibility with limited resources and lack of expertise is achieved through Crowdsourcing and Integration with ongoing development and its Knowledge Management functionality based on stakeholder-specific user interfaces, and queryable standards-based representations were presented. AWM:RE RQ1 was answered through the knowledge discovery technique defined in section 5.3, its integration into ongoing development and its realization as crowdsourced activity. AWM:RE RQ2 was answered through the introduction of the novel Crowdsourced Reverse Engineering strategy presented in section 5.5. AWM:RE RQ3 was answered through specification of the Annotation Platform presented in section 5.4.

5.7 Summary

This chapter presented the AWM Reverse Engineering method, which facilitates problem and solution domain knowledge rediscovery for ISVs with limited resources, specifying a novel crowdsourced Concept Assignment strategy. AWM:RE integrates with ongoing development and provides a queryable open Web standards-based knowledge representation to facilitate integration with other Web Migration methods. The AWM:RE techniques and their implementation in the Annotation Platform have been described. Experimental evaluation has shown the feasibility of the approach, quality of its results, and low related cost and effort. It has furthermore provided insights on crowdworker behavior and suitable quality control measures.

AWSM Risk Management Method

6

This chapter addresses research objective RO2:

To enable demonstration of desirability and feasibility of a potential Web-based version of the Legacy System with limited resources and lack of Web Engineering expertise.

First, the current situation is briefly analyzed to identify requirements and review related work. Three research questions are derived from RO2, inspired by the analysis results. To address the research questions, three sections outline the three main contributions towards RO2 in the context of the AWSM:RM method: the novel concept of Rapid Web Migration Prototyping is introduced and a process for rapid, semi-automatic creation of Web Migration prototypes is described, a technique for business logic reuse in web migration prototypes based on WebAssembly is presented, and a technique for automatic Transformation of legacy user interface to Web-based user interface prototypes is described. Finally, the method is evaluated against the requirements, and the evaluation is supported by three experiments, which include analysis of both empirical evaluations and objective measurements.

6.1 Analysis

The following analysis briefly summarizes the situation of doubts about desirability and feasibility of a Web-based version of the Legacy System and the challenges of demonstrating these properties through application of Rapid Prototyping in the Web Migration domain. Requirements are derived from this analysis and an overview of related work on the two key paradigms used for AWSM:RM is given.

6.1.1 Situation and Challenges

Our field research has revealed *doubts about desirability* and *doubts about feasibility* as two major factors rendering ISVs hesitant to commence a Web Migration, as described in section 2.2.2. *Feasibility studies* are the most common risk management approach, employed by 7 of the 23 approaches assessed in section 3.2. They are often conducted as *pilot migration projects*, i.e. migration trials with a restricted scope that migrate a small portion of the Legacy System in order to identify insuperable technical obstacles as early as possible (Harry M. Sneed et al., 2010b), to test the migration process and toolchain (Amazon Web Services Inc., 2018) and to facilitate an informed decision in favour or against commencing a full-scale migration (Harry M. Sneed et al., 2010b). However, these feasibility studies and migration pilots put the main focus on the technical feasibility of the migration itself. They fail to create *concrete and tangible results* that allow assessing the *plausibility of a Web-based version* of the Legacy System and the desirability with regard to *advantages of Web Systems*, leading to a gap as identified in G2.

In Forward Engineering, Prototyping (Wallmüller, 2001) and in particular *Rapid Prototyping* (ISO/IEEE, 2017b; Gordon and Bieman, 1995) is used as *effective risk management technique* (Wallmüller, 2001) for risk reduction (ISO/IEEE, 2017b) to address desirability and feasibil-

ity in early stages, allowing to create a running software prototype as concrete and tangible result (Alavi, 1984). Prototyping is particularly appropriate for unfamiliar situations with limited experience to draw from (Tripp and Bichelmeyer, 1990), like the new target environment in Web Migration, which significantly differs from the legacy Desktop Application environment. In Web Engineering, quick creation of cheap, throw-away versions representing the core functionality and characteristics of the envisioned Web System has become industry practice as it enables discussing solution alternatives and facilitates communication with stakeholders (Alavi, 1984; IDEO, 2015). This communication and risk reduction can accelerate modernization plans (Forrester Research, 2011) by addressing resistance within the organization (Khadka, Batlajery, et al., 2014; Harry M. Sneed et al., 2010a) making reasons and consequences of modernization visible. In this way, prototypes represent a concrete and tangible contribution to the business case that serves as a means of communication for stakeholders to support decision making. Transfer of these potential benefits into the Web Migration domain is the primary motivation of AWSM:RM.

Web Migration, however, is inherently different from forward Web Engineering. While a prototype in forward Web Engineering is created from scratch based on early and potentially unclear versions of requirements, a *web migration prototype* is built based on elaborate requirements represented by and potentially recovered from an existing and mature Legacy System already known to stakeholders and source of their expectations. This *brownfield* (Hopkins and Jenkins, 2008) situation constitutes the main difference between Web Migration and forward Web Engineering. An ISV employing Rapid Prototyping for forward Web Engineering can rely on a staff base with Web Engineering expertise that can leverage open Web standards like HTTP, HTML, CSS, JavaScript and popular Web Development platforms and frameworks

like node.js⁸³, Rails⁸⁴, React⁸⁵, Angular⁸⁶ to quickly create running prototypes. In contrast, an ISV, as described in section 2.1.1, has a staff base with expertise in legacy platforms like C, C++, Java and Desktop Application frameworks like MFC, WPF⁸⁷, Swing⁸⁸.

Rapid Prototyping is, therefore, not directly applicable to Web Migration. The Rapid Prototyping paradigm needs to be adapted to the characteristics of migration and the situation of ISVs with non-Web legacy desktop software, following principle P4. Thus, the challenge addressed by AWSM:RM in order to achieve RO2 is to specify a suitable risk management method for quick creation of concrete, tangible prototypes of a Web-based version of the legacy desktop system that can be used to demonstrate desirability and feasibility for communication decision making of migration stakeholders with limited resources and Web Engineering expertise.

6.1.2 Requirements

The following requirements specify RO2, based on the analysis presented above and the AWSM principles.

Effectiveness. Risk management should be enabled through the creation of concrete, tangible results allowing to demonstrate desirability and feasibility of Web Migration.

Efficiency. Creation of web migration prototypes should be supported by tools to require fewer resources compared to manual creation.

⁸³<https://nodejs.org/> Retrieved: 6.12.2019

⁸⁴<https://rubyonrails.org/> Retrieved: 6.12.2019

⁸⁵<https://reactjs.org/> Retrieved: 6.12.2019

⁸⁶<https://angular.io/> Retrieved: 6.12.2019

⁸⁷<https://msdn.microsoft.com/en-us/library/aa663364.aspx> Retrieved: 6.12.2019

⁸⁸<https://docs.oracle.com/javase/8/docs/technotes/guides/swing> Retrieved: 6.12.2019

Expertise. Creation of web migration prototypes should be feasible with available expertise of the ISV's staff.

Reuse. Creation of web migration prototypes should reuse existing functionality from the source Legacy System.

Plausibility. The web migration prototypes should enable assessment of the plausibility of a Web-based version of the Legacy System by representing significant architectural changes: Client/Server communication, URL-based navigation, and UI Rendering in a Web browser.

6.1.3 Related Work

This section introduces Prototyping and Rapid Prototyping as key paradigms of AWSM:RM. It establishes the semantics of the key terminology of Prototyping used in the following description of AWSM:RM and provides a minimal overview of related work in these fields.

Prototyping is a risk reduction method (ISO/IEEE, 2017b) defined as:

Definition 9: Software Prototyping (IEEE Computer Society, 2014)

Software prototyping is an activity that generally creates incomplete or minimally functional versions of a software application, usually for trying out specific new features, soliciting feedback on software requirements or user interfaces, further exploring software requirements, software design, or implementation options, and/or gaining some other useful insight into the software.

Software prototypes represent a *functional, demonstrative product*, proving “the relevance of a solution” (ISO/IEEE, 2017b). *Explorative Prototyping* seeks to integrate user feedback in the design process and creates *horizontal prototypes*, which focus on user interaction (Wallmüller,

2001). A *demonstrative prototype* communicates ideas of the user interface and capabilities of the envisioned system, allowing decision makers to assess desirability and benefits (Wallmüller, 2001).

Rapid Prototyping is widely used in Agile Software Development, as it suits the “working software over comprehensive documentation” value (Fowler and Highsmith, 2001) and the principles of continuous and frequent delivery of valuable software, collaboration of business people and developers and working software as primary measure of progress of the Agile Manifesto⁸⁹. Rapid Prototyping can be used as a starting point for Agile Model-Driven Web Engineering (MDWE) as in MockupDD (Rivero and Rossi, 2013; Rivero, Heil, Grigera, Gaedke, et al., 2013; Rivero, Heil, Grigera, Robles Luna, et al., 2014). Rapid Prototyping is defined as follows:

Definition 10: Rapid Prototyping (ISO/IEEE, 2017b)

type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process

Prototyping in Web Migration, in contrast, has not received much attention. Among the Web Migration approaches assessed in section 3.2, only IC4 (Fowley, Elango, et al., 2017) proposes the use of prototypes in cloud migration. IC4 prototypes, however, focus on technical feasibility; prototypes are created to compare different Reengineering options and create performance benchmarks. IC4 takes a more coarse-grain perspective comparing top-level architectural decisions regarding component allocation, e.g. virtualized relational database vs. PaaS SQL services vs. NoSQL. To a lesser extent, rapid relocation approaches like

⁸⁹<https://agilemanifesto.org/> Retrieved: 6.12.2019

AppCloak (Tak and Tang, 2014) can be considered a form of Rapid Prototyping for Web Migration as they allow to quickly create Web (cloud-based for AppCloak) versions of legacy Desktop Applications. However, the focus is on technical benchmarks, and the disadvantages of Encapsulation approaches, as discussed in section 3.3, apply. Other Web Migration methodologies fail to address benefits and early customer feedback in pre-migration phases.

Due to the success of Rapid Prototyping in achieving similar goals – demonstration of an envisioned software for communication, feedback and decision making – in Forward Engineering, section 6.3 describes the application of Rapid Prototyping in Web Migration by specification of a technique for creating demonstrative, functional, horizontal prototypes based on existing Legacy Systems.

6.2 Research Questions

The analysis presented above raises three research questions:

AWSM:RM Research Question RQ1: How to demonstrate desirability and feasibility of a potential Web-based version of the Legacy System with limited resources and lack of Web Engineering expertise?

AWSM:RM Research Question RQ2: How to transfer the Rapid Prototyping paradigm into a Web Migration context?

AWSM:RM Research Question RQ3: How to provide guidance and automation for Rapid Web Migration Prototyping to enable execution with limited Web Engineering and Web Migration expertise?

To design and evolve a solution for research objective RO2, this chapter addresses these three research questions in the following sections.

6.3 Rapid Web Migration Prototyping

To enable the demonstration of desirability and feasibility of a potential Web-based version of the Legacy System set as research objective RO2, AWSM:RM allows rapid creation of a Web-based functional demonstrative prototype of the Legacy System in early stages of Web Migration. It provides an answer to AWM:RM RQ1. Addressing the resources and expertise constraint of RO2, AWM:RM defines reuse-based techniques and tools which allow the creation of such prototypes with limited resources and lack of Web Engineering expertise. Section 6.3.1 provides an overview of the method, section 6.3.2 addresses integration of the method with ongoing development activities, and section 6.3.3 describes the use and adaption of an existing method for API prototyping.

6.3.1 Overview

This section presents an overview of the AWM:RM method and its related tools as visualized in fig. 6.1.

The core idea of AWM:RM is to transfer Rapid Prototyping to the Web Migration domain, answering AWM:RM RQ2. AWM:RM is a method realizing the novel concept of *Rapid Web Migration Prototyping* (Heil, Siegert, et al., 2018), which we define as follows:

Definition 11: Rapid Web Migration Prototyping

Rapid Web Migration Prototyping is a type of Rapid Prototyping which allows the rapid creation of demonstrative, horizontal prototypes, that represent future Web-based versions of existing non-Web Legacy System allowing to assess and demonstrate their plausibility and desirability for decision making in Web Migration.

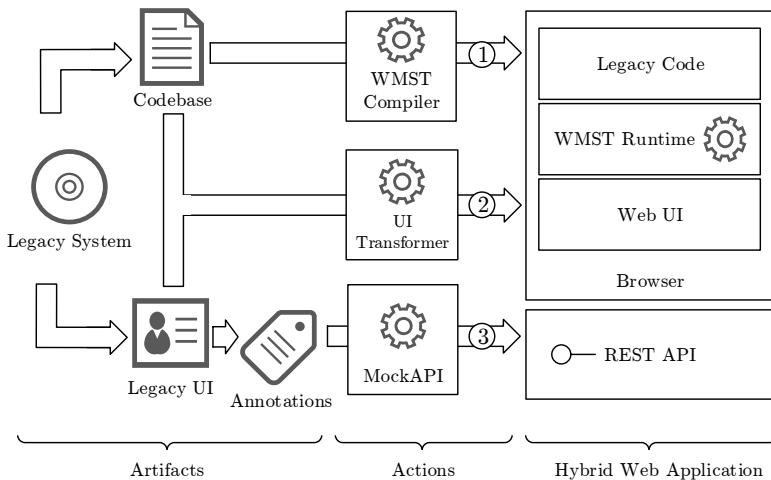


Figure 6.1: AWSM:RM Prototyping Overview

AWSM:RM prototyping is explorative and the demonstrative prototypes created are *horizontal prototypes* (Wallmüller, 2001) focusing on the UI and application logic layer to demonstrate capabilities and interaction of a Web-based version of Legacy System \mathfrak{L} . Thus, the reuse to enable application of the method with limited resources and expertise focuses on the upper application layers of business logic and user interface. As shown in fig. 6.1, AWSM:RM supports three **actions**:

1. Reuse-oriented migration of legacy business logic leveraging a suitable *web migration support technology* (*WMST*) for execution in the Web browser, described in section 6.4
2. Semi-automatic Transformation of the legacy user interface into a Web user interface, described in section 6.5
3. Rapid Prototyping of a RESTful API leveraging the *MockAPI* approach, as described in section 6.3.3

For these actions, AWSM:RM consumes and produces the following **artifacts**:

- Legacy System \mathcal{L} is the primary input source, of which
- Codebase B is used in action 1 and 2
- User Interface U based on its description in B is used in action 2 and U based on the legacy executables in E in action 3
- Persistent data objects D as represented in U are used in action 3
- Annotations created as intermediary artifacts as part of action 3

To conduct the actions, several human and system actors are involved, assuming the following roles. The Prototyping Engineer is the human actor of the AWM:RM actions. The role is assumed by a Migration Engineer of the ISV (cf. table B.1). The Annotator is the human actor creating the annotations required for action 3 and is a specialization of a Migration Engineer. WMST Compiler is a system actor representing the toolchain for reuse of business logic. It works in combination with the WMST Runtime to allow execution of legacy code in the Web browser, based on a suitable WMST. The UI Transformer is a system actor representing the toolchain for the Transformation of the legacy UI to the Web. MockAPI is the system actor representing the toolchain for rapid creation of a RESTful API.

As seen in fig. 6.1, the resulting web migration prototype is a *Hybrid Web Application*, i.e. it consists of parts of reused legacy code, a transformed Web UI, and a generated RESTful API, which demonstrates feasibility and desirability of a Web Migration with limited effort. The prototype life cycle can end after serving its purpose of means of communication and Web Migration decision making, or it can be incrementally improved into a full Web Application by migrating the business logic contained in the legacy code towards client- or server-side Web programming

platforms. Figure 6.2 shows a screenshot of a web migration prototype⁹⁰. The Web user interface is controlled by a client-side backend, which reuses parts of the original legacy code.

Representing action 1, section 6.4 describes a technique for reuse of business logic for Rapid Web Migration Prototyping. Realization of action 2 is presented in section 6.5 through a semi-automatic Transformation of legacy user interfaces to Web user interfaces. Action 3 is based on an existing approach from forward Web Engineering, which we briefly summarize and adapt to AWSM:RM in section 6.3.3.

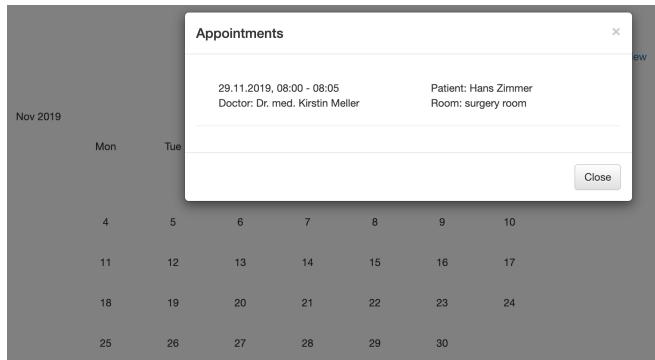


Figure 6.2: Web Migration Prototype with Web Frontend and Reused Legacy Business Logic in the Backend

6.3.2 Integration with Ongoing Development

Integration with ongoing agile software development activities (C4 Agile) is an important requirement for all AWSM methods the neglect of which limits applicability of many Web Migration approaches as described in G1. To avoid this, AWSM:RM method needs to be integrated

⁹⁰source code & life demo: <https://vsr.informatik.tu-chemnitz.de/demos/ReWaMP>
Retrieved: 6.12.2019

with ongoing daily development activities of the ISV on the process and also on the artifacts level. In Forward Engineering, Rapid Prototyping is applied early in the development process due to its feedback function and many agile development approaches advocate iterative and incremental Prototyping towards an operational prototype (Rivero and Rossi, 2013; Rivero, Heil, Grigera, Gaedke, et al., 2013; Rivero, Heil, Grigera, Robles Luna, et al., 2014). Thus, integration of AWSM:RM is intended in the early stages of a Web Migration project. As the Rapid Prototyping paradigm is widely practiced in agile development in industry, AWSM:RM fits nicely in ongoing agile development processes. The paradigm itself, i.e. its motivation, value, and high-level process, is known to the developers and does not require further introduction efforts. The activities required for its three actions can be integrated with ongoing Forward Engineering activities in the same context where prototyping activities for implementation of new functionality or testing new technologies are scheduled. In the context of the Scrum-based development process in section 2.1.1, AWSM:RM prototyping activities can be integrated as backlog items in a sprint. The required effort is limited so that an initial AWSM:RM web migration prototype can be achieved within one sprint in parallel to other development activities without blocking too many resources. On artifacts level, the resulting prototype represents a product increment.

6.3.3 API Prototyping using MockAPI

Web migration prototypes focus on UI and application logic on the client side. The persistence layer and server side are not the primary concern of a horizontal prototype, because the main motivation is to provide a preview of a potential Web-based version of the legacy application, not technical feasibility analysis. However, to achieve a realistic demonstrative prototype in a Web Migration context, the client/server communication via a public API needs to be represented. The vast majority (95.8%)

of public Web APIs are REST APIs (Neumann et al., 2018). To solve the technical challenge of providing a prototype with client/server communication via a REST API, AWSM:RM integrates with an existing method for agile prototyping of REST APIs, MockAPI (Rivero, Heil, Grigera, Gaedke, et al., 2013; Rivero, Heil, Grigera, Robles Luna, et al., 2014) and adapts it to Rapid Web Migration Prototyping as described in the following.

MockAPI, as shown in fig. 6.3, is an agile MDWE approach for rapid creation of RESTful API prototypes based on the annotation of *user interface mockups*. These visual sketches of an envisioned user interface are annotated with *tags* according to the *domain-specific language* (DSL) defined in the MockAPI metamodel (Rivero, Heil, Grigera, Gaedke, et al., 2013) and its extended version, ELECTRA (Rivero, Heil, Grigera, Robles Luna, et al., 2014) supported by a Web-based tagging tool⁹¹. The DSL allows to specify the data model in terms of data objects and Create, Read, Update, Delete, Search (CRUDS) operations, as well as additional constraints and filters. Based on the specifications in tags, a running RESTful API prototype is then generated for different backends like WebComposition/DataGridService (Chudnovskyy and Gaedke, 2010) or node.js.

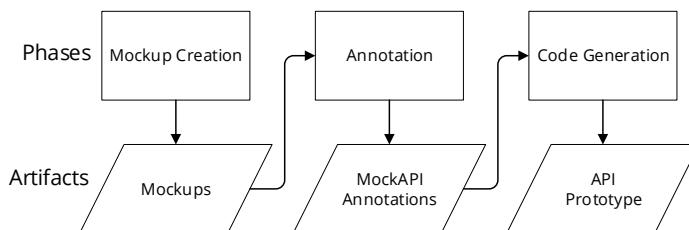


Figure 6.3: MockAPI Process Flowchart

⁹¹<http://agilemdd.lifia.info.unlp.edu.ar/mockapi/> Retrieved: 6.12.2019

Integration of AWSM:RM with MockAPI on process level is achieved by following the MockAPI process for action 3 in fig. 6.1. Integration on artifacts level is achieved by using screenshots of legacy UIs u instead of UI mockups as input for MockAPI. As described in section 4.6.1, these visual artifacts can be retrieved from the executables E of Legacy System \mathcal{L} . As MockAPI was designed to support different levels of mockups, hand-drawn sketches, as well as digital mockups, created using tools like balsamiq⁹², legacy UI screenshots and mockups can be used interchangeably. Figure 6.4 shows the annotation of a screenshot from a legacy desktop application user interface in the MockAPI editor.

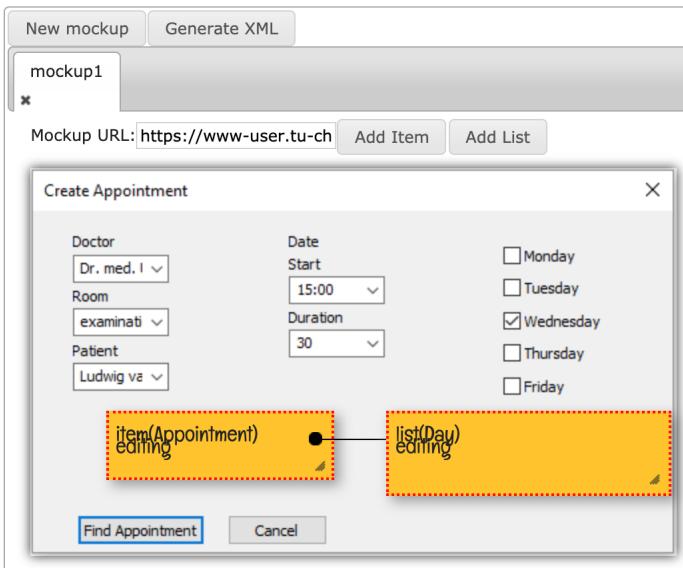


Figure 6.4: Applying MockAPI to a Legacy Desktop User Interface Screenshot

⁹²<https://balsamiq.com/> Retrieved: 6.12.2019

6.4 Reuse of Business Logic

To achieve the rapidness of the Rapid Web Migration Prototyping with limited resources and to make use of the available expertise of ISV staff according to RO2, this section describes a technique for Business Logic reuse in Rapid Web Migration Prototyping. Its focus on rapidness provides the second part of the answer to AWSM:RM RQ2. Due to the horizontal nature of AWSM:RM prototypes, the focus is on the reuse of behavioral business logic (BBL) on the higher layers of UI and application logic, similar to the controller layer in an MVC architecture.

Reuse of business logic in the approaches assessed in section 3.2 has been achieved through Encapsulation – with the shortcomings with regard to representativeness for user interaction with a real Web Application as outlined in section 3.3 – or through Transformation approaches for the server side. Client-side reuse of business logic, in contrast, has not received much attention.

Section 6.4.1 assesses suitable technologies for Client-side reuse of business logic, section 6.4.2 outlines the reuse technique of AWSM:RM, and section 6.4.3 presents the supporting toolchain and section 6.4.4 describes a guided prototyping assistance facilitating execution of the technique on the toolchain.

6.4.1 Assessment of Web Migration Support Technologies

Recently, a group of technologies that allow for the execution of legacy code on the client side, in a Web browser, became available. We refer to these technologies as WMSTs, because they can be used as a basis for client-side migration of legacy code to the Web. To assess potential WMSTs, a comparative experiment was conducted based on

a medical appointment scheduling scenario application with the characteristics described in table 2.1. As shown in table 6.1, we consider seven candidate WMSTs that allow executing legacy C/C++ code in the browser. They can be grouped into compilers producing JavaScript, runtimes, browser plugins, and assembly languages.

Table 6.1: Web Migration Support Technology Groups and Implementations

Group	Technology
Compilers producing JavaScript	emscripten (Zakai, 2011) cheerp ⁹³
Runtimes	Google Native Client ⁹⁴
Browser Plugin APIs	NPAPI ⁹⁵ PPAPI ⁹⁶ js-ctypes ⁹⁷
Assembly Languages	WebAssembly (W3C, 2019)

Following the assessment process in fig. 6.5, candidate technologies that had not reached the end of support were applied for rapid migration prototyping of the scenario application.

⁹³<http://www.leaningtech.com/cheerp/> Retrieved: 6.12.2019

⁹⁴<https://developer.chrome.com/native-client> Retrieved: 6.12.2019

⁹⁵<https://developer.mozilla.org/en-US/docs/Plugins/Guide> Retrieved: 6.12.2019

⁹⁶https://developer.chrome.com/native-client/pepper_stable Retrieved: 6.12.2019

⁹⁷<https://developer.mozilla.org/en-US/docs/Mozilla/js-ctypes> Retrieved: 6.12.2019

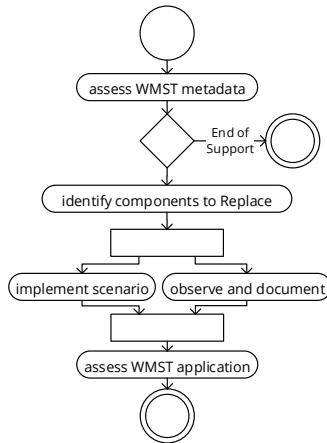


Figure 6.5: Web Migration Technology Assessment Process

The observations provided input for the following five assessment parameters:

- Migration effort
- Limitations
- Platform and OS support
- Currentness of the technology
- Prevalence and supporting partners

According to our experiment, WebAssembly (WASM) is the most promising WMST. WASM (W3C, 2019) is an open Web standard (cf. principle P1) currently designed by the W3C WebAssembly Community Group as a format for compilation to the Web aiming at portability and size- and load-time-efficiency. The Community Group includes all major browser-developing companies: Google, Mozilla, Apple, and Microsoft. WASM combines the experience from previous work on emscripten and Native Client. It defines both a binary format optimized for fast execution and a textual format for pretty-printing for human source readers. Conver-

sion between the two is possible using the WebAssembly Binary Toolkit. Using parts of the emscripten compiler toolchain, C/C++ code can be compiled to WASM instead of asm.js⁹⁸. WASM supports dynamic linking for loading DLL dependencies at runtime. WASM has reached the WebAssembly Consensus milestone⁹⁹ with implementation of version 1.0 available in all major browsers. Figure 6.6 shows an overview of WASM.

In the experiment, WASM showed the fewest limitations, is compatible with all major platforms, is the most current and actively developed technology, and sees comprehensive support from major companies due to being an open standard. Thus, AWSRM realizes client-side reuse of business logic in action 1 based on WASM, as described in the next subsection.

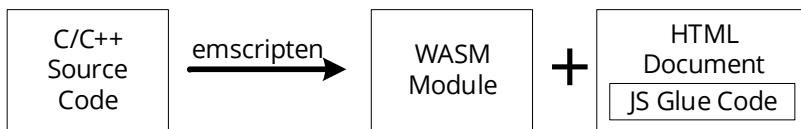


Figure 6.6: WebAssembly Overview¹⁰⁰

6.4.2 Rapid Web Migration Prototyping leveraging WebAssembly

ReWaMP (Heil, Siegert, et al., 2018) specifies a process and toolchain for Rapid Web Migration Prototyping enabled through client-side reuse of business logic leveraging WASM. This section provides an overview of the architecture of ReWaMP prototypes, shown in fig. 6.7, and the

⁹⁸<http://asmjs.org/spec/latest> Retrieved: 6.12.2019

⁹⁹<https://webassembly.org/roadmap/> Retrieved: 27.11.2019

¹⁰⁰adapted from <https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts>

process model of ReWaMP, shown in fig. 6.8. The implementation of this process, the supporting toolchain, and the guided Prototyping assistance system are described in section 6.4.3.

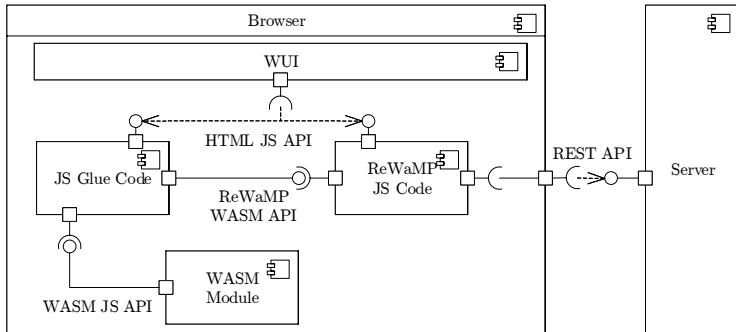


Figure 6.7: Architecture of ReWaMP Prototypes
(adapted from Heil, Siegert, et al., 2018)

One ReWaMP prototype is created per user interface $u \in U$. ReWaMP prototypes (Heil, Siegert, et al., 2018) consist of the following components: The *Web User Interface (WUI)* is a set of HTML/CSS definitions of layout and content structure as created in action 2 of fig. 6.1. The *WASM Module* is created through the compilation of the legacy business logic to WASM, as shown in fig. 6.6. For each legacy user interface, the corresponding *WASM Module* contains all UI functionality and semantics. For communication with other components, it is bound to the *JS glue code*. The *JS Glue Code* is created during the compilation of legacy code to WASM by the emscripten compiler. It realizes the *JS API* of WASM, providing standard functionalities like memory management and serialization of strings for message exchange. The *ReWaMP JS Code* contains infrastructure for communication with the server and abstraction of the generated WUI. Thus, it provides Web-Migration-Prototyping-specific API extensions for the *JS API* of WASM required to control the behavior from legacy code compiled to WASM.

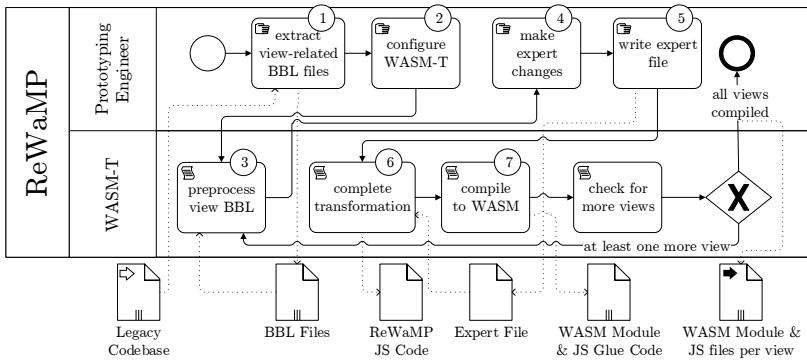


Figure 6.8: ReWaMP Process (adapted from Heil, Siegert, et al., 2018)

The ReWaMP process (Heil, Siegert, et al., 2018) in fig. 6.8 produces a ReWaMP prototype by creating the WASM Module, JS Glue Code and ReWaMP JS Code for each selected user interface $u \in U$ – in the following these top-level UI container elements without parents are referred to as *view*, as they correspond to views in the MVC pattern. This is done by extraction of relevant information from the legacy codebase B , Transformation into a WASM-compilable version, and compilation to the WASM target using emscripten. To support the rapid creation of WASM Modules, the *WASM Transformator* (*WASM-T*) provides infrastructure for C++-codebases using MFC as in the scenario in section 2.1.2. The responsibilities of the Prototyping Engineer and the WASM Transformator are represented as separate lanes in fig. 6.8.

The Prototyping Engineer (1) extracts the source files $f \in B_u, B_u \subset B$ containing view-related behavioral business logic from the legacy codebase B , and (2) provides the WASM Transformator with the file location and the main BBL file $f^* \in B_u$ per view. Like main functions in C++, *main BBL files* f^* are those files that directly interact with the view u , defining event handlers for UI elements. WASM-T transforms

the extracted legacy code to make it compilable with WebAssembly. It uses a semi-automatic process since automated semantic analysis of source code is complex, and the behavioral business logic in $f_i \in B_u$ can reference dependencies $dep_j \in Dep$, i.e. $D_{i,j} = 1$, which are not available for compilation because they are binaries (KDM: *BinaryFile*). In (3), the WASM Transformator pre-processes the behavioral business logic by deleting/rewriting GUI framework-specific segments $s_{fs} \in f$ and extracting UI coupling information for introducing the Web UI.

Then, the Prototyping Engineer re-engineers segments $s^* \in f$ which WASM-T was not able to transform, either through *expert changes* in situ (4) or within a separate *expert file* (5). Expert changes replace or remove complex constructs to resolve missing dependencies. The expert file contains missing declarations and definitions of code items (KDM: *CodeItem*) such as classes, variables, and functions.

The required information can be found in the related BBL files $f \in B_u$ from step 1. The Prototyping Engineer *mocks*¹⁰¹ classes and functions, as the classes require only DTO-like (data transfer object) versions, and the bodies of server-side functions are filled by WASM-T in (6). WASM-T completes the Transformation (6) through the generation of function bodies in the expert files and ReWaMP JS code for communication support between WASM and WUI and WASM and the server side, respectively. Finally, the code is compiled to WASM (7).

¹⁰¹cf. *mock object* (ISO/IEEE, 2017b)

6.4.3 Rapid Web Migration Prototyping Toolchain

Implementation of ReWaMP based on WASM introduces four technical challenges, which are briefly outlined in the following. *WASM data serialization*, owing to its focus on accelerating computation-intensive Web Applications like in-browser games, is a challenge since WASM only supports 32 and 64-bit integers and floats. Passing more complex data like arrays or objects into and out of the WASM Module must be implemented via memory allocation (heap-based) and passing of pointers via JS Glue Code. For strings, JS Glue Code provides a UTF API. ReWaMP uses this API to serialize any non-numerical data to UTF strings in JSON. *Client/Server Communication* between WASM Module and REST API is implemented via the ReWaMP JS Code transparently: it handles the data serialization, communication via AJAX, and object instantiations based on responses to the WASM Module as synchronous method invocation. *DOM Access* from WASM Module for reading input data from the WUI and changing the UI state is supported by ReWaMP JS Code similar to the Client/Server communication mechanism. *UI event handling* is provided by the ReWaMP runtime by forwarding events to automatically exported methods of the WASM Module within ReWaMP JS Code.

Figure 6.9 shows the architecture of the WASM Transformator, which provides a toolchain supporting the ReWaMP process. WASM-T is implemented as command-line tool in python and uses the python bindings of libclang¹⁰² for parsing the legacy source code.

The main components are ReWaMP processor and the C++ parser. The *configuration* interacts with the Prototyping Engineer as in step 2 in fig. 6.8 and stores the list of all main and related BBL files and the tool paths of libclang and emscripten.

¹⁰²<https://clang.llvm.org/docs/Tooling.html> Retrieved: 6.12.2019

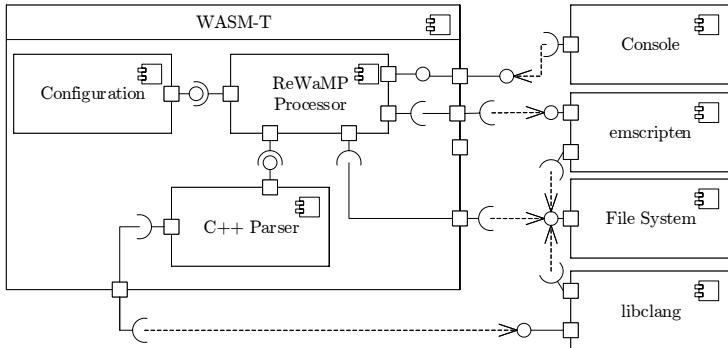


Figure 6.9: WASM Transformer Architecture

The *C++ parser* component parses the codefiles $f \in B$ using libclang to identify segments $s \in f$ for Transformation and provides a custom interface for the ReWaMP processor. The clang parser creates an AST representation of f , which is traversed to identify segments s representing program elements (KDM: *CodeItem*) like includes, variables, functions, classes, as shown in fig. E.1 and provide the ReWaMP processor with this information.

The *ReWaMP processor* is the core component of WASM-T, implementing the code Transformation. It is controlled via a console interface and uses the emscripten compiler to compile all related BBL files to the WASM Module. Algorithm 6.1 shows the pseudo code of the algorithmic realization of WASM-T Transformation in the ReWaMP processor. It implements steps 3 (line 4-12), 6 (line 15-30) and 7 (line 31) of fig. 6.8. *BBL preprocessing* manages the includes in related files $f \in B_u$ by replacing GUI framework-specific types, function calls and extracts message maps representing module communication. Following the manual steps of the Prototyping engineer, for *Transformation completion* the ReWaMP processor fills function body stubs with generated code for

Algorithm 6.1: ReWaMP Process Algorithm

Input: Configuration Cfg, view behavior files B_u per view u
Output: ReWaMP runtime for each view u

```
1 function ReWaMPPProcess()
2    $B_u = \text{readAllFilesInDir}(\text{Cfg.workingDir})$ 
3   foreach  $f^*$  in  $\text{Cfg.mainFiles}$  do
4      $cursor_{now} = \text{getMainCursor}(f^* \text{ in } \text{Cfg.workingDir},$ 
      True)
5      $B'_u = \text{getRelatedFiles}(cursor_{now}, B_u)$ 
6      $B'_u = \text{editIncludings}(cursor_{now}, B'_u, \text{Cfg.headers})$ 
7      $B'_u, maps = \text{minorChanges}(B'_u)$ 
8      $dir_{rel} = \text{createRelatedDir}(f^*)$ 
9      $cursor_{now} = \text{saveRelFilesGetCursor}(f^* \text{ in } dir_{rel}, B'_u)$ 
10     $B'_u = \text{deleteUnusedParams}(cursor_{now}, B'_u)$ 
11     $B'_u, bodies, funcs = \text{editMFCFunctions}(cursor_{now}, B'_u)$ 
12     $\text{saveRelatedFiles}(B'_u)$ 
13     $\text{printHints}(cursor_{now}, B'_u)$ 
14     $\text{waitForME}()$ 
15     $B'_u = \text{loadRelatedFiles}()$ 
16     $cursor_{now} = \text{getMainCursor}(f^* \text{ in } dir_{rel})$ 
17     $bodies = \text{buildExpertFunctions}(cursor_{now}, B'_u, bodies)$ 
18     $B'_u = \text{addJSONMethods}(cursor_{now}, B'_u)$ 
19     $B'_u = \text{processFlags}(B'_u)$ 
20     $cursor_{now} = \text{saveRelFilesGetCursor}(f^* \text{ in } dir_{rel}, B'_u)$ 
21     $embinds, funcs = \text{collectEmbinds}(cursor_{now}, B'_u,$ 
      maps, funcs)
22     $cursor_{now} = \text{saveRelFilesGetCursor}(f^* \text{ in } dir_{rel}, B'_u)$ 
23     $B'_u = \text{addFunctions}(cursor_{now}, B'_u, funcs)$ 
24     $cursor_{now} = \text{saveRelFilesGetCursor}(f^* \text{ in } dir_{rel}, B'_u)$ 
25     $B'_u = \text{writeNewBodies}(cursor_{now}, B'_u, bodies)$ 
26     $B'_u = \text{addSendStatusFunction}(B'_u)$ 
27     $B'_u = \text{addEmbinds}(embinds, B'_u)$ 
28     $\text{saveRelatedFiles}(B'_u)$ 
29     $\text{copyLibraryFiles}(dir_{rel})$ 
30     $\text{addJSInitFunction}(embinds, maps)$ 
31     $\text{compileFilesToWasm}(B'_u)$ 
```

calling functionality on the server side, serialization, and to expose code for the UI event handling from JavaScript. The WASM-T support of semi-automatic Transformation is based on the technical assumptions listed in table E.1. Listing 6.1 shows an example of source code generated by the ReWaMP processor. This code handles the passing of return values between a legacy function in the WASM-compiled C++ code and JavaScript code. It is implemented with the *Inline JavaScript* EM_ASM() method of emscripten. The example shows the passing of a string through UTF8 serialization, memory allocation, and management of the WASM heap using a char array pointer.

Listing 6.1: Example of Code Generated by ReWaMP to Handle Passing of Return Value from Function Call

```
std::string rs = (char *) EM_ASM_INT({
    var data_string = UTF8ToString($0);
    var return_string = send_server_function(
        ↳ data_string);
    var lengthBytes = lengthBytesUTF8(return_string)
        ↳ + 1;
    var stringOnWasmHeap = _malloc(lengthBytes);
    stringToUTF8(return_string, stringOnWasmHeap,
        lengthBytes + 1);
    return stringOnWasmHeap;
}, cha);
```

6.4.4 Guided Web Migration Prototyping Assistance

To achieve the limited resources expertise constraint of RO2, AWSM:RM provides the *Rapid Web Migration Prototyping Assistance (RWMPA)*, an assistance system that guides Prototyping Engineers through the ReWaMP process, providing an answer to AWSM:RM RQ3. The need for providing guidance was identified in initial experiments with ReWaMP

and its toolchain WASM-T (cf. section 6.6.2) due to the complexity of the ReWaMP process and the observed demand for improved tool support compared to the console-based interaction with WASM-T and external editor-based manual interventions. The resulting guided process is referred to as *RWMPA workflow* in the following. It contains adaptions to the process in fig. 6.8 based on the feedback from experimentation with ReWaMP to implement the process at a higher level of detail for better usability. The detailed RWMPA workflow is presented in fig. E.2. Table 6.2 shows the mapping between the process introduced in fig. 6.8 and the tasks of the adapted workflow. RWMPA is implemented as extensible model-driven Web Application: the basic workflow was modeled in BPMN 2.0 and is instantiated using the Camunda Workflow Engine¹⁰³ on top of which the RWMPA Web Application provides a Web user interface guiding the Prototyping Engineer. In this way, the process can be adapted and extended to specific requirements diverging from the scenario in section 2.1 when employed with different Web Migration approaches.

Table 6.2: ReWaMP tasks and realization in RWMPA workflow. M and T indicate manual tasks and tasks by WASM-T in ReWaMP, U and S designate user and service tasks in RWMPA

Original (ReWaMP) process	Guided (RWMPA) workflow
1M extract view-related BBL files	5S find rel. files, 6U select files automatically, not in workflow
2M configure WASM-T	7S preprocess
3T preprocess view BBL	10U del. code, 11U repl. code, 16U exp. changes
4M expert changes	16U expert changes
5M expert file	17S save
6T complete transformation	

¹⁰³<https://camunda.com/products/bpmn-engine/> Retrieved: 6.12.2019

Original (ReWaMP) process	Guided (RWMPA) workflow
7T compile to WASM	14S compile to WASM

Figure 6.10 shows a screenshot of the start page of the RWMPA system. It provides an overview of the entire workflow in BPMN on top and detailed instructions on how to conduct the Prototyping. The instructions can be re-visited in any step of the guided process. Additionally, each step displays a similar instruction page explaining the use of the RWMPA tools to the Migration Engineer, including screenshots.

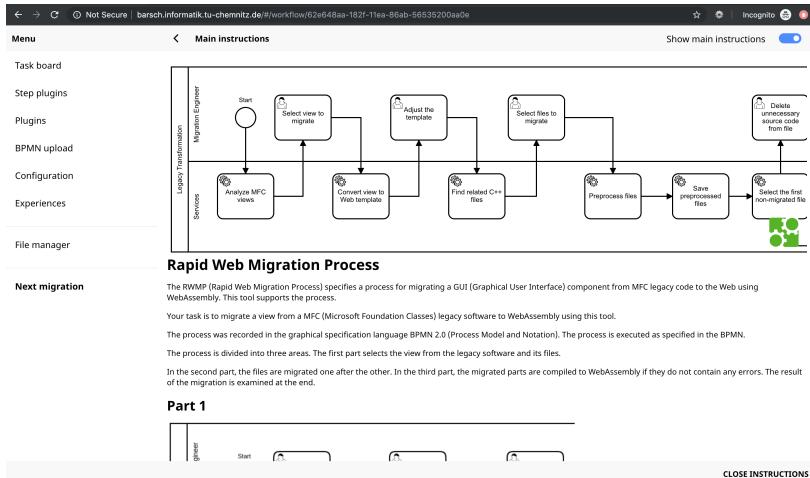


Figure 6.10: RWMPA Workflow Start Page explaining Required Steps

As shown in fig. E.3, RWMPA is designed in a modular style: each step in the workflow is interfaced in the WUI in detail and in the context of the overall process. RWMPA provides a default implementation for the details of each step (*step worker page*) that displays a description of required activities. Via its Plugin Service, specific implementations of work-

flow steps can be registered, consisting of frontend *plugins* and backend *migration services*. RWMPA manages the context of the plugins, e.g. providing the required input and storing the output artifacts and invokes the plugin implementation when the Prototyping Engineer reaches the corresponding step. This allows for extension of RWMPA with additional automation and with rapid Web migration tools with improved usability. The RWMPA frontend is implemented in TypeScript with Angular-based Ionic¹⁰⁴ 4. Plugins are compiled to W3C WebComponents (W3C, 2018) for embedding in a step worker page using on the stencil¹⁰⁵ compiler, RWMPA migration services are implemented in node.js.

RWMPA implements a set of plugins, including file management and source code editing for steps 4 and 5 in fig. 6.8 with syntax highlighting. Figure 6.11 shows a merge view of RWMPA supporting the Prototyping Engineer to review and accept or reject changes from WASM-T preprocessing. The left side shows the code of the original legacy code; the right side shows the code after preprocessing by the WASM Transformator, which tries to resolve dependencies by adding corresponding includes and adapting the types accordingly. The Merge View highlights changes which can be accepted or rejected by the Prototyping engineer, and also allows direct editing in the code editor on the right.

6.5 User Interface Transformation

To demonstrate a prototypical Web-based version of the Legacy System for RO2, a Web user interface similar to the legacy user interface needs to be created. AWSM:RM action 2 in fig. 6.1 defines this creation as semi-automatic Transformation of the Legacy System to meet the resources and expertise constraint of RO2 and answering the demand

¹⁰⁴<https://ionicframework.com/> Retrieved: 6.12.2019

¹⁰⁵<https://stenciljs.com/> Retrieved: 6.12.2019

The screenshot shows a browser window with two tabs open. The left tab is titled 'Original file' and contains the following C++ code:

```
1 // Date_View2.cpp : Implementierung des Datei-Dialogfelds
2 //
3 // This source file is part of the Microsoft .NET Framework Sample.
4 // Copyright (C) Microsoft Corporation. All rights reserved.
5 // This source code is licensed under the Microsoft Permissive License.
6 // See LICENSE.txt for details.
7 // 
```

The right tab is titled 'Processed Date_View2.cpp' and contains the following C++ code:

```
1 // Date_View2.cpp : Implementierung des Datei-Dialogfelds
2 //
3 // This source file is part of the Microsoft .NET Framework Sample.
4 // Copyright (C) Microsoft Corporation. All rights reserved.
5 // This source code is licensed under the Microsoft Permissive License.
6 // See LICENSE.txt for details.
7 // 
```

A large blue arrow points from the 'Original file' tab to the 'Processed Date_View2.cpp' tab, indicating that the processed code is identical to the original.

Figure 6.11: RWMPA Step Worker Page Example: Merge View

for automation of AWSM:RMRQ3. This section presents the realization of the user interface transformation of action 2. Section 6.5.1 outlines the conceptual problem of mapping legacy to Web layouts, section 6.5.2 defines a process for computing the mapping based on evolutionary optimization, and section 6.5.3 describes the implementation of this process in the UI Transformer tool.

6.5.1 Mapping Legacy to Web Layouts

Transformation of a legacy user interface u to a Web-based version is difficult due to conceptual differences in the user interface layouts. Legacy desktop user interfaces are built using a variety of heterogeneous frameworks and a legacy layout l_{legacy} is described as pixel-based mapping of controls c into two-dimensional cartesian space as defined in the formalism in section 4.6.3. Modern Web user interfaces adhere to the *Responsive Web Design* paradigm (Marcotte, 2010; Nebeling and

Norrie, 2013): they dynamically adapt to different viewing environments, i.e. aspect ratios, resolutions, screen sizes etc. This is realized through fluid grids, defining the WUI as *grid layout* l_{grid} consisting of *rows* and *columns*. Implementations of grid layouts are based on popular frameworks like bootstrap¹⁰⁶ or on the CSS Grid Layout standard (W3C, 2017a). Figure 6.12 represents the two different layout paradigms, pixel-based on the left side and grid layout on the right side, which need to be mapped for the Transformation.

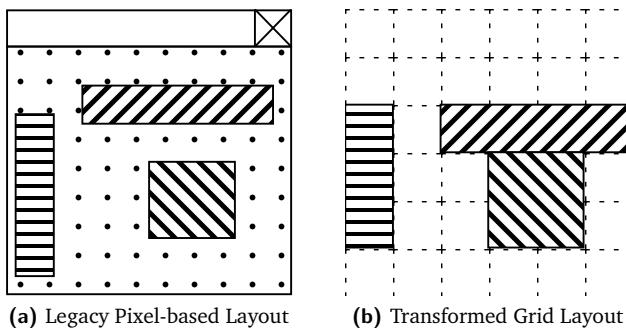


Figure 6.12: Layout Transformation: Mapping Pixel-based to Grid Layouts

Thus, realizing action 2 of AWSM:RM as shown in fig. 6.1 requires a mapping from pixel-based to grid layouts. Finding this mapping is an *optimization problem* with discrete variables: it tries to minimize an *objective function* under a set of *hard and soft constraints*. The objective function is a similarity measure. The discrete variables are the possible positions and sizes in the grid layout, and the constraints represent the nature of grid layouts, as outlined in the following.

Compared to pixel-based layouts, the rows and columns of grid layouts divide the continuous Euclidean plane into discrete *cells*, thus imposing

¹⁰⁶<https://getbootstrap.com/> Retrieved: 6.12.2019

constraints on the placement of controls. A grid layout can have an infinite number of rows, but it has a fixed number of columns $n_{grid} \in \mathbb{N}^+$, distributed equally within the width of the viewport $w_{viewport}$. A control c is assigned to one or more cells.

The bounding box $b = (x, y, w, h)$ defined in eq. (4.6) of a control c in l_{grid} reflects these grid constraints as follows:

$$\begin{aligned} x &\in \left\{ i \cdot \frac{w_{viewport}}{n_{grid}} \mid 0 \leq i < n_{grid} \right\} & y &\in \{j \cdot h_{row} \mid j \in \mathbb{N}_0\} \\ w &\in \left\{ k \cdot \frac{w_{viewport}}{n_{grid}} \mid 1 \leq k < n_{grid} - x \right\} & h &\in \{l \cdot h_{row} \mid l \in \mathbb{N}\} \end{aligned} \quad (6.1)$$

Valid bounding boxes have horizontal positions x that are a multiple of the column width $\frac{w_{viewport}}{n_{grid}}$, vertical positions y that are a multiple of the row height h_{row} , a width w as multiple of column width that ensures that c is assigned to at least one column and that prevents a row overflow by ensuring the constraint $x + w \leq w_{viewport}$ and a height h a multiple of h_{row} that ensures that c is assigned to at least one row. L_{grid} is the set of all valid grid layouts.

6.5.2 User Interface Transformation through Evolutionary Optimisation

Due to the limited resources in RO2, layout Transformation needs to be automated. This requires solving the optimization problem of mapping a given pixel-based l_{legacy} to a grid layout l_{grid} in the constrained

grid-cell space through an algorithm. For AWSM:RM, the optimization is implemented as *evolutionary algorithm* (Weise, 2009). Figure 6.13 shows the process of UI Transformation using evolutionary optimization.

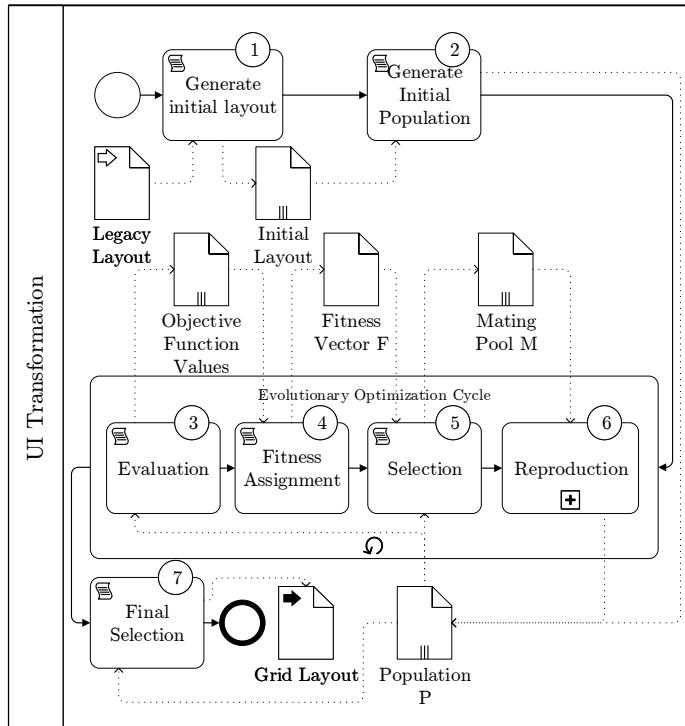


Figure 6.13: UI Transformation Process

The algorithm tries to optimize the characteristics by creating new *individuals* – grid layouts – from existing ones. Based on an *initial population* $P_0 \subset L_{grid}$, a cycle of *evaluation*, *fitness assignment*, *selection*, and *reproduction* is repeated, incrementally creating new populations of grid layouts P . The process ends by selecting an individual p from the last population. An individual is defined by its *genotype* $p.g$, which

defines the optimization's *search space* and its *phenotype* $p.x$, which defines the optimization's *problem space* in which solutions of the optimization are contained. Genotype $p.g$ is an instance $l_{grid} \in L_{grid}$. As all information of l_{grid} is observable, genotype and phenotype are identical $p.x = p.g$ and thus populations are $P \subset L_{grid}$. The steps of the UI Transformation process are defined as follows.

Initial population P_0 is created in steps 1 and 2 by calculating an initial grid layout from mapping legacy bounding boxes $b_l = (x_l, y_l, w_l, h_l)$ onto grid bounding boxes $b_g = (x_g, y_g, w_g, h_g)$ with the following *index approximations*:

$$\begin{aligned} x_g &= \left\lfloor \frac{x_l \cdot n_{grid}}{w_{viewport}} \right\rfloor & y_g &= \left\lfloor \frac{y_l}{h_{row}} \right\rfloor \\ w_g &= \max \left(1, \left\lfloor \frac{w_l \cdot n_{grid}}{w_{viewport}} \right\rfloor \right) & h_g &= \max \left(1, \left\lfloor \frac{h_l}{h_{row}} \right\rfloor \right) \end{aligned} \quad (6.2)$$

Evaluation (3) of individuals p is calculated according to a set of objective functions f , the resulting vector $F(p.x)$ is used for the calculation of the fitness function $v(p.x)$, mapping the objective vector to \mathbb{R} . This multi-objective optimization follows *Pareto optimality*: To calculate the **fitness** (4) of individual $p \in P$, let

$$P^* = \{p_i \in P \setminus \{p\} \mid f_k(p_i) \leq f_k(p) \forall f_k \in F\} \quad (6.3)$$

be the set of all individuals $p_i \neq p$ that are dominated by p , then the fitness of p is

$$v(p.x) = |P^*| \quad (6.4)$$

defined by the number of individuals dominated by p (Weise, 2009). Ordering P by this fitness value, Pareto-optimal solutions are put first in the ordered population.

Selection (5) is realized as *truncated selection*: A subset of the ordered population $M \subset P$ forms the *mating pool*, which comprises the fittest $|M| = 0.5 \cdot |P|$ individuals.

Reproduction (6) is realized with the individuals of M , producing the new generation of the population P_{next} through search operations identifying new genotypes. For any pair $p_1, p_2 \in M$, the following search operations can be applied: *duplication* by adding p_1 to P_{next} , *mutation* by modifying a random characteristic of $p_1.g$ and adding the resulting p'_1 to P_{next} , *recombination* by creating a new individual p_3 with characteristics from p_1 and p_2 . Recombination occurs with probability P_r , mutation with P_m , independently, i.e. a new individual created through recombination can also be mutated. Else duplication occurs. Mutation is realized as one incremental change of one parameter of the bounding box b_{grid} of one randomly selected control $c \in p_1$, i.e. movement by one column or row, increase/decrease of size. Recombination assigns vertical position and size of corresponding controls in p_2 to p_1 . The recombination selects individuals based on the Neighborhood Cultivation Genetic Algorithm (NCGA) (S. Watanabe et al., 2002), which defines an ordering of M .

according to one $f_k \in F$ selected round-robin. This makes genotypes more similar to their parents, which puts more emphasis on searching around possible solutions compared to random selection of individuals.

Final selection (7) of an optimized grid layout is computed after reaching the maximum $n_{generations}$ repetitions of the optimization cycle from the final generation P_{last} based on F :

$$p_{opt} = \arg \max_{p \in P_{last}} F(p.x) \quad (6.5)$$

6.5.3 User Interface Transformation Tool

To apply the evolutionary algorithm described above to concrete legacy user interfaces and generate grid-based Web user interfaces, the *UI Transformer* tool implements the process presented in fig. 6.13 as a model-driven Reengineering process, achieved through model-to-model Transformation between two CIMs, as shown in fig. 6.14. In addition to the conceptual challenge of mapping layouts described in the previous subsection, UI Transformer also solves the technical challenges of layout analysis and generation in order to extract the required input information for the evolutionary algorithm and generate concrete user interfaces from its results.

UI Transformer supports conversion of pixel-based legacy layouts based on the MFC framework (cf. section 2.1) to responsive fluid grid layouts based on bootstrap. It is implemented in Python 3 with a C implementation of computationally complex Transformation and objective functions evaluation. UI Transformer transforms legacy layouts in three stages

as shown in fig. 6.14: Reverse Engineering as CIM creation, restructuring as model-to-model Transformation between two CIMs, Forward Engineering as model-to-text Transformation from CIM to PSM

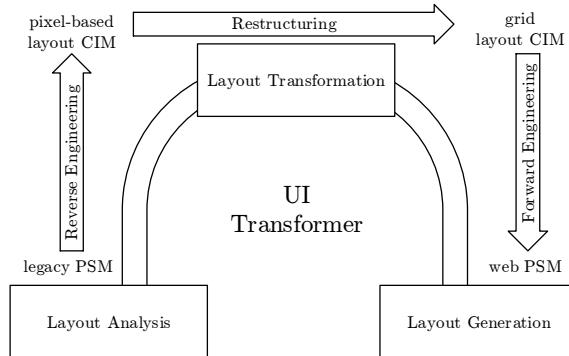


Figure 6.14: UI Transformer as model-driven reengineering process

Layout Analysis. The first stage of UI Transformer extracts the required information according to the CIM-level formalisms introduced in section 4.6.3 from the input artifacts, i.e. the files $f \in B_u$, $B_u \subseteq B$ describing user interface $u \in U$ of \mathfrak{L} . In the context of the MFC framework used in the scenario, these files are *Resource Files*¹⁰⁷ containing definitions of various application resources. Using a tool like *Resource Hacker*¹⁰⁸ they can also be retrieved from executables in E . Descriptions of user interfaces u are expressed in *dialog* elements. Listing 6.2 shows an example of an MFC resource file describing a dialog. As seen, position and overall viewport of the dialog are stated in the first line, and the three control elements are positioned and sized in pixel units. UI Transformer parses the resource definitions to extract viewport, controls, types, positions, and sizes. To determine the *hierarchy* of controls, which occur

¹⁰⁷<https://docs.microsoft.com/en-us/cpp/ide/resource-files-cpp> Retrieved: 6.12.2019

¹⁰⁸ <http://www.angusj.com/resourcehacker/> Retrieved: 6.12.2019

non-nested in resource files, bounding boxes are tested for complete containment, identifying container and nested container elements, i.e. the parent controls c_p . The result of layout analysis is an instance l_{legacy} .

Listing 6.2: Example of a User Interface Description in MFC Resource File
(shortened)

```
1 DIALOGEX 0, 0, 289, 90
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION |
    ↳ WS_SYSMENU
CAPTION "Appointment_Search"
FONT 8, "Tahoma"
{
    ...
CONTROL "", 6205, COMBOBOX, CBS_DROPDOWNLIST |
    ↳ CBS_SORT | WS_CHILD | WS_VISIBLE |
    ↳ WS_VSCROLL | WS_TABSTOP,
    7, 71, 152, 30
CONTROL "Find", 1, BUTTON, BS_DEFPUSHBUTTON |
    ↳ WS_CHILD | WS_VISIBLE | WS_TABSTOP, 171,
    ↳ 69, 50, 14
CONTROL "Cancel", 2, BUTTON, BS_PUSHBUTTON |
    ↳ WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232,
    ↳ 69, 50, 14
    ...
}
```

Layout Transformation. Layout Transformation takes l_{legacy} as input and implements the optimization algorithm in fig. 6.13. An extract of the implementation is shown in listing 6.3, showing the creation of the initial population and subsequent optimization through the evolutionary optimizer. Listing 6.4 shows the implementation of mutation of a single layout individual and the crossover of two layouts for the

generation of new populations during optimization. The Transformation is computationally complex due to the complexity of the search space \mathbb{G} with increasing numbers of controls. Therefore, parallel programming was employed to leverage multi-core processors: Transformation is run in parallel for several layout instances $l_{\text{legacy},i}$ and the calculation of several objective functions f_k on one l_{legacy} is also run in parallel. Our experiments have shown that *processes* are required, since *threads* did not improve performance due to the *Global Interpreter Lock*. Objective functions f_k are calculated based *measurements* so that they can reuse measurement results.

The *Levenshtein distance* (Black, 2019), used to identify variations in the order of controls $c \in u$, is a computationally complex measure. Therefore, a C-based implementation¹⁰⁹ was used for Levenshtein distance calculation.

Listing 6.3: Layout Transformation

```
def transform_and_generate_single(self, input_file,
    legacy_layout, silent=True):
    transformer = transformations.
        ↪ SimpleLayoutTransformation()
    initial_grid_layout = transformer.transform(
        ↪ legacy_layout)
    optimizer = EvolutionaryOptimizer(self.
        ↪ _target_functions,
        legacy_layout, population_seed=
            ↪ initial_grid_layout)
    optimized_grid_layout = optimizer.optimize(
        self._optimizer_iterations, silent=silent)
    ...
    ...
```

¹⁰⁹<https://github.com/ztane/python-Levenshtein/> Retrieved: 6.12.2019

Listing 6.4: Mutation and Crossover

```
def mutate_individual(self, individual,
    ↪ all_controls=False):
    control_count = len(individual.controls)
    control_index = self.randint(0, control_count -
        ↪ 1)
    control = individual.controls[control_index]
    mutations = [self._mutate_control_xpos,
        self._mutate_control_ypos, self.
            ↪ _mutate_control_width,
        self._mutate_control_height]
    numpy.random.shuffle(mutations)
    ...
    return new_individual

def crossover_individuals(self, individual1,
    ↪ individual2):
    new_individual = GridLayoutModel(self.
        ↪ _input_layout.name,
        self._input_layout.title)
    controls = self.join(individual1, individual2)
    for (c1, c2) in controls:
        c = copy.deepcopy(c1)
        c.rc.v_start = c2.rc.v_start
        c.rc.v_span = c2.rc.v_span
        new_individual.add_control(c)
    return new_individual
```

Objective Functions $f_k : L_{\text{legacy}} \times L_{\text{grid}} \mapsto \mathbb{R}_0^+$ compare a legacy layout instance with a grid layout based on the calculation of measurements on both layouts and a calculation to aggregate the two measurements into a non-negative real number. The Transformation aims at creating grid-based layouts that represent the legacy layout. Similarity of layouts is a phenomenon highly related to human perception, not yet very well understood and thus an active field of research (cf. also chapter 7). Evolutionary optimization, however, needs a basis for the calculation of fitness. The objective functions, therefore, represent a set of features that can be automatically calculated from the layout instances without human intervention for fitness calculation. Five objective functions are implemented: Whitespace Ratio, Control Density, Order of Controls, Edge Alignment, and Area Loss. Whitespace Ratio f_W sums the area covered by all visible controls and determines the ratio between this sum and the entire viewport area. Control Density is a local spatial feature calculated based on Gini coefficient $f_{D,G}$ and Herfindahl Index $f_{D,H}$. Order of Controls f_O counts changes in order by mapping controls to strings, determining sequences in 8 different orientations (left to right top to bottom, bottom to top right to left, etc.) and calculating the Levenshtein distance of these sequences. Edge Alignment f_A identifies aligned vertical and horizontal edges of controls and counts alignment violations in the generated layout. Area Loss f_L compares bounding box sizes of controls in both layouts and sums up the differences.

A prototypical alternative implementation of the Transformation in C exists, comprising the evolutionary algorithm and the objective function Order of Controls, which is part of experimentation related to performance aspects. The C-based optimizer integrates

with the core python implementation via runtime loading of the library using ctypes¹¹⁰. Figure 6.15 shows an example of a legacy layout and the resulting transformed version.

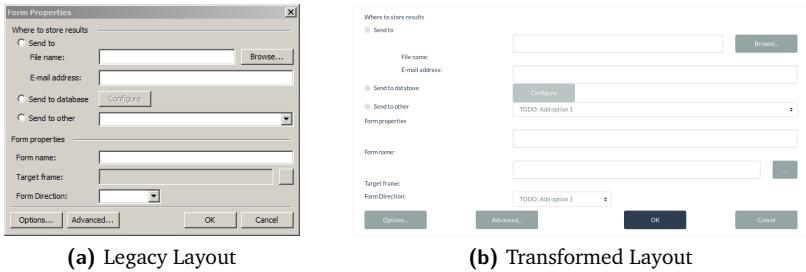


Figure 6.15: Layout Transformation Sample

Layout Generation. The third stage of UI Transformer is a model-to-text generation, taking the $l_{grid,opt}$ CIM resulting from the final selection step in fig. 6.13 and generating the PSM of the WUI, i.e. an HTML file with corresponding CSS directives, which can be rendered in a Web browser. Layouts are generated using bootstrap, as older browsers do not support CSS Grid. Field research in section 2.2.1, however, showed that the ISV customers do not frequently use current operating systems and browsers. In particular, Internet Explorer 11, does not support CSS Grids.

The HTML output is generated using the Mako¹¹¹ template engine, allowing to separate design and implementation to facilitate the adaption of the output structure without changes to the generation code. Generation is based on the hierarchy of controls as represented in the c_p parent relationships of the CIM since nested sub-grids can occur. Controls c_1, c_2 with a common parent element $c_{1,p} = c_{2,p}$ are grouped by the parent

¹¹⁰<https://docs.python.org/3/library/ctypes.html> Retrieved: 6.12.2019

¹¹¹<https://www.makotemplates.org/> Retrieved: 6.12.2019

row and then ordered by column. Iteration over the hierarchy starts with root-level elements (having $c_p = 0$) based on a pointer indicating the current position within the grid for cell offset calculation. The text is generated using the template engine, line by line and cell by cell and recursion for each container element, creating a sub-grid. Representation of controls is based on a mapping from MFC controls to HTML elements. A placeholder is generated for controls without HTML equivalent (e.g. scrollbar), which can be implemented as custom element using WebComponents. Listing 6.5 shows the implementation of the layout generation, filling the cells and rows in a bootstrap grid layout.

Listing 6.5: Layout Generation

```
def generate_grid(self, layout, parent_id=None):
    filtered_controls = layout.controls_by_parent(
        ↪ parent_id)
    o_controls = SortedDict()
    for c in filtered_controls:
        row_key = c.rc.v_start
        if row_key not in o_controls:
            controls_in_cell = SortedList(
                [c], key=lambda x: x.rc.h_span * x.rc.v_span
            )
            cells_in_row = SortedDict()
            cells_in_row[c.rc.h_start] = controls_in_cell
            o_controls[row_key] = cells_in_row
        else:
            column_key = c.rc.h_start
            if column_key not in o_controls[row_key]:
                controls_in_cell = SortedList(
                    [c], key=lambda x: x.rc.h_span * x.rc.v_span
                )
```

```

o_controls[row_key][column_key] =
    ↪ controls_in_cell
else:
    o_controls[row_key][column_key].add(c)
self._current_position = Rectangle(0, 0, None,
    ↪ None)
r_code = []
for row in o_controls.keys():
    self._current_position.h_start = 0
    self._current_position.v_start = row
    assert len(o_controls[row]) > 0
    r_code.append(self.generate_grid_row(
        layout, o_controls[row]
    ))
layout_text = '\n'.join(r_code) if len(r_code) > 0
    ↪ else ''
page_template = templ_lookup.get_template('/grid.
    ↪ html')
output_text = page_template.render(grid_rows=
    ↪ layout_text)
return output_text

```

6.6 Evaluation

This section evaluates AWSM:RM regarding different aspects. Section 6.6.1 evaluates AWSM:RM against the requirements in section 6.1.2. Three additional experimental evaluations further address the effectiveness, efficiency, and expertise in section 6.6.2, section 6.6.3, and section 6.6.4. Section 6.6.5 revisits research objective RO2 in the light of the evaluation results.

6.6.1 Assessment of Requirements

This section reports on satisfaction of the requirements for AWSM:RM.

Effectiveness. The effectiveness of AWSM:RM is achieved through specification of a Web Migration prototyping technique that creates concrete, tangible web migration prototypes which can be used to demonstrate desirability and feasibility of Web Migration. A detailed evaluation of results quality is presented in the following evaluation experiments.

Efficiency. The efficiency of AWSM:RM is achieved through reuse of business logic via a semi-automatic WebAssembly-based compilation process and runtime environment and an automatic model-driven Transformation process converting legacy pixel-based user interfaces into grid-based Web user interfaces. A detailed evaluation showing the results achievable with limited and no manual interventions is presented in the following evaluation experiments, reporting on time measurements.

Expertise. The expertise requirement is addressed in AWM:RM in two ways: the business logic reuse process focuses on the existing expertise of the Prototyping Engineer in the legacy platform and lowers the required Web Engineering and migration expertise requirements by providing a semi-automatic process with a guidance system including explanations and support tools. The Web Engineering expertise generally required for the creation of Web-based user interfaces is avoided through a fully automatic process without manual interventions. The following evaluation experiments demonstrate the applicability of AWM:RM with test subjects with limited Web Engineering and no Web Migration expertise. AWM:RM meets the expertise requirement as it is a feasible method based on available expertise with limited Web Engineering and no Web Migration expertise.

Reuse. The AWSM:RM method focuses on reuse as key enabler for the rapidness of Web migration prototyping. It reuses both legacy business logic through the WebAssembly-based ReWaMP process and legacy user interfaces through the automatic creation of WUIs based on an automated model-driven Transformation abstracting from the legacy UI PSM to a pixel-based layout CIM, transforming it into a grid-based layout CIM and generating the WUI PSM. AWSM:RM fulfills the reuse requirement as it is based on reuse on both layers of horizontal prototypes: the user interface and the business logic.

Plausibility. The web migration prototypes resulting from AWSM:RM are Web Applications using the standard Web technology stack of HTML, CSS, and JavaScript in addition to the new WebAssembly standard. Unlike wrapper approaches, they run entirely independent of the Legacy System, and they run in any major Web browser without additional requirements such as extensions, plugins, etc. The architecture of the web migration prototypes represents the major architectural changes of Web Migration: client/server communication, URL-based navigation, encapsulation of the persistence layer behind a RESTful API. AWSM:RM fulfills the plausibility requirements as its prototypes are demonstrative Web Applications representative of all major Web Application characteristics.

6.6.2 Experimental Evaluation of ReWaMP

Experimental evaluation of ReWaMP focuses on the aspects of time and effort for ReWaMP, on required expertise of the Prototyping Engineer, on a detailed understanding of the most prolonged/most difficult tasks in ReWaMP and on possible results from the application of ReWaMP.

Setup. The basis for experimentation is the medical appointment scheduling scenario application, as characterized in section 2.1.2. The scenario codebase B comprises 3373 LOC, of which the *test dataset* $B_T \subset B$ contains the 3 main views $B_T = \{u_{cal}, u_{appointments}, u_{new}\}$

shown in fig. 6.16. The test dataset B_T comprises 937 SLOC defining 35 types, 237 methods, and using 177 third-party elements. Additionally, an executable $e \in E$ of the legacy application was available. The evaluation platform was a Windows 10 Education N x64 running on an Intel i7 930 CPU (2,8GHz), 14 GB RAM.

The experiment was conducted through observed test runs of the ReWaMP process with test subjects. The *test subjects* impersonated the role of Prototyping Engineer and were observed during execution by a *researcher*. Table F.1 shows the guidelines for interaction of the researcher and the test subject. The questionnaire shown in appendix F.1 implemented as Google Form was set up to capture test subject data and subjective perceptions of the ReWaMP process and WASM-T tool support.

Procedure. Each test subject attended a brief introduction to the scenario, its context visualized as fig. 6.1, and the ReWaMP process visualized as fig. 6.8. The executable $e \in E$ of the scenario application was available for the test subjects to explore the functionality. The researcher supervised the test run by setting up the required tools, backing up intermediate results, and providing the same environment state for each step for each test subject, strictly following the guidelines. The test subjects followed the sequence of steps of ReWaMP and performed the manual tasks, extracting BBL files from B , configuring WASM-T, and applying expert interventions (expert changes and expert files). Clion¹¹² was used by the test subjects to navigate the codebase and perform expert interventions; Windows Explorer was used for task 1: one window showing the codebase B_T and one window the working directory of WASM-T. To reduce the time required for one evaluation run, tasks 2, 4 and 5

¹¹²<https://www.jetbrains.com/clion/> Retrieved: 6.12.2019

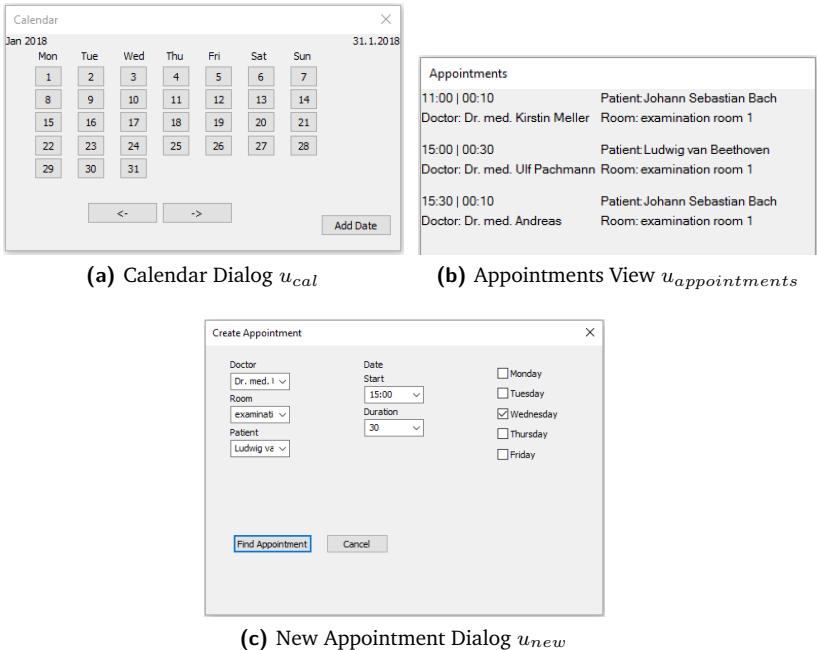


Figure 6.16: Test Dataset B_T Views

were conducted only for the two more complex views u_{cal} and u_{new} . At the end of each test run, the test subject had 5 minutes to familiarise with the migration prototype, before answering the questionnaire.

In addition to the subjective perceptions captured in the questionnaire, objective observations were made through measurements. As Rapid Prototyping implies quick and easy creation of prototypes, time and effort were measured during the experiment for each task.

Time $t = t_M + t_T$ has two main components, t_M the time required by the Prototyping Engineer to perform the manual tasks (1,2,4,5) in

fig. 6.8, and t_T the time required by WASM-T to analyze and transform the legacy code. t_M was measured by stopwatch, t_T was measured programmatically. Time measurement started at the start of manual tasks after preparation of the tools, was interrupted if WASM-T started or resumed processing, and ended by finishing the last expert interventions. t_T is a system environment-dependent measure: on the evaluation platform described above, 87 other processes were active during measurement, and an average load of 13,8% was determined.

Effort evaluates the amount and extent of changes to the existing legacy code required by ReWaMP. Effort can be measured in terms of *code churn*, i.e. the lines of code added/changed/deleted, which has been shown to have a strong linear correlation with effort (Sjoberg et al., 2013). Thus, e_T represents the code churn (SLOC) by WASM-T and e_M the code churn by the Prototyping Engineer. e_M was measured through observation of the test subject and e_T by comparison of inputs and results of one continuous task set completion. All efforts were measured per task and per category (create, update, delete).

Experimental results and descriptive statistics. The experiment was conducted with 6 test subjects (5 male, 1 female), with an age range of 23 – 35 (mean 26), 2-10 years of programming experience (mean 6.33) and Web Engineering expertise (mean 2.83 on Likert scale 1-5). Table 6.3 provides an overview of the time results. The test runs' overall time was between 01:14:27h and 02:06:36h, (mean 01:38:27), which includes processing views as described above. Averaged for one complete process instance of one view, time t was between 00:38:07h and 01:05:13h ($\bar{t} = 00:50:24h$), of which an almost constant ($\sigma = 2s$) time of $\bar{t}_T = 00:00:47h$ was required by WASM-T. The average time for manual tasks was $\bar{t}_M = 00:49:36h$ ($\sigma = 00:10:06h$). Considering time distribution on the tasks shown in fig. 6.17, task 4 (expert

changes) took the longest ($\approx 61\%$ of t), followed by task 1 ($\approx 23\%$ of t). For task 4 and 5, times for u_{cal} ($t4.1, t5.1$) and u_{new} ($t4.2, t5.2$) are shown separately. While task 4 times are close, for task 5 (expert file) u_{cal} was faster than u_{new} .

Table 6.3: ReWaMP Time Measurement Statistics

$\overline{t_M}$	$\overline{t_T}$	$\sigma(t_M)$	$\sigma(t_T)$	$\overline{t1}$	$\overline{t2}$	$\overline{t4}$	$\overline{t5}$
49:36	00:47	10:06	2s	11:22	02:20	30:49	05:06

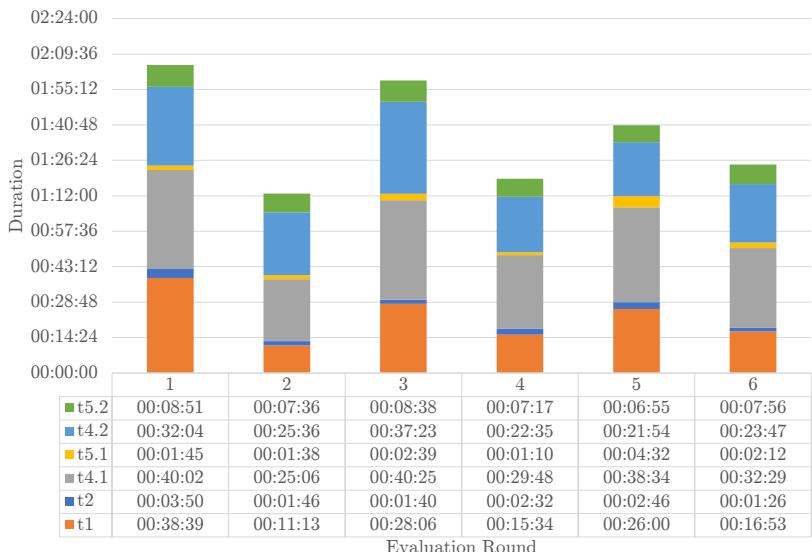


Figure 6.17: Time distributions per task and test run

Table 6.4 provides an overview of the effort results, averaged over the test subjects per on view. Figure 6.18 shows the effort distributions. WASM-T performs almost 3/4 of all required changes (fig. 6.18a). The

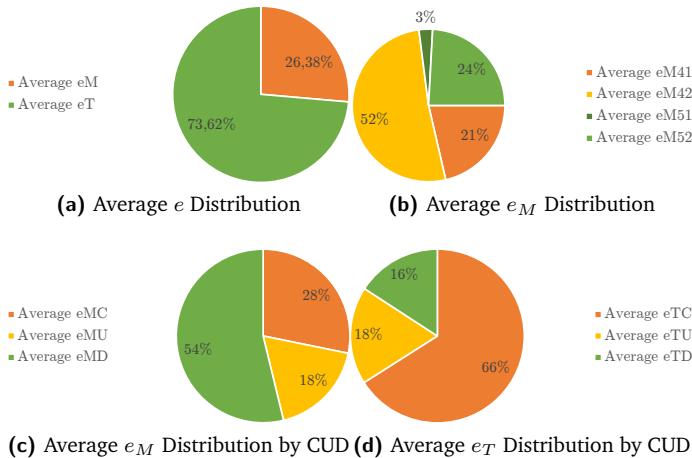


Figure 6.18: Effort Distributions

manual effort e_M (fig. 6.18c) distribution shows that the Prototyping Engineer mainly performs delete operations, whereas 2/3 of the automatic changes e_T from WASM-T (fig. 6.18d) add code.

Table 6.4: Effort Measurement Statistics, in SLOC

$\overline{e_M}$	$\overline{e_T}$	$\sigma(e_M)$	$\sigma(e_T)$	$\overline{e_M^C}$	$\overline{e_M^U}$	$\overline{e_M^D}$	$\overline{e_T^C}$	$\overline{e_T^U}$	$\overline{e_T^D}$
121	338	4.5	4.5	34	22	65	224	62	54

Figure 6.19 shows the questionnaire results. As can be seen, understanding of the ReWaMP process was rated very differently across the test subjects, but overall not easy (median 2). The manual work re-

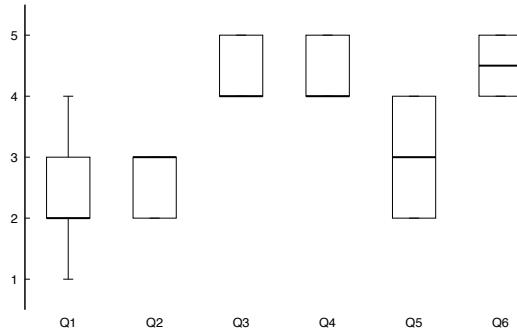


Figure 6.19: ReWaMP Questionnaire Results, 5-level Likert scale, Q1 ReWaMP easy to understand, Q2 manual work for ReWaMP, Q3 WASM-T did much work, Q4 without WASM-T more work, Q5 process was short, Q6 without WASM-T more time

quired is considerable, rated medium (median 3), but test subjects agreed that WASM-T took over a large share of work and that without WASM-T the work would have been significantly more extensive (both median 4). The process was considered medium short (median 3) and subjects agreed that without WASM-T it would have taken significantly longer (median 4.5).

A live example of a ReWaMP prototype created during evaluation and its WASM source are available online¹¹³, a screenshot of this prototype is shown in fig. 6.2.

Analysis. The measured times show a high variation in the time for manual tasks (σ about 10min, i.e. about 20% of t_M). These variations are highly dependent (Pearson's $\rho = -0.823$, $p = 0.0442$, highly significant at $\alpha = 0.05$) on the programming expertise of the Prototyping Engineer: the more experienced the test subject, the lower t_M . Con-

¹¹³<https://vsr.informatik.tu-chemnitz.de/demos/ReWaMP> Retrieved: 6.12.2019

sidering that the ReWaMP process was new to the test subjects, the measured times are rather short with the first view, u_{cal} finished in less than 45 minutes. As expected, the expert interventions took the largest share of time. The relatively long time for task 1 was not expected and may be the result of two factors: low decomposability/readability of the legacy code and unfamiliarity with code base.

While the latter problem would not be faced to the same extent by Prototyping Engineers from the ISV, the former represents a relevant characteristic in Web Migration. The apparent increase in time for task 5 (expert file) between u_{cal} and u_{new} represents a difference in the complexity of these views: while the expert file for u_{cal} must contain only two empty functions, the expert file for u_{new} is considerably more complex as it requires mock classes for room, doctor, patient, etc.

Effort measurements show little variation, neither for manual nor WASM-T tasks ($\sigma = 4.5$ for both), because for the same views, very similar code changes were required. This is also an indicator of a relatively low complexity of the changes since more complex code would show more variety, depending on the test subjects' expertise. The overall manual effort of 121 SLOC per view is relatively small, in particular, compared to the 338 SLOC of changes by WASM-T. Analysis of the effort distribution in the create/update/delete categories shows that the Prototyping Engineer performs mainly simple delete operations. In contrast, WASM-T mainly adds code. Similar to time analysis, the higher complexity of u_{new} shows in the difference of manual efforts for task 5: u_{cal} has $\overline{e_M^{5,1}} = 7$ SLOC, compared to u_{new} with $\overline{e_M^{5,2}} = 59$ SLOC, due to the more complex expert file required. There is no significant correlation between measured times and measured efforts. The subjective perceptions of test subjects ranked manual work effort

medium, but are in agreement with the objective effort distribution that WASM-T took over a large share of work, and without the tool support, their work would have been more extensive.

Threats to Validity. *Construct validity* of this experiment is threatened by possible variations in the execution of the test runs. This was addressed through the provision of the same, well-defined state for all test subjects. Identical execution of the test runs was achieved due to the underlying formalized ReWaMP process. The time and effort measurements of the experiment address the two main characteristics of Rapid Prototyping.

Internal validity of this experiment is threatened through potential subjective biases. This was addressed through the establishment of guidelines that strictly governed interactions between test subjects and the researcher. While some test subjects might have been biased towards providing slightly positive responses to the survey, the design of the evaluation experiment combining the subjective perceptions with objective time and effort measurements still provides a reasonable basis for an analysis of overall complexity, comparison of tasks and effort distribution between manual tasks and tasks automated in the toolchain.

External validity of the experiment is threatened through limitations in the generalizability of results. The main factor is the test subjects, that are from a similar age and experience range. However, the low Web Engineering expertise in comparison to general programming expertise represents an essential characteristic of the Migration Engineer stakeholder group. To increase generalizability, a large-scale study with test subjects from different ISVs would be required. While these studies are feasible in the context of funded research projects – a single test run requires between 1.25 to more than 2 hours – they are out of

scope for this thesis. The second factor is the legacy code base from which the two test objects were taken. This was indeed representative, as the two main views from the scenario application described in table 2.1 were selected as test objects.

Conclusion of ReWaMP experimentation. The experiments have shown that ReWaMP can successfully produce web migration prototypes, reusing legacy business logic through compilation to WebAssembly and the ReWaMP runtime environment. All test subjects were able to complete the test run successfully. The ReWaMP process and WASM-T toolchain support the Prototyping Engineers to successfully and rapidly create web migration prototypes. Observations show that the initial understanding of ReWaMP is not easy for test subjects. Thus additional guidance in the process is required. Automation and tool support has a significant impact on both objective measurements and subjective perceptions. Thus, adding more support for the most complex manual tasks, task 1 and 4 is desirable. Task 1 could also benefit from the knowledge extraction results of AWSM:RE. It can be argued that a comparable prototype can be created by an experienced Web Engineer in comparable time. The main contribution of ReWaMP, however, is to enable software engineers without significant Web Engineering expertise from the existing ISV staff to achieve similar results. The experiment showed that experience of the legacy programming language and legacy UI framework are essential. Fundamental Web Engineering expertise is helpful but not required.

6.6.3 Experimental Evaluation of RWMPA

The following experimental evaluation of RWMPA focuses on the aspect of improving support for manual interventions by the Prototyping Engineer.

Setup. Experimental evaluation of RWMPA was based on a similar setup like the ReWaMP evaluation with the test dataset $B_T \subset B$ consisting of the 2 views $B_T = \{u_{cal}, u_{new}\}$ from the scenario application also used in all tasks of the ReWaMP evaluation. The experiment was conducted through observed test runs of the guided ReWaMP process in RWMPA with test subjects. The *test subjects* impersonated the role of Prototyping Engineer and were observed during execution by a *researcher*. Table G.1 shows the guidelines for interaction of the researcher and the test subject. The questionnaire shown in appendix G.1 implemented as Google Form was set up to capture test subject data and subjective perceptions of the RWMPA workflow, RWMPA tool support, and task-related data.

Procedure. Each test subject attended to a brief introduction to the scenario and the RWMPA workflow. The researcher prepared the test run by resetting RWMPA to its start state: the scenario migration was in the RWMPA backlog, all migration services running, all help systems activated, the data from previous evaluation runs backed up and removed, and the measurement data reset. Interaction strictly followed the guidelines. The test subjects started the workflow and read the workflow explanation page, which visualizes and explains the entire RWMPA workflow. Then, the test subjects were asked to explain the workflow in their own words. If all user tasks have been described correctly, the researcher took a record of the workflow being *understood*. Then, the test subjects followed the sequence of steps of ReWaMP by performing the user tasks of the workflow. After reading the instructions provided by RWMPA for each user task, the researcher asked the test subjects for an explanation in their own words and took record if the specific user task was understood or not and intervened according to the guideline if required. All user tasks were performed within RWMPA; no external tools were used. After completion of the evaluation runs, test subjects responded to the questionnaire. Similar to the ReWaMP evaluation,

objective observations were made through measurements. The focus of the measurements was on the workflow and guidance, capturing time measurements $t = t_U + t_S$ for each user and service task, respectively. A stopwatch was used to measure t_U , t_S was measured in RWMPA via the JavaScript performance object and stored in `localStorage`.

Experimental results and descriptive statistics. The experiment was conducted with 7 test subjects, with an age range of 22-40 (mean 26), 7-15 years of programming experience (mean 10.29) and Web Engineering expertise (mean 2.71 on Likert scale 1-5). The test runs' overall time¹¹⁴ was between 00:59:12h and 01:30:54h. Table 6.5 shows aggregated times, for the full time evaluation data refer to table G.4.

Table 6.5: RWMPA Time Measurement Statistics

\bar{t}_U	\bar{t}_S	$\sigma(t_U)$	$\sigma(t_S)$	\bar{t}_6	\bar{t}_{10}	\bar{t}_{11}	\bar{t}_{16}
37:00	00.07	05:43	7s	01:11	01:58	16:59	08:30

On average per view, $\bar{t} = 00:37:52h$, of which $\bar{t}_S = 00:00:52h$ ($\sigma = 7s$) was required by the service tasks. The average time for user tasks was $\bar{t}_U = 00:37:00h$ ($\sigma = 00:05:43h$). Considering time distribution on the tasks for u_{cal} , 11 *replace code* took the major share of time (mean 00:23:58h), in contrast, task 6 *select files* was the shortest (mean 00:01:35h). For u_{new} , 16 *expert changes* was the longest (mean 00:14:11h), followed by 11 *replace code* (mean 00:10:01h). The number of correction cycles (complete sequences of tasks 14, 15, 16, 17) was higher for u_{cal} (mean 3.4) than for u_{new} (mean 7). The times of tasks early in the workflow are noticeably reduced.

¹¹⁴time measurements reported are based on five test subjects due to invalidity of measurements for u_{new} for two test subjects

Table G.2 shows the full empirical evaluation data. As shown in fig. 6.20, the subjective perceptions captured in the questionnaire show high agreement for the statements “The workflow is easy to understand” (median 4 on a 5-level Likert scale), “The workflow has reduced my work effort a lot” (median 4) and “The help page has strongly supported my understanding of the workflow” (median 4). The perceived manual interventions were considered medium (median 3). Test subject was aware of what is they are required to do (median 4), which was supported by the task guidance (median 4). Expert changes were considered the most difficult task by 4 test subjects, followed by replace code (3 test subjects). Manual configuration of migration tools was considered very low (median 1).

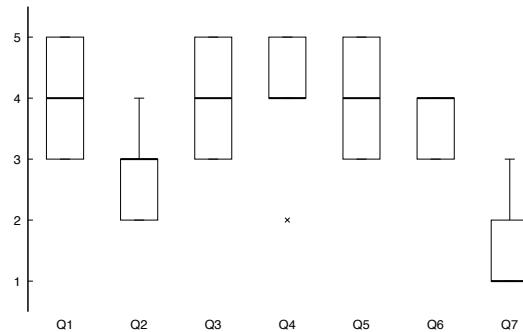


Figure 6.20: RWMPA Questionnaire Results, 5-level Likert scale, Q1 workflow easy to understand, Q2 much manual work required, Q3 workflow reduced work effort, Q4 workflow guidance supported workflow understanding, Q5 task guidance supported task understanding, Q6 aware what is to do, Q7 manual configuration of tools

Analysis. In comparison to ReWaMP, the time for manual interventions of the Prototyping Engineer was reduced from $\bar{t}_M = 00:49:36h$ ($\sigma = 00:10:06h \approx 20\%\bar{t}_M$) in ReWaMP to $\bar{t}_U = 00:37:00h$ ($\sigma = 00:05:43h \approx$

$15\% \bar{t}_U$). The time distribution across tasks in comparison to equivalent tasks of ReWaMP table 6.2 shows some differences. In particular, the long time for ReWaMP task 1 (BBL file extraction, mean 00:11:22h) was highly reduced in RWMPA task 6 (select files, mean 00:01:11h) due to automatic dependency analysis in service task 5, reducing the complexity of user task 6 to mainly accepting or rejecting the automatic results. Likewise, the time for the main activities of the Prototyping Engineer in ReWaMP, task 4 (expert changes, mean 00:30:49h), and 5 (expert file, mean 00:05:06h), requiring an average of 00:35:55h in total, is reduced in RWMPA workflow, in equivalent tasks 10 (delete code, mean 00:01:58h), 11 (replace code, mean 00:16:59) and 16 (expert changes, mean 00:08:30), with an average of 00:27:28h in total. This can be an indication that the automatic suggestions for deletion and replacement autocompletion based on WASM-T provided methods based on the extended preprocessing in RWMPA task 7 provided an acceleration for the expert interventions. The apparent increase of the time for task 16 (expert changes) and of the correction cycles between u_{cal} and u_{new} is a consequence of differences between the two views in terms of dependencies already pointed out in ReWaMP evaluation.

The results of the subjective evaluation based on the test subjects' feedback shows a rather high agreement to the usefulness of the RWMPA process guidance. In particular, compared to ReWaMP, the required sequence of tasks and the tasks themselves are better understood by the test subjects. The higher degree of automation and the consistent UI of RWMPA for the underlying WASM-T toolchain can be seen in the low perceived manual configuration of migration tools.

Threats to Validity. Due to the high similarity of this experiment with the ReWaMP experiment, the same threats to validity as in section 6.6.2 hold. This paragraph outlines the differences concerning *construct validity*. Identical execution of the test runs was not only guaranteed

due to the underlying formalized RWMPA workflow but also through the RWMPA process guidance. Due to an error in time measurements, the test runs on u_{new} of two test subjects were not measured correctly. To address this, measurements of these subjects on both test objects were excluded from the results. As both test subjects, however, completed both test runs, their feedback in the questionnaire has been considered. The comparisons between ReWaMP and RWMPA evaluation results should not be considered a comparative study since the test subjects were not the same. While this was not feasible and subsequent test runs of ReWaMP and RWMPA would have influenced the measurements due to experience transfer, the test objects, i.e. u_{cal} and u_{new} are identical.

Conclusion of RWMPA experimentation. The experiments have shown that RWMPA can improve the ReWaMP process of Rapid Web Migration Prototyping. In addition to the successful completion of all test runs as in the ReWaMP experiment, observations showed a better understanding of the workflow and its tasks by the test subjects. Due to the significantly extended guidance, they were aware of their context and what to do next and understood what was required in the tasks. The increased degree of automation and support for performing the tasks has successfully reduced the required time, in particular for task 1.

6.6.4 Experimental Evaluation of UI Transformer

The following experimental evaluation of UI Transformer focuses on two aspects: it explores the performance impact of the computational complexity of UI Transformation as well as perceived similarity and its influence factors in the transformed grid-based WUIs.

Setup. The *test dataset* L_T comprises 50 pixel-based legacy layouts $L_T = \{l_1, \dots, l_{50}\}$ with different levels of complexity (amount of controls $\#c$ between 2 and 130, $\overline{\#c} = 22.42$) and depth (levels of nesting d

between 0 and 2, $\bar{d} = 0.32$) represented as MFC resource files. To reach the high variety required for testing the automatic Transformation, these have been extracted from executables of various Desktop Applications using Resource Hacker. The test platform for the performance evaluation is an Intel Xeon E5-2670 CPU (2,6 GHz) with 16GB DDR3-1333 RAM running a minimal Arch Linux installation (Kernel version 4.15) without Desktop environment. Hyperthreading supports 16 threads on 8 CPU cores. For empirical evaluation, a subset $L'_T \subset L_T$ of 5 layouts was used with complexity from 7 to 130 controls ($\overline{\#c} = 8.24$, median $\#\tilde{c} = 26$). The full test dataset L_T is used for performance analysis. The questionnaire shown in appendix H.1 was set up to capture subjective perceptions of the similarity of legacy and transformed layouts in 7 criteria, each measured on a 5-level Likert scale of agreement.

Procedure. For performance analysis, the full test dataset L_T was transformed with UI Transformer, and time measurements were taken. Three times were measured corresponding to the three stages in fig. 6.14: time for layout analysis t_A , time for layout Transformation t_T and time for layout generation t_G . Time $t_T = t_0 + t_{opt}$ is further divided into the time for the initialization t_0 and the time for the optimization t_{opt} . These times were measured programmatically using pythons `time` object with a precision of one millisecond. Each test object (legacy layout) was transformed using 11 different combinations of objective functions: 1 $\{f_O, f_A\}$, 2 $\{f_W, f_A, f_L\}$, 3 $\{f_{D,H}, f_A, f_L\}$, 4 $\{f_{D,G}, f_A, f_L\}$, 5 $\{f_W, f_{D,H}, f_{D,G}\}$, 6 $\{f_O\}$, 7 $\{f_A, f_W, f_{D,G}\}$, 8 $\{f_{D,H}, f_O\}$, 9 $\{f_W, f_{D,G}, f_L\}$, 10 $\{f_W, f_{D,H}, f_{D,G}, f_O, f_A, f_L\}$, 11 $\{f_O\}$. Combination 10 comprises all objective functions. Combination 11 was measured on the C-based implementation to evaluate the potential for performance improvement. The following parameters were used for

the performance evaluation of each combination: 900 iterations of the evolutionary cycle, population size $|P| = 30$, mating pool size $|M| = 22$, recombination probability $P_r = 0.5$, mutation probability $P_m = 0.15$.

For empirical evaluation, L'_T was transformed to $L'_{T,grid}$ using combinations 1-5. Along with the initialisation approximation described in section 6.5, this created $|L'_{T,grid}| = 5 \cdot (5 + 1) = 30$ grid layouts for evaluation through test subjects. Screenshots were taken from these layouts to ensure that all test subjects would see the same visual layout, independent of browser, platform, etc. The study was conducted with 10 *test subjects* with an age range of 21 to 52. Each test subject was shown in random order pairwise combinations of a legacy layout from L'_T , and a corresponding transformed layout from $L'_{T,grid}$ and was asked to rate the pair with regard to the seven questionnaire criteria for each pair.

Table 6.6: UITransformer Performance Evaluation, all times but \sum in ms

Measure	Min	Max	Mean	σ	\sum (mm:ss)
t_1	13826	343834	60029	48040	14:31
t_2	11275	454761	75610	67207	19:39
t_3	19827	532608	107487	80564	26:12
t_4	26228	519752	111868	77746	26:17
t_5	32520	1040988	178444	152766	46:59
t_6	11046	312285	45344	42799	11:43
t_7	26766	745364	137568	111041	35:10
t_8	20692	635362	107390	91441	27:22
t_9	24792	663514	124481	98935	30:58
t_{10}	42196	1165719	206154	166035	50:47
t_{11}	62	15843	1501	2304	00:31

Experimental results and descriptive statistics. The entire benchmark took about 2.5h to complete and produced 50 grid layouts as outputs, based on $50 \cdot 30 \cdot 900 = 1.35$ million intermediate layouts in 900 generations. Table 6.6 shows the results of the performance evaluation per combination, aggregated over the 50 test objects; the detailed performance evaluation data is shown in table H.1.

Each test subject contributed $30 \cdot 7 = 210$ ratings, resulting in 2100 ratings overall, 300 per question in the questionnaire. Figure 6.21 shows the results of the questionnaire, the descriptive statistics on the averages aggregating the test subjects, layouts and combinations grouped by the 7 criteria are shown in table 6.7; the detailed data is available in table H.3.

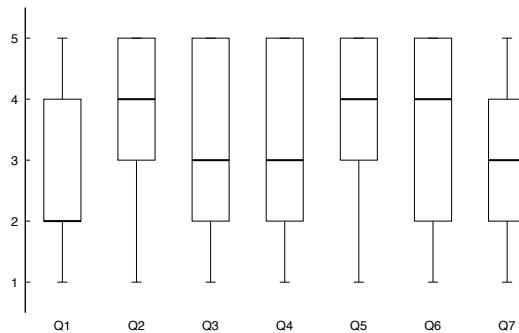


Figure 6.21: UI Transformer Questionnaire Results, 5-level Likert scale, Q1 same size, Q2 elements at least as large, Q3 left alignment consistent, Q4 right alignment consistent, Q5 same labels, Q6 same order, Q7 layouts identical

Analysis. The results show that layout Transformation through evolutionary optimization is a computationally complex task due to the high number of iterations (900 generations) and intermediate results

(1.35 million for 50 layouts) required. The time is highly dependent on the combination of objective functions, varying from about 45s for the single-function combination 6 to 206s for combination 10 of all 6 objective functions, with some functions like f_W (compare t_2 and t_3) having significantly less performance impact than others. The measured times vary significantly not only between combinations but also within the measurements of the same combination. For instance, t_6 ranges from 11046ms to 312285ms, with a standard deviation of more than 0.94 of the mean. These fluctuations are due to the different complexities of the user interfaces, which is the main influence factor on the Transformation times. The time measurements t_i and the number of controls $\#c$ all show highly significant ($\alpha = 0.001$), very high correlations (Pearson's $\rho \geq 0.945$, $p < 3.28 \cdot 10^{-5}$ for all t_i). The overall Transformation time of about 2.5h for 50 user interfaces seems high. However, this layout Transformation is a one-time-only process that is not required to be run repeatedly. The comparison between t_6 and t_{11} shows that the potential performance increase through implementation in a low-level hardware-oriented platform like C can significantly accelerate the Transformation.

Table 6.7: UITransformer Empirical Evaluation

Criterion	Min	Max	Mean	σ
Q1	1.5	3.9	2.59	0.65
Q2	2.8	4.6	3.92	0.42
Q3	1.2	5	3.11	1.14
Q4	1.2	5	3.09	1.15
Q5	2.1	5	3.77	0.79
Q6	1.9	5	3.37	1.07
Q7	1.5	4.9	2.85	0.97

The empirical results show that the transformed grid layouts are rated only medium with regard to the 7 criteria. In particular, criterion Q7 (agreement to “the layouts are identical”) was rated with an average of 2.85 by the test subjects. The perceived similarity was dependent on the layout complexity. While for l_1 ($\#c = 7$) similarity was rated at 3.53, for l_5 ($\#c = 130$) it drops to 2.15. There is a significant (at $\alpha = 0.05$) strong negative correlation between Q7 and $\#c$ ($\rho = -0.915, p = 0.0294$). Combination 1 ($\{f_O, f_A\}$) was rated best by the test subjects ($Q7_1 = 3.08$), combination 5 $\{f_W, f_{D,H}, f_{D,G}\}$ received the lowest rating ($Q7_5 = 1.94$). The highest similarity rating overall was achieved by the initial approximation ($Q7_6 = 4.1$). A comparison of Q7 with the other criteria allows identifying influences on the similarity perception of the test subjects. Kendall tau ranking correlation analysis was used for this aspect. The strongest correlations exist with Q6 (“elements have the same order”, Kendall’s $\tau = 0.605, p < 10^{-3}$) and Q5 (“elements are assigned with the same labels”, $\tau = 0.522, p < 10^{-3}$). However, these correlations differ widely across test subjects: for test subject 1, the strongest correlation with Q7 is Q1 ($\tau = 0.592, p < 10^{-3}$), for test subject 3 it is Q3 ($\tau = 0.657, p < 10^{-3}$). All correlations are highly significant at $\alpha = 0.001$.

Threats to Validity. *Construct validity* of this experiment is mainly influenced by the difficulty to measure subjective similarity perceptions. The seven criteria of the questionnaire attempt to map similarity onto concrete sub-aspects. Interpretation of these aspects, however, is subjective. This situation, however, is characteristic of empirical surveys on subjective perceptions. Formulation of Q7 (“identical”) might have been too strong, leading to lower overall ratings compared to “very similar”. However, the main focus of the experiment was on comparative statements, as in the analysis above. Rankings do not represent absolute values well.

Internal validity of this experiment refers to potential biases. While subjective biases from the researcher to the test subject were not a threat as there was no interaction between them during the experiment, some other factors are relevant. Test subjects' understanding of criteria might have changed over the time of the experiment. This was addressed by the relatively high number of overall rankings and the order randomization so that these effects are reduced by averaging. Identical viewing conditions were ensured by providing screenshots of the layouts to avoid bias by rendering differences in different platforms, screen sizes, etc.

External validity of the experiment is threatened through limitations in the generalizability of results. The main factor is the test subjects, that are from a similar age range. However, since subjective perceptions of similarity do not require specific knowledge or experience, evaluating with more test subjects might not significantly improve generalisability, similar to the Poisson relationship between test subjects and usability problems identified (Nielsen and Landauer, 1993), which also only requires relatively low numbers of test subjects without specific qualifications. The second factor is the legacy layout dataset L_T , which provided the test objects for both performance and empirical evaluation. To improve generalisability, these were taken from 6 different legacy MFC Desktop Applications and representing a wide range of different layout complexities.

Conclusion of UI Transformer Experimentation. The experimentation with UI Transformer has shown that the approach can be used to rapidly create grid-layout-based Web versions of pixel-based legacy user interfaces. The ordinal ranking scales from the empirical evaluation facilitated comparisons between different optimization variants. The index approximations of the initial population (cf. eq. (6.2)) were rated the highest with regard to similarity ($Q7_6 = 4.1$ of 5), which means

that it provides a reasonable quality that can be achieved with high performance. Optimization comes at the cost of lower performance, but is still acceptable for a process run once and the acceleration potential of low-level programming languages has been demonstrated. The experiment has shown that user interface similarity is not yet well understood and requires more research to map subjective similarity perceptions onto objectively measurable functions. This is addressed in more detail in chapter 7. It was observed that the complexity of user interfaces plays a vital role in similarity perception and computational complexity.

Improved objective functions with regard to higher similarity and better performance can be easily integrated into the model-driven UI Transformation framework of UI Transformer to enhance the web migration prototypes created by AWSM:RM. An alternative would be an investigation of crowdsourcing layouts for Web migration prototyping as explored by Nebeling (Nebeling, Leone, et al., 2012), in particular as a tool-supported adaption process similar to CrowdAdapt (Nebeling, Speicher, et al., 2013) based on the initial index approximations.

6.6.5 Research Results

The research objective addressed by this chapter, RO2, to enable demonstration of desirability and feasibility of a potential Web-based version of the Legacy System with limited resources and lack of Web Engineering expertise, is achieved. The effectiveness, efficiency, and expertise requirements of the AWSM:RM Method for creating a demonstrative web migration prototype have been demonstrated and detailed in three different experiments. The created prototypes are representing the main characteristics of Web Applications and can be created with limited effort and expertise through reuse-focused techniques, automation, and guidance. AWSM:RM RQ1 was answered through the prototyping approach presented in section 6.3. AWSM:RM RQ2 was answered

through the introduction of the novel Rapid Web Migration Prototyping strategy presented in section 6.3.1 and the business logic reuse technique presented in section 6.4. AWSM:RM RQ3 was answered through the guidance system presented in section 6.4.4 and the automated UI transformation specified in section 6.5.

6.7 Summary

This chapter presented the AWSM Risk Management Method, which facilitates the demonstration of feasibility and desirability of Web Migration and the plausibility of a Web-based version of the Legacy System, specifying a novel Rapid Web Migration Prototyping strategy. This strategy is supported through a WebAssembly-based toolchain and has been enhanced with a guidance and automation system. The conceptual model of the AWSM:RM techniques and their implementation in the Toolsuite have been described. Evaluation comprising three separate experiments has shown the feasibility of the approach, quality of its results and low effort. It has furthermore provided insights on activity complexity differences and suitable support mechanisms as well as on the need for research on the relation between subjective similarity perceptions and computable features, which is addressed in the following chapter.

AWSM Customer Impact Control Method

7

This chapter addresses research objective RO3:

To enable control of the impact of user interface changes through Web Migration on customers with limited resources and lack of Web Engineering expertise.

First, three research questions are derived from RO3, and the current situation is briefly analyzed to identify requirements and review related work. To address the research questions, two sections outline the two main contributions towards RO3 in the context of the AWSM:CI method: the concept of Visual Analysis of UI Similarity for customer impact control is introduced including a model for computation of similarity and a process for calibrating the similarity computation, and a technique for automatically detecting user interface elements is presented. Finally, the method is evaluated against the requirements, and the evaluation is supported by empirical experimentation.

7.1 Analysis

The following analysis briefly summarizes the situation of customer impact through visible changes in the user interface caused by Web Migration and the challenges of measuring the similarity of user interfaces for controlling the customer impact. Requirements are derived from this analysis and an overview of related work on the two enabling fields of research used for AWSM:CI is given.

7.1.1 Situation and Challenges

ISVs, as described in section 2.1, have accumulated large userbases, with long-running customer relationships and contracts for updates and maintenance, that are familiar with the Legacy System and well-adapted to its legacy user interface. Adaption can extend up to the business process level: we observed that the business processes in many doctors' offices are governed by the processes represented in the PMS (cf. section 2.2.1). This makes it hard to introduce changes in user interaction or even replace the PMS because of the potential *customer impact* (cf. fig. B.1). In industry contexts, it is therefore often a requirement to limit differences and maintain the *look and feel* of the legacy user interface to avoid forcing end users to change their working habits (Lucia, Francese, Scanniello, and Tortora, 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006; Distante, Perrone, et al., 2002). Customers often “require new applications to look like the old ones through similar screens and interaction ways even if new applications are Web-based” (Barbier, Deltombe, et al., 2013). Learning how to use a new Web user interface requires an effort for customers. Thus changing the UI poses a risk for the ISV.

Web Migration visibly impacts the user interface and, therefore, its users. The legacy UI consists of windows, rendering controls according to the desktop GUI framework and platform. A Web UI, in contrast, is rendered in a Web browser in browser tabs, rendering controls according to the Web UI framework, platform, and specific CSS styles. The industry requirement of maintaining a similar look and feel causes the challenge of *UI Similarity*. Analysis of UI Similarity is a research area of Human-Computer interaction (HCI), application of which exceeds Web Migration, supporting testing, cross-platform GUI matching, GUI modernization, and project effort estimation (Grechanik, C. W. Mao, et al., 2018).

To “avoid presenting the user with a drastically different application” (Distante, Tilley, and Canfora, 2006), changes need to be controlled, favoring an incremental transition evolving the software over abrupt changes (cf. also fig. B.1). Design decisions concerning the user interface need to be informed through UI Similarity Analysis. Changes that affect UI Similarity can occur in several dimensions: *Task* (Bakaev, Khvorostov, et al., 2017b; Stroulia, El-Ramly, and Sorenson, 2002), i.e. the workflow required to achieve a specific user goal on the UI, *Behavior* (Stroulia, El-Ramly, and Sorenson, 2002), i.e. the way the user interacts with the user interface, *Thesaurus*, i.e. the textual vocabulary used in the user interface for labels etc. and the visual vocabulary such as icons, images etc., *Layout*, i.e. how UI controls are placed and sized, and *Material*, i.e. how UI Elements look like (Bakaev, Khvorostov, et al., 2017a). UI Similarity in the context of Web Migration focuses on dimensions in which changes cannot be avoided, to allow for an informed choice within the degrees of freedom of the UI design space.

Changes in Layout and to some degree in Behavior and Material follow conclusively from the *change of environment* through Web Migration. Layout changes result from the different layout paradigms (pixel-based/grid-based, cf. section 6.5), material changes (GUI framework controls/CSS-styled controls), and viewing conditions (fixed-size desktop window/variable-sized browser window), which cause changes in the positioning and size of UI controls. Positions and sizes of controls define the degrees of freedom of the design space for creating a Web-based version of a legacy UI. As Layout describes the visual organization of content, it provides orientation for the user and has a considerable impact on the Behavior dimension. Thus, similarity of Layout is particularly relevant for UI Similarity in the context of Web Migration.

Measurability is necessary to quantify the state of Legacy Systems (Masak, 2006). However, existing legacy metrics are focused on the Legacy System only and do not consider target and source system together. To control the customer impact of user interface UI changes through Web Migration, changes affecting UI Similarity must be made measurable. Existing Web Migration approaches as in section 3.2 barely consider UI Similarity beyond an abstract level, without providing a concrete way of measuring it. Existing UI Similarity Analysis approaches cannot be applied to legacy and Web UIs because they are based on code analysis or target a different degree or dimension of similarity. Measurement of UI Similarity is therefore not directly applicable to Web Migration and needs to be adapted to the characteristics of migration and the situation of ISVs with non-Web legacy desktop software.

Thus, the challenge is to specify a suitable customer impact method for measuring the UI Similarity between the legacy UI and Web-based versions of the UI that can be used to inform design decisions, control the amount of changes introduced and incrementally evolve the user interface towards a modern Web user interface with limited resources and Web Engineering expertise.

7.1.2 Requirements

The following requirements specify RO3, based on the analysis presented above and the AWSM principles.

Effectiveness. Customer impact control of Web Migration should be enabled through measurement of UI Similarity between legacy and Web user interfaces.

Efficiency. Measurement of UI Similarity should be supported by analysis tools to automate the measurement process.

Expertise. Measurement of UI Similarity should be feasible with available expertise of the ISV's staff.

Applicability. Measurement of UI Similarity should be applicable to a wide range of legacy and Web user interfaces, including user interface design prototypes (UI mockups), independent of the specific UI technology.

Calibratability. UI Similarity measurement results should be calibratable to the perceptual characteristics of specific customer target groups.

7.1.3 Related Work

This section introduces UI Layout Similarity Analysis and Analysis of UI Structure as key research areas enabling the AWSM:CI method. It establishes the semantics of the terminology and the blueprint strategies used in the following description of AWSM:CI and provides a minimal overview of related work in both fields.

UI Layout Similarity Analysis is an active field of research, comprising GUI Differencing and Cross-Browser Inconsistencies detection. *GUI Differencing* (Grechanik, C. W. Mao, et al., 2018; Grechanik, Xie, et al., 2009b; Grechanik, Xie, et al., 2009a) focuses on automatic GUI test script adaption after maintenance activities changing the GUI. Automated *sound* ($\text{precision}=1$) and *complete* ($\text{recall}=1$) GUI differencing is *undecidable* (Grechanik, C. W. Mao, et al., 2018). *Cross-Browser Inconsistencies (XBI) detection* is the automatic detection of layout variations of Web UIs rendered in different browsers/mobile platforms/viewports. Most approaches follow a *DOM-based strategy* (W. M. Watanabe et al., 2019; Roy Choudhary, Prasad, et al., 2014; Roy Choudhary, Versee, et al., 2010), requiring the DOM for segmentation of the WUI or identification of XBIs. *Computer-vision strategies* (Saar et al., 2016) work based on analysis of WUI screenshots. *WebDiff* (Roy Choudhary, Versee, et al., 2010) introduced the blueprint two-step *differential testing* (McKeeman, 1998) strategy followed by many XBI approaches: 1) segmentation of the WUI according to its DOM structure and 2) comparisons of corresponding UI controls across the different versions of the user interface.

Both GUI Differencing and XBI detection analyze UI Layout Similarity, GUI Differencing within a fixed environment, i.e. on the same platform and framework, XBI detection within different HTML rendering engines. However, the UI Layout variations are more subtle compared to the consequences of migration from a desktop GUI to the Web: GUI Differencing addresses intended UI changes through perfective maintenance activities, XBI detection addresses even more subtle unintended changes through rendering differences of a WUI with identical description. While UI Similarity Analysis for Web Migration requires a similar matching of UI controls across different UI versions, these changes are more coarse-grain and prominent. Thus, the aspect of subjective perception of similarity is more important in contrast to the detection of fine-grain UI changes for GUI Differencing or XBI detection.

Analysis of UI Structure is an important pre-requisite for UI Analysis to identify the visually and semantically coherent two-dimensional segments, that constitute the UI. UI controls form atomic segments of a UI and create a hierarchy through containment and alignment relationships (Choudhary et al., 2013). *Page Segmentation* approaches address the problem of identification of segments in WUIs and are employed in Web data extraction, crawling, archiving, accessibility, visual quality evaluation, and automatic page adaption or retargeting (Sanoja and Gancarski, 2014; Kumar, Talton, et al., 2011; Wei Liu et al., 2010; Cai et al., 2003). The majority of approaches require the DOM as input for segmentation, which is not available for the source systems of Web Migration. *Vision-based Segmentation* approaches like (J. Kong et al., 2012) instead apply image processing for recognition of atomic interface objects. This requires two activities: segmentation into candidates, called Regions of Interest (ROIs), using edge detection and analysis of geometric shapes, and classification of GUI Element type or rejection of the candidate. Other vision-based strategies like

Sketch2Code¹¹⁵, AirBnB’s Sketching tool¹¹⁶, and pix2code¹¹⁷ focus on the generation of UIs from hand-drawn UI mockups through analysis of UI objects. While they provide strong examples of the power of computer-vision based approaches and the use of Artificial Intelligence (AI) in UI Similarity, our experiments could not produce any useable results when using legacy GUI screenshots¹¹⁸.

Consideration of *UI Similarity in Web Migration* is sparse. The few approaches which do, fail to provide a concrete way of measuring it (Barbier, Deltombe, et al., 2013; Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, and Tortora, 2008; Distante, Tilley, and Canfora, 2006), focus on interactions, i.e. the Behavior and Task dimension of UI Similarity (Cajas et al., 2019), disregarding layout and visual aspects, are targeting legacy systems with TUIs (Stroulia, El-Ramly, Iglinski, et al., 2003; Bodhuin, Guardabascio, et al., 2002) or which have already WUIs (Lucia, Scanniello, et al., 2007) and do only use static page structure information, falling significantly behind the state of the art in other fields like XBI detection or page segmentation.

Due to the independence of computer-vision-based strategies for UI Similarity Analysis from heterogeneous textual representations of UIs, section 7.3 describes the application of computer-vision to UI Similarity Analysis in Web Migration by specification of a technique for measuring the similarity between the similarity of layouts between legacy non-Web and Web versions of graphical user interfaces throughout Web Migration with limited resources and lack of Web Engineering expertise.

¹¹⁵<https://sketch2code.azurewebsites.net/> Retrieved: 6.12.2019

¹¹⁶<https://airbnb.design/sketching-interfaces> Retrieved: 6.12.2019

¹¹⁷<https://github.com/tonybeltramelli/pix2code> Retrieved: 6.12.2019

¹¹⁸a sample result for a legacy interface used in our own evaluation experiments (Heil, Bakaev, et al., 2016) is available here: <https://sketch2code.azurewebsites.net/generated-html/350d13fa-6bf3-43e4-9e6d-04a58d77bb0c> Retrieved: 6.12.2019

7.2 Research Questions

The analysis presented above raises three research questions:

AWSM:CI Research Question RQ1: How to control the impact of Web Migration on customers with limited resources and lack of Web Engineering expertise?

AWSM:CI Research Question RQ2: How to measure the similarity of non-Web and Web versions of a user interface?

AWSM:CI Research Question RQ3: How to analyze visual UI Similarity through computation of objective measurements?

To design and evolve a solution for research objective RO3, this chapter addresses these three research questions in the following sections.

7.3 UI Similarity Analysis

To enable the control of the impact of user interface changes caused by Web Migration on existing customers set as research objective RO3, AWSM:CI allows to measure the visual similarity between legacy and Web user interfaces. Addressing the resources and expertise constraint of RO3, AWSM:CI defines a technique for computation of similarity based on objective measures and a tool for automatic detection of user interface elements. It provides an answer to AWM:CI RQ1. Section 7.3.1 provides an overview of the method, section 7.3.2 specifies a technique for calibration of the method to target user groups, section 7.3.3 describes the computation of similarity and section 7.3.4 addresses integration of the method with ongoing development activities of ISVs.

7.3.1 Visual Analysis of UI Similarity

For analyzing visual similarity of user interfaces in Web Migration, a computer-vision based approach is required. This section addresses the challenge of specification of a computer-vision approach for User Interface Similarity Analysis, answering AWSM:CI RQ2.

Visual UI Similarity Analysis in the context of Web Migration aims at measuring the similarity between the non-Web legacy GUI u_{legacy} and the Web-based versions of the GUI $\{u_{\text{web},1}, \dots, u_{\text{web},n}\}$ created throughout Web Migration based on *visual aspects* (Heil, Bakaev, et al., 2016; Bakaev, Khvorostov, et al., 2017a). These visual aspects are *features* x_i defined through *feature engineering* for a *design mining* process (Bakaev, Khvorostov, et al., 2017b) and used as vector components for a *vector-space representation* $\vec{x}_u \in \mathbb{R}^d$ of UI u required for comparison and classification. Calculation of similarity measurements is based on these features, identifying *regions of interest* $R_u = \{r_1, \dots, r_l\}$, and extracting visual features for these regions. AWSM:CI defines two categories of features: *atomic features* (equivalent to *base measures* ISO/IEEE, 2017a) which can be assessed per UI independently such as density (Heil, Bakaev, et al., 2016) or complexity (Bakaev, Khvorostov, et al., 2017a) and *relative features* (equivalent to *derived measures* ISO/IEEE, 2017a) which are assessed on a pair of user interfaces and describe differences such as order or orientation (Heil, Bakaev, et al., 2016).

These two feature types closely relate to different levels of UI Similarity. Atomic features require a lower structural and content similarity than relative features. For instance, complexity can be assessed of two arbitrary UIs and compared, whereas changes in order can only be assessed on a pair of UIs sharing a common subset of UI controls. GUI Differencing and XBI detection focus on a very narrow notion of similarity

of UI versions with little to no changes based on relative features such as position and size changes. A design repository and search engine like Webzeitgeist (Kumar, Satyanarayan, et al., 2013), on the other hand, requires a wide notion of similarity to be applicable to arbitrary UIs mainly based on atomic features such as GIST features (Oliva and Torralba, 2001). Web Migration is located between these two extremes: the UI differences are higher than between versions of the same UI (GUI Differencing) or WUIs rendered in different browsers (XBI detection), but lower than between arbitrary UIs (Design Search Engines) and legacy and Web-based UI share a common subset of controls. Thus, UI Similarity in the context of Web Migration requires both atomic and relative features. Based on these features, a *similarity function*

$$sim : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R} \quad (7.1)$$

can be calculated. The similarity function sim needs to be an approximation of *perceived UI Similarity* \mathfrak{S} , which is influenced by subjective cognitive aspects (Bakaev, Khvorostov, et al., 2017a) and semantic structure of the GUI:

$$\mathfrak{S} : U \times U \mapsto \mathbb{R} \quad (7.2)$$

UI Similarity is still an active field of research and perceived UI Similarity \mathfrak{S} is not yet well understood. \mathfrak{S} can be empirically measured, e.g. as we demonstrated using a set of *Kansai scales* (Bakaev, Khvorostov, et al., 2017a) and pair-wise comparisons. Yet, empirical measurement requires test subjects from the target user group, i.e. the ISV's customers in section 2.1. If a similarity function sim calculated from feature

vectors is indicative of \mathfrak{S} , then it can be used to replace tedious empirical comparisons of UI pairs in Web Migration and thus facilitate predicting perceived similarity compared to manual assessment. For sim to perform as approximation of \mathfrak{S} , at least a *concordant* (positive monotonic) relationship is required, i.e. an arbitrary (but unknown) monotonic function can describe the relationship between \mathfrak{S} and sim for larger sample sets (*generalization assumption*):

$$sim(\vec{x}_u, \vec{x}_{u+}) > sim(x_u, \vec{x}_{u-}) \implies \mathfrak{S}(u, u^+) > \mathfrak{S}(u, u^-) \quad (7.3)$$

That means that *Visual UI Similarity Analysis for Web Migration* is a process of constructing a similarity function sim based on vector-space representations of two user interfaces derived from visual features, that is indicative of perceived UI Similarity \mathfrak{S} . The conceptual model of this process is shown in fig. 7.1.

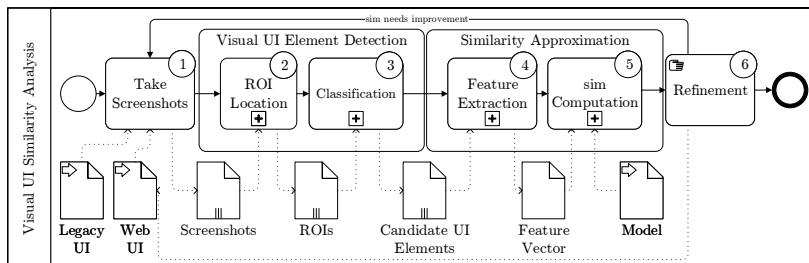


Figure 7.1: Visual UI Similarity Analysis Process

The process follows a visual bottom-up version of the WebDiff strategy, starting with screenshots of both UIs in (1), the Visual UI Element Detection phase, which consists of locating ROIs (2) and classification

(3), followed by the similarity approximation phase based on feature extraction (4) and computation of *sim* (5) which can be then used to drive design decisions and refine (6) the WUI and repeat the cycle.

This process defines the AWSM:CI method. Section 7.3.2 describes a technique for creating a version of the Model input artifact for *sim* Computation calibrated to a specific target user group, section 7.3.3 defines the computation of the similarity function and section 7.4 presents a technique and tool for automatically detecting visual UI Elements.

7.3.2 Calibration

Visual perception is highly human-dependant and subjective by definition. Our experiments show different influence of visual features in perceived UI Similarity \mathbb{S} for different test subjects (Bakaev, Laricheva, et al., 2018; Bakaev, Khvorostov, et al., 2017a). Thus it is unlikely to construct a similarity function *sim* that will generalize well on larger populations. Instead, *sim* needs to be a *parametric function* allowing for *calibration* on a *limited target group*, i.e. ISV's customers, through adjustment of parameters with representatives of the target group. These parameters assign weights to the visual features in section 7.3. Adjusting weights based on a set of samples is equivalent to *training a model*, shown as input artifact in step (5) of fig. 7.1, in *supervised machine learning* contexts. As alternative solution design, complete supervised learning of *sim* can be considered, i.e. a supervised regression taking a pair of screenshots from two user interfaces as input and learning to predict the value of *sim* as output. This strategy requires a *deep neural network* (DNN) architecture. In particular convolutional neural networks (CNNs) have been shown to be effective for computer vision. However, DNNs require large sample sets for training (Weibo Liu et al., 2017) in order to model the relationship from the independent variables (the pixels of the screenshots) to the dependent variable (the value of

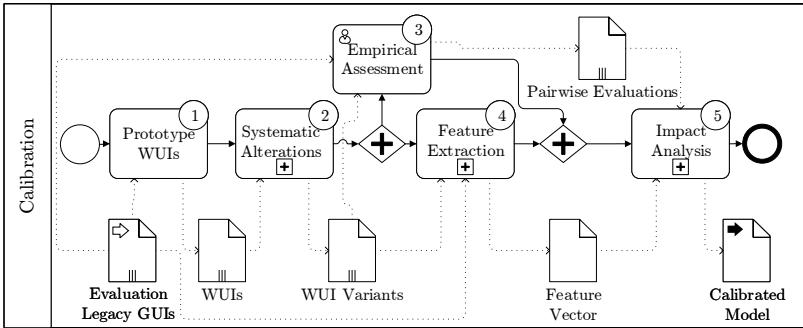


Figure 7.2: Calibration Process

sim). For instance, the popular *ImageNet* dataset (Deng et al., 2009) contains well over 14 million labeled samples. Creation of a sample set for DNN-based visual Similarity Analysis would require a large amount of pair-wise similarity evaluations by human test subjects and is therefore not feasible as it would violate the limited resources constraint of RO3.

Instead, AWSM:CI tries to find an adjustable mapping onto a limited number of influence factors (visual relative features) which can be computed algorithmically and calibrate the weights of these factors, constituting the model, with a limited number of empirical evaluations in the target group. This restricts the generalization assumption in eq. (7.3) to a much smaller scope and allows calibration of the model for the perceptual characteristics and viewing conditions, i.e. devices, aspect ratios, resolutions, etc., of the target group.

The calibration process is shown in fig. 7.2. It creates Web user interface prototypes (1) for a set of evaluation GUIs from the Legacy System, e.g. using the UI Transformation technique of AWSM:RM described in section 6.5. Then, these Web user interfaces are systematically altered

with regard to different visual features, similar to the genome mutation strategy in section 6.5. Representatives of the target user group provide pair-wise similarity evaluations for pairs of one legacy GUI and one Web user interface variant through empirical assessment (3). This step is equivalent to the evaluation of fitness in section 6.5. Figure 7.3 shows an example of a layout variation, altering the base layout of a patient data input form in fig. 7.3a in terms of Orientation, which results in fig. 7.3b. The two highlighted form groups A and B are re-arranged, creating a layout variation with a more pronounced wide-screen aspect ratio, and the aspect ratios of A and B changed in the opposite direction. Such pairs can be used for the empirical assessment to calibrate the similarity model. As becomes evident in the example, layout dimensions are not entirely independent: the prominent change in Orientation also influences Density and Order to some extent.

The calibration process uses the same feature extraction (4) like step (4) of the overall Visual UI Similarity Analysis process in fig. 7.1, producing a feature vector. Through combined analysis of the feature vector and the pair-wise evaluations, a *calibrated model* is derived, representing weights according to the impact of features. This calibrated model can then be used as input model for step (5) in fig. 7.1. The computation of Visual UI Similarity described in section 7.3.3 supports this calibration process through computation of similarity measures with adjustable weights for different perceptual features.

7.3.3 Computing Visual UI Similarity

This section describes the implementation of the AWSM:CI method by describing the realization of Visual UI Similarity Analysis and Calibration through computation of adjustable similarity measures. The section provides an answer to the AWSM:CI research question RQ3.

Similarity measures play a crucial role in Case-based reasoning (CBR) systems, also known as similarity searching systems. A similarity measure quantifies the “degree of resemblance between a pair of cases” (Liao et al., 1998). For AWSM:CI these cases are two versions of a user interface. To compute the AWSM:CI similarity measure, the *distance-based (computational) approach* is applied, which defines the similarity measure based on a *distance* calculated from objects (features) of the cases (UIs). We define three variations of the **AWSM:CI similarity measure**:

sim_E based on Euclidean distance:

$$sim_E(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - DIST_E(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - \sqrt{\sum_{i=1}^d w_i^2 dist^2(x_{u_1,i}, x_{u_2,i})} \quad (7.4)$$

sim_H based on Hamming distance:

$$sim_H(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - DIST_H(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - \sum_{i=1}^d w_i dist(x_{u_1,i}, x_{u_2,i}) \quad (7.5)$$

and sim_R based on Tversky ratio mode:

$$sim_R(\vec{x}_{u_1}, \vec{x}_{u_2}) = \frac{\alpha \times \text{common}}{\alpha \times \text{common} + \beta \times \text{different}} \quad (7.6)$$

for $\vec{x}_{u_1}, \vec{x}_{u_2} \in \mathbb{R}^d$; $w_i \in \mathbb{R}^+$ and common, different $\in \mathbb{N}_0$; $\alpha, \beta \in \mathbb{R}^+$.

The similarity functions sim_E , sim_H and sim_R are *similarity measures* (ISO/IEEE, 2017a), not *metrics*, because a metric requires a *metric space* fulfilling the three axioms of positive definiteness, symmetry, and triangle inequality. This is not the case for sim_E , sim_H and sim_R . However,

DIST_E and DIST_H can be shown to be *pseudo-semimetrics*, i.e. fulfilling positivity, symmetry, and the relaxed positive definiteness axiom as one-sided implication $\vec{x} = \vec{y} \implies \text{DIST}_E(\vec{x}, \vec{y}) = \text{DIST}_H(\vec{x}, \vec{y}) = 0$.

Form A (Top):

Land,PLZ,Ort	D	98052	Bamberg
Strasse, Nr.	Musterringstrasse 487		
Anschriftenzusatz	Anschriftenzusatz		
Postfach	Postfach		
PF-Land,PLZ,Ort	PFLand	PFPLZ	PFOrt
Vorname	Patient		
Nachname	Concept		
Akademischer Titel	Titel	Zusatz	Zusatz
Kontaktdaten	Pivot	654646466	
Doppelkilometer 0 ? Wegpauschale 0			
optionale Pat.-Nr. 2 Geburtsdatum 1978-04-05			
verstorben am verstorben Geschlecht Männlich			
Patient seit 2013-02-20			

Form B (Bottom):

Privatpatient		Einlesedatum	Einlesedatum
Kostenträgerkennung	100100008	Versicherten-Nr.	0545454545
Kostenträger	BARMER GEK	versichert als	Mitglied
DMP-Kennzeichen		WOP(Kennzeichen)	
KTAB	Primärabrechnung	versichert ab	VersichertAb
besondere Personengruppe	4 - BSHG Bundesosozialhilfegesetz	versichert bis	2016-12-31
Stammart	A-KZE Karl-Heinz Zeus (ZEUS1)		
Kostenträger	Aktuelle Daten	von	2013-01-01
		bis	bis
schnell neuer Termin Daten ändern Sichern Patient fertig der Nächste bitte			

(a) Layout

Form A (Left):

Nachname	Concept	Geburtsdatum	1978-04-05
Vorname	Patient	verstorben am	verstorben
Akademischer Titel	Titel	Geschlecht	Männlich
Strasse, Nr.	Musterringstrasse	Wegpauschale	0
Anschriftenzusatz	Anschriftenzusatz	Doppelkilometer	0 ?
Land,PLZ,Ort	D 98052 Bamberg	optionale Pat.-Nr.	2
Postfach	Postfach	Patient seit	2013-02-20
PF-Land,PLZ,Ort	PFLand PFPLZ PFOrt	schnell neuer Termin Daten ändern Sichern Patient fertig der Nächste bitte	
Kontaktdaten	Pivot	Kontakt-Daten	

Form B (Right):

Kontakt-Daten	Actualle Datei	von	2013-01-01	bis	bis
Privatpatient					
Kostenträger	BARMER GEK	Kostenträgerkennung	100100008		
KTAB	Primärabrechnung	WOP(Kennzeichen)			
versichert als	Mitglied	versichert ab	VersichertAb		
DMP-Kennzeichen		versichert bis	2016-12-31		
besondere Personengruppe	4 - BSHG Bundesosozialhilfegesetz	Daten ändern			
Stammart	A-KZE Karl-Heinz Zeus (ZEUS1)				

(b) Orientation Variation

Figure 7.3: Layout Variations

Feature-based normalized distance functions dist define the calculation of DIST_E and DIST_H , and the weight factors w_i allow for the calibration process in section 7.3.2 to adjust the impact of different features. AWSM:CI uses three distances in the Layout dimension of UI Similarity:

Orientation, Order, and Density (Heil, Bakaev, et al., 2016). Figure 7.4 illustrates these three aspects for distance calculation. When migrating a legacy layout like in the top left, orientation changes can result from different available screen sizes, as shown in the top right. Order of elements or container elements can change due to the distribution into rows and columns, causing some elements to be assigned to a new row, as shown in the bottom left. Density can change due to different element sizes caused by changes in the Material dimension, i.e. different visual appearance and resulting size of controls like buttons, as shown in the bottom right. As seen, changes in one aspect may also produce changes in other aspects.

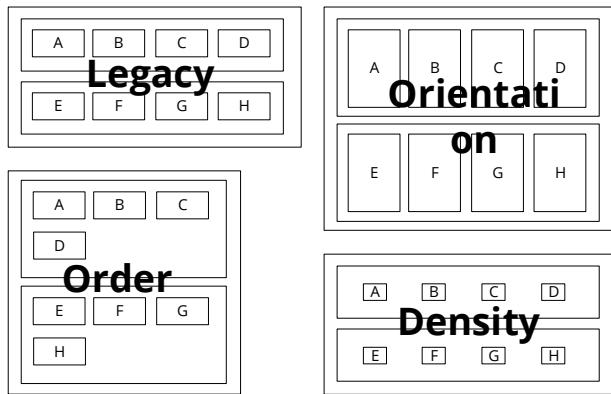


Figure 7.4: Legacy Layout and Changes in Orientation, Order and Density

All three distances are normalized to percentages using Tversky difference ratios with $\alpha = \beta = 1$:

$$\text{dist}(x_{u_1,i}, x_{u_2,i}) = \frac{\text{different}}{\text{different} + \text{common}} \quad (7.7)$$

These distances represent *base measures* (ISO/IEEE, 2017a) and are computed on the four different levels of the UI hierarchy introduced in section 7.4, region, block, group, and atomic elements. For *Orientation*, this results in the share of ROIs that have a different visual orientation. These orientations can be one of {vertical, horizontal, other}. Vertical orientation represents aspect ratios of bounding box b (cf. eq. (4.6)) with $w/h < \theta_O$ (i.e. portrait), horizontal orientation represents aspect ratios of $w/h > 1/\theta_O$ (i.e. landscape), other orientation implies $w/h \approx 1$ (i.e. approximate square). These can be calculated from a pair-wise comparison of the bounding boxes using threshold parameter $\theta_O \in \mathbb{R}, 0 < \theta_O < 1$. Assignment to class different or common is based on equality comparison of the resulting nominal orientation variable. Figure 7.3 shows an example of a WUI created for the scenario application described in section 2.1 and a layout variation with different orientation. *Order* differences can be computed using Levenshtein distances in 8 different orientations (left to right top to bottom, bottom to top right to left, etc.) on sequences, as described in section 6.5.3. Assignment to class different is equivalent to a Levenshtein distance of $\text{dist}_L > 0$, else common is assigned. *Density* represents the share of ROIs with different visual density. Computation of Density is based on bounding boxes, calculating the white space ratio W (Bakaev, Heil, Khvorostov, et al., 2019), i.e. the ratio of the area occupied by nested ROIs to the overall area. Assignment to class common is based on threshold parameter $\theta_D \in \mathbb{R}, 0 < \theta_D < 1$ if and only if $W_1/W_2 \in [\theta_D, 1/\theta_D]$. The distances are specific for Web Migration as they require the two compared user interfaces u_1 and u_2 to share a significant common subset of UI Elements: $\frac{|E_{u_1} \cap E_{u_2}|}{|E_{u_1} \cup E_{u_2}|} > 1 - \varepsilon$.

7.3.4 Integration with Ongoing Development

As described in G1, the neglect of integration with ongoing agile software development activities (C4 Agile) limits the applicability of many Web Migration approaches. Therefore, the AWSM:CI method needs to be integrated with ongoing daily development activities of the ISV. Figure 7.1 specifies a measurement-based process to be used during the Forward Engineering phase, i.e. the creation of the target system, of a Web Migration, corresponding to a *design measurement* of the implementation phase of software maintenance (IEEE Computer Society, 1998). Thus, its integration into ongoing forward development activities can be easily achieved. The measurement is based on automatic computer vision-based analysis and computation of similarity measures and does not require manual interaction apart from triggering the measurement. The measurement activity should be repeatedly triggered during the creation of Web versions of user interfaces to guide design decisions. The integration point can be parallel to other software quality measurements like cyclomatic complexity or function points (Kan, 1996) for both model-driven and non-model-driven development processes (cf. principle P3). If no dedicated measurement activities are specified in the ongoing development model, integration of AWM:CI has to be achieved in an ad-hoc manner into user interface design activities following the IEEE recommendations on performing measurement processes (IEEE Computer Society, 2014). While the measurement process itself can be started by any Migration Engineer, the calibration process should be executed by staff with expertise in empirical methods. For an ISV as in section 2.1.1, the usability experts and UIX designers, forming a dedicated team called Team U in medatixx, are well-suited due to their understanding of user interaction and qualification for empirical user analysis. Thus, when dedicated UIX experts are available in the ISV, integration of AWM:CI can be achieved in the context of the ongoing

Forward Engineering UIX analysis and approval activities. On artifacts level, the similarity measures represent another software quality measurement, if measurement activities are present, and are to be integrated into the corresponding set of measurement artifacts. If no measurement activities are present, the similarity measures represent a new type of software artifact which should be integrated into configuration management and expressed using an open standard (cf. principle P1) such as SMM (Object Management Group, 2012) or the IEEE measurement information model (ISO/IEEE, 2017a) to allow for interoperability.

7.4 Visual UI Element Detection

To conduct the Visual UI Similarity Analysis described in section 7.3 with limited resources according to RO3, this section describes a technique Visual UI Element Detection. Computation of the similarity function sim is based on distances in UI layouts. Most of these distances are depending on UI Elements: Distance in Orientation is calculated from aspect ratios of bounding boxes of high-level container UI Elements. Order distances compare positions of UI Elements on the viewport. Distance in Density uses Whitespace Ratio, which requires distinguishing areas occupied by UI Elements from areas that are empty. While UI Elements can be easily identified by human actors for whom graphical user interfaces are designed, this manual detection would be a very time-consuming activity that has to be repeated many times when using AWSM:CI, violating the limited resources constraint of RO3. Therefore, the technical challenge that needs to be addressed lies in the automation of Visual UI Element Detection as defined by steps (2) and (3) of fig. 7.1, providing the second part of the answer to AWSM:CI RQ3.

Section 7.4.1 outlines the conceptual model for Visual UI Element Detection, and section 7.4.2 describes the implementation in the Visual UI Element Detector Tool of the AWSM Toolsuite.

7.4.1 Conceptual Model

This section defines the conceptual model of Visual UI Element Detection in AWSM:CI by identifying the three involved perspectives on user interfaces and defining the related concepts and relationships, as presented in fig. 7.5.

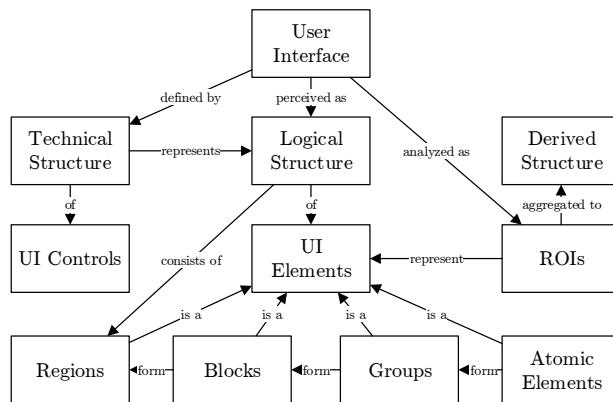


Figure 7.5: User Interface Concept Map

By extending the user interface formalism in section 4.6.3, this conceptual model allows for a visual bottom-up Reverse Engineering process of the user interface that works across legacy and Web platforms, addressing the applicability requirement in section 7.1.2. As it focuses on the identification of ROIs and deriving the structure through computer vision techniques instead of parsing the technical structure, the segmentation is independent of specific GUI technologies.

The user interface formalism in section 4.6.3 introduced the concept of user interfaces u being composed by a set of UI controls C_u , with controls $c \in C_u$ forming a hierarchy through parent-child relationships. These controls represent the *technical perspective* on a user interface u and are derived from their descriptions (KDM SourceFiles, ConfigFiles, Documents) in codebase B . To represent the *visual perception perspective*, *UI Elements* are introduced. UI Elements are the constituents of u visually perceived by human users forming its *logical/semantic structure*. There is no direct mapping between UI Controls and UI Elements: a UI Control may not be visually perceived as UI Element, and one UI Element may be composed of several UI Controls.

UI Elements form a similar hierarchical structure like UI Controls defined through spatial relationships like containment, alignment, and proximity. To represent aspects of *visual complexity* (Bakaev, Heil, Khvorostov, et al., 2019) in this hierarchy, which is known to have significant influence on perception (Tuch et al., 2009), we introduced four distinct levels of UI Elements: Regions, Blocks, Groups and atomic Elements (Heil, Bakaev, et al., 2016). *Atomic Elements* (Bakaev, Heil, Khvorostov, et al., 2019) are UI Elements without nested UI Sub-Elements; examples are buttons, inputs, or checkboxes. Through alignment and proximity, they form *Groups* like vertically aligned labels in a form. *Blocks* are formed from Groups and typically represent domain entities, e.g. entire forms. *Regions* are formed from Blocks and represent the top-level visual structure of a page such as navigation, header, content area.

To automate visual analysis, automatic recognition of UI Elements is required. This introduces the third perspective on a UI: the , which represents the perspective of a technical solution to approximate the visual perception perspective. Visual UI Element Detection tries to simulate human understanding of a user interface

through the identification of its constituents based on visual aspects. This consists of two tasks: identification of Regions of Interest and classification of the type of these ROIs. Thus, ROIs are the recognition perspective concept on a similar level, like UI Control and UI Element. However, the completeness and correctness of the mapping between UI Elements and ROIs depend on the quality of the GUI Segmentation. Also, ROIs can be smaller than Atomic Elements, e.g. the text label on a button is an ROI detectable through Optical Character Recognition (OCR).

7.4.2 Visual UI Element Detector

This section describes the implementation of the Visual UI Element Detection subprocess in fig. 7.1. Visual UI Element Detection consists of locating and classifying UI Elements in screenshots of legacy desktop or Web user interfaces. Thus, the process of Visual UI Element Detection shown in fig. 7.6 is split into two parts: location of ROIs and classification. Detection of UI Elements is object detection with specific features that distinguish it from other application domains: absence of image phenomena appearing in photos such as noise, glare, uneven lighting, perspective distortion, movement, incomplete or covered objects make analysis easier (Bakaev, Heil, Khvorostov, et al., 2019). On the other hand, UI Elements are relatively small (e.g. a checkbox) objects, have very similar, non-distinctive shapes (e.g. buttons and input fields), have different stateful visual appearances (e.g. checked and unchecked radio buttons), have different appearances in different platforms (e.g. buttons in MacOS vs. Windows), can occur on top of background images without differences in lighting or focus helping to distinguish layers, and have composite nature requiring context (e.g. checkbox and its label, a group of tabs), which makes UI Element Detection difficult. A *sound* and *complete* (i.e. high precision and maximum recall) mapping of UI Elements in two UIs based on computer vision is an *undecidable problem* (Grechanik, C. W. Mao, et al., 2018).

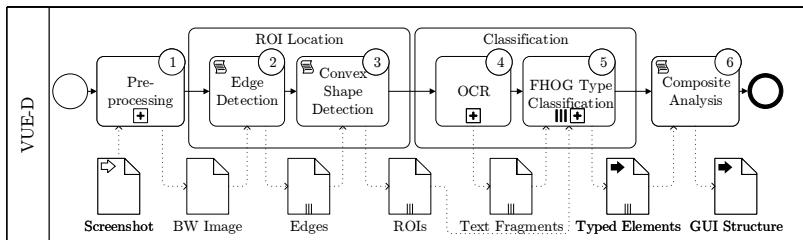


Figure 7.6: VUE-D Analysis Process

The *Visual UI Element Detector (VUE-D)* is implemented based on the computer vision strategy by J. Kong et al. (2012) as part of the HCI Vision approach (Bakaev, Heil, Khvorostov, et al., 2019; Bakaev, Heil, Khvorostov, et al., 2018) and is embedded in the UI Metrics integration platform¹¹⁹ (Bakaev, Heil, Perminov, et al., 2019). Figure 7.6 shows the VUE-D analysis process. The UI screenshot is subject to *preprocessing* (1) to improve detection results. It consists of grayscale conversion, upscaling, and black-and-white conversion using thresholding. OpenCV¹²⁰ is used for these tasks. *Detection of ROIs* is implemented using OpenCV's edge detection (2) for vertical and horizontal lines in the binary image. The rectangles are identified from edges as convex shapes (3) with 4 corners of a minimum area as boxes b , according to eq. (4.6). For *OCR* (4), VUE-D combines OpenCV close edge detection with *Tesseract*¹²¹. It identifies areas of high-frequency edges, does upscaling, and then performs OCR using Tesseract. For successful recognitions, the results are represented as bounding boxes b , and the recognized text is annotated as additional information. The *type classification* (5) uses specialized classifiers for different types of UI Elements. Each classifier is trained on one specific type using supervised learning on the Felzenszwalb histogram

¹¹⁹<http://va.wuikb.info/> Retrieved: 8.12.2019

¹²⁰<https://opencv.org/> Retrieved: 6.12.2019

¹²¹<https://github.com/tesseract-ocr/tesseract> Retrieved: 6.12.2019

of oriented gradients (FHOG) latent SVM feature (Felzenszwalb et al., 2010) extractor implemented in dlib¹²² which represents each input as 31-dimensional FHOG feature vector. Due to high visual differences, stateful elements like checkboxes, radio buttons are trained separately, as are platform variations. The classification result is then annotated as additional information to the ROI. *Composite Analysis* (6) identifies composite UI Elements using alignment, proximity, and containment, for instance combining recognized text labels with checkboxes, etc. by merging their bounding boxes. The architecture of VUE-D is shown in fig. 7.8. Figure 7.7 shows application of VUE-D on a screenshot of the GUI from fig. 7.3b with detection results highlighted.

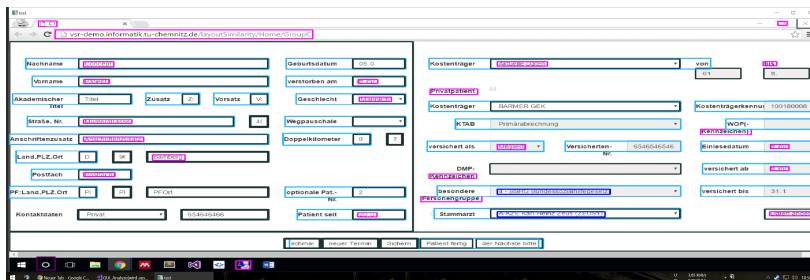


Figure 7.7: Visually Segmented Screenshot from VUE-D, pink highlights show OCR results, black hightlights indicate indicate convex shape detection results, blue highlights indicate composites

The UI Elements detected by VUE-D are represented using the XML metamodel employed in the Pascal VOC Challenge datasets¹²³, a de-facto standard for representation of Object Detection results, providing access to an existing environment of viewers, editors and libraries. Listing 7.1 shows a shortened example of VUE-D results serialized as Pascal

¹²²<http://dlib.net/> Retrieved: 6.12.2019

¹²³<http://host.robots.ox.ac.uk/pascal/VOC/index.html> Retrieved: 6.12.2019

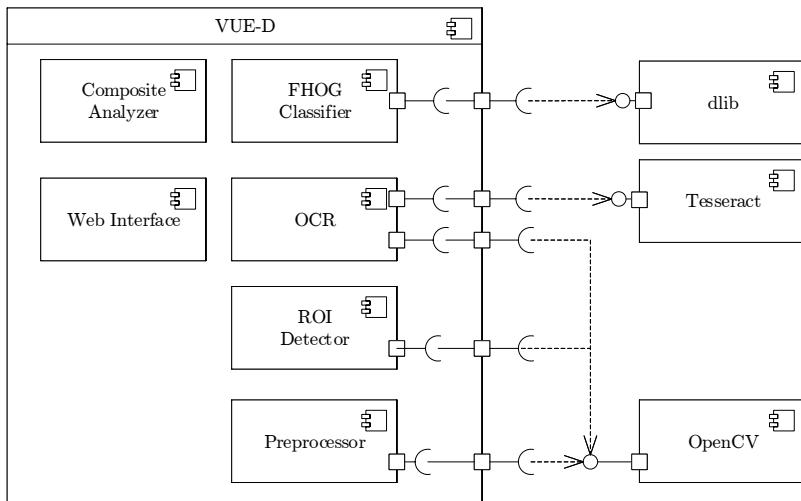


Figure 7.8: VUE-D Architecture

VOC XML. The sample indicates the screenshot file that was analyzed, the size of the viewport, and it contains three detected objects: a text field, a checkbox, and a button, together with their bounding boxes.

Listing 7.1: VUE-D Detection Results (shortened) represented as Pascal VOC model

```
<annotation>
<folder>test-set</folder>
<filename>1001.jpg</filename>
<path>/Users/sebastian/test-set/1001.jpg</path>
<size>
  <width>2880</width>
  <height>1628</height>
  <depth>3</depth>
</size>
```

```
<object>
    <name>input_text</name>
    <bndbox>
        <xmin>644</xmin>
        <ymin>65</ymin>
        <ymax>816</ymax>
        <ymax>127</ymax>
    </bndbox>
</object>
<object>
    <name>checkbox</name>
    <bndbox>
        <xmin>462</xmin>
        <ymin>1295</ymin>
        <ymax>489</ymax>
        <ymax>1328</ymax>
    </bndbox>
</object>
<object>
    <name>button</name>
    <bndbox>
        <xmin>237</xmin>
        <ymin>1469</ymin>
        <ymax>357</ymax>
        <ymax>1531</ymax>
    </bndbox>
</object>
</annotation>
```

7.5 Evaluation

This section evaluates AWSM:RM regarding three aspects. Section 7.5.1 evaluates AWSM:CI regarding the requirements in section 7.1.2. Additional experimental evaluation further address effectiveness, applicability, and calibratability in section 7.5.2. Section 7.5.3 revisits research objective RO3 in the light of the evaluation results.

7.5.1 Assessment of Requirements

This section reports on satisfaction of the requirements for AWSM:CI.

Effectiveness. The effectiveness of AWSM:CI is achieved through specification of a UI Similarity between legacy and Web user interfaces that produces concrete measurements to assess different versions of user interfaces, which can be used as a basis for decision making when comparing design alternatives. A detailed analysis of measurement results is presented in the following evaluation experiment.

Efficiency. Measurement of UI Similarity should be supported by analysis tools to automate the measurement process. The efficiency of AWSM:RM is supported through specification of similarity measures that can be computed based on automatically derived distances. The computer-vision-based approach for UI Element Detection of legacy and Web user interfaces is automated but does not yet produce results with production-grade quality. Thus, the efficiency requirement is considered only half-fulfilled.

Expertise. The computation of similarity measures and the calibration process of AWSM:CI do not require specific expertise and can be conducted using the artifacts (questionnaire) and analysis strategy (factor impact analysis and corresponding model calibration) pre-

sented in the following experimental evaluation as a blueprint by any ISV staff member with reasonable expertise in basic statistics. Likewise, the test subjects do not have any specific expertise requirements, as the similarity evaluation can be performed by any non-expert target group representative with normal vision. Thus, the Expertise requirement is considered as satisfied.

Applicability. The AWSM:CI method follows a pure computer vision strategy and does not rely on any specific technology-related analysis. Instead, the proposed distances and similarity measures can be computed on any image representation of a user interface, both conceptual and implemented, as they consider solely visual aspects of the user interfaces. Therefore, the Applicability requirement is satisfied.

Calibratability. The AWSIM:CI similarity measures can be calibrated to the perceptual characteristics of the target group following the calibration process described in section 7.3.2. The following experimental evaluation shows in more detail how the calibration process can be used to align the similarity measures with empirical evaluation results gathered from a small group of representatives. The Calibratability requirement is satisfied.

7.5.2 Experimental Evaluation of AWSM:CI

Experimental evaluation of AWSM:CI demonstrates the application of the similarity computation and calibration process based on a dataset of user interfaces from the real-world scenario application *x.concept* (cf. section 2.1) with empirical assessments of perceived similarity by test subjects (Heil, Bakaev, et al., 2016).

Setup. The *test dataset* U_T comprises 15 user interfaces that are based on 3 legacy user interfaces $U_L = \{u_1, u_2, u_3\}$ at low, medium

and high level of complexity respectively (amount of atomic elements $\#e_1 = 7, \#e_2 = 22, \#e_3 = 48$) and depth (i.e. levels of nesting, $d_1 = 0, d_2 = 1, d_3 = 3$). Interface u_1 is a simple graphical shift schedule, u_2 represents a calendar for appointment scheduling, and u_3 is an extensive patient data form. For each legacy user interface $\forall u \in U_L$, a Web-based version $u' = \text{web}(u)$ was manually created using HTML, JS, CSS, and bootstrap. The Web user interfaces $U_W = \{u'_1, u'_2, u'_3\}$ are copies of their original counterparts without any intentional layout changes introduced to them apart from changes due to the changed environment. Following the calibration process in fig. 7.2, a set of UI variants U_V was derived from U_W applying systematic alterations $V : U \mapsto U$ of Orientation, Order and Density $U_V = \{Vu' | \forall u' \in U_W, \forall V \in \{v_{\text{orientation}}, v_{\text{order}}, v_{\text{density}}\}\} = \{u''_{1,\text{ori}}, u''_{1,\text{ord}}, u''_{1,\text{den}}, u''_{2,\text{ori}}, \dots, u''_{3,\text{den}}\}$. Orientation alteration $v_{\text{orientation}}$ changed the layout from horizontal to vertical and vice versa by repositioning groups of UI Elements. Order alteration v_{order} changed positions of atomic elements within regions, maintaining proximity relationships between elements and their labels to avoid unintended Thesaurus changes. Elements were not mixed across regions (e.g. moving patient data inputs to the billing region) to avoid unrealistic changes unlikely to result from Web Migration. Density alteration v_{density} was achieved through changing the whitespace between elements for u_2 and u_3 and by replacing solid background fill colors with letters in u_1 . While each of these three intentional changes has its primary effect on its respective layout similarity dimension, the changes are not entirely orthogonal as they influence the other dimensions. The resulting test dataset $U_T = U_L \cup U_W \cup U_V$ of $3 + 3 + 9 = 15$ user interface is available online¹²⁴. All user interfaces from U_T were functionally identical. No changes were introduced in the Task dimension. For Behaviour,

¹²⁴<https://vsr.informatik.tu-chemnitz.de/demos/LayoutSimilarity> Retrieved: 6.12.2019

only minor changes through the Web environment exist. Task lists $T_i = \{t_1, \dots, t_n\}$ of different tasks for each original user interface u_i were specified. The questionnaire shown in appendix I.1 was designed to capture subjective perceptions of the similarity between legacy and Web user interfaces in 3 criteria, each measured on a 5-level Likert scale. The *testing environment* contained all Web user interfaces from $U_W \cup U_V$ and was set up to present all three Web versions per one legacy user interface in random order. In this way, the testing environment fulfills a similar function to the experimental platform of GUIDE, providing the user with random selected GUIs with controlled differentials among them (Grechanik, C. W. Mao, et al., 2018). Table 7.1 describes the detailed complexity measures of the three groups of WUIs.

Table 7.1: Amount of UI Elements in WUI per hierarchy levels

UI	Regions	Blocks	Groups	Elements
u'_1	1	2	4	7
u'_2	1	3	7	22
u'_3	1	3	7	48

Procedure. The evaluation experiment is designed as *within-subject* experiment with the distances and similarity measures as independent variables, and the test subjects' similarity perceptions as dependent variables. At the start of the experiment, the test subjects were shown how to achieve a group of tasks $t_j \in T_i$ on the corresponding legacy user interface u_i . Then, the testing environment randomly selected one of the four (one equivalent version u'_i and its three variations $u''_{i,V}$) WUIs and the test subjects were asked to complete the same tasks t_j in this user interface. When the task list was completed, test subjects were asked to respond to the questionnaire, providing their subjective

impressions. Then the testing environment randomly selected another UI from the remaining WUIs, and the process was repeated. The entire process was repeated for all three legacy user interfaces until all 12 WUIs have been assessed. The evaluation sessions were scheduled and conducted within one week in April 2016, with one session requiring about one hour of time. All sessions were performed on the same PC and screen in order to ensure consistent visual representation and interaction across all test subjects.

Experimental results and descriptive statistics. The experiment was conducted with 7 test subjects (5 male, 2 female), with an age range of 21 – 50 years (mean 28.4, $\sigma = 9.83$). All but one participant were German; all were proficient in German and English. The test subjects did not have prior experience with the test system, nor did they have HCI-related expertise. The subjective dataset captured from the test subjects using the questionnaire comprises 252 empirical evaluations, averages per UI are shown in section 7.3.3 and table I.1 shows the raw evaluation data. Table 7.2 shows the distances and table 7.3 the similarity measures computed according to section 7.3.3. The Orientation, Order, and Density Differences columns in table 7.2 indicate the differences of the UI in comparison to the corresponding legacy UI with regard to the hierarchy levels used for calculating the Tversky distances in eq. (7.7). The letters R, B, G, E preceded by a number are indicating the cardinality of the different-set per region, block, group, and atomic element, respectively.

The distances in table 7.2 have their minima for the primary Web versions u'_i of each UI. The mean distances are between $\overline{\text{dist}_{\text{ord}}} = 8.5\%$ and $\overline{\text{dist}_{\text{den}}} = 33.3\%$, standard deviations between $\sigma(\text{dist}_{\text{ord}}) = 13.7\%$ and $\sigma(\text{dist}_{\text{ori}}) = 20.7\%$. Density was observed to be the most affected

through Web Migration with even basic versions differing between 8.3% up to 33.3% from their legacy counterparts. Also, Density is the most affected by all other dimensions.

Table 7.2: UI Differences and distances

UI	Orientation		Order		Density	
	Diff.	dist _{ori}	Diff.	dist _{ord}	Diff.	dist _{den}
u'_1	1B	0.125	none	0	1G	0.083
$u''_{1,ori}$	1R, 2B, 2G	0.625	1B	0.167	1R, 1G	0.417
$u''_{1,ord}$	none	0	1B	0.167	1B, 1G	0.250
$u''_{1,den}$	1B	0.125	none	0	2G	0.167
u'_2	none	0	none	0	1B, 1G	0.089
$u''_{2,ori}$	1R, 1B	0.333	none	0	1R, 2B, 1G	0.603
$u''_{2,ord}$	none	0	3B	0.333	1B, 1G	0.089
$u''_{2,den}$	1B	0.083	none	0	1R, 2B, 1G	0.603
u'_3	none	0	none	0	1R	0.333
$u''_{3,ori}$	1R, 2B	0.417	none	0	1R	0.333
$u''_{3,ord}$	none	0	4G, 24E	0.357	1R	0.333
$u''_{3,den}$	none	0	none	0	1R, 2B	0.556

As all weights are $w_i = 1$ before calibration, Euclidean and Hamming similarity measures are identical. The means are $\overline{sim}_{E/H} = 0.57$ and $\overline{sim}_R = 0.807$ with standard deviations $\sigma(sim_{E/H}) = 0.208$ and $\sigma(sim_R) = 0.172$. While maxima co-occur for the basic Web version u'_i , $sim_{E/H}$ provides a more differentiating measurement. Regarding the empirical evaluations, difficulty was rated the lowest (mean

1.94, $\sigma(\text{difficult}) = 1.057$) and similarity the highest (mean 3.619, $\sigma(\mathfrak{S}) = 1.029$). Figure 7.9 shows a scatter plot of the similarity measures $sim_{E/H}$ and sim_R in relation to \mathfrak{S} .

Table 7.3: Computed sim measures and empirical Difficulty, Like and \mathfrak{S} evaluations, $w_i = 1$ for $sim_{E/H}$ and $\alpha = \beta = 0.5$ for sim_R , maxima highlighted in bold

UI	$sim_{E/H}$	sim_R	Difficult	Like	\mathfrak{S}
u'_1	0.8498	0.8571	1.7143	3.4286	4
$u''_{1,\text{ori}}$	0.2306	0.4286	2	2.5714	2.8571
$u''_{1,\text{ord}}$	0.6995	0.7857	2	3.5714	4.2857
$u''_{1,\text{den}}$	0.7917	0.7857	1.5714	2.5714	3.7143
u'_2	0.8413	0.9394	1.8571	3.2857	4.1429
$u''_{2,\text{ori}}$	0.3108	0.8182	2.1429	2	3
$u''_{2,\text{ord}}$	0.6308	0.8485	1.7143	3.7143	3.2857
$u''_{2,\text{den}}$	0.3911	0.8485	1.8571	3.1429	3.5714
u'_3	0.6667	0.9831	2.1429	3.1429	3.7143
$u''_{3,\text{ori}}$	0.4664	0.9322	2.8571	2	2.7143
$u''_{3,\text{ord}}$	0.5115	0.5085	2.1429	3	4
$u''_{3,\text{den}}$	0.4444	0.9492	1.2857	3.5714	4.1429

Analysis. To test the assumption in eq. (7.3), the relationship between the similarity measures and empirical similarity is analyzed. Spearman's r_s is used on the median values of \mathfrak{S} and the similarity measures to show the required concordant (i.e. positive monotonic) relationship. There is a strong positive correlation between $sim_{E/H}$ and \mathfrak{S} ($r_s = 0.7174$, (two-sided) $p = 0.0086$) which is significant at $\alpha = 0.01$ even for the uncalibrated model with $w_i = 1$. In contrast, the Tversky similarity sim_R does not exhibit a significant relationship. Also, $sim_{E/H}$ shows stronger

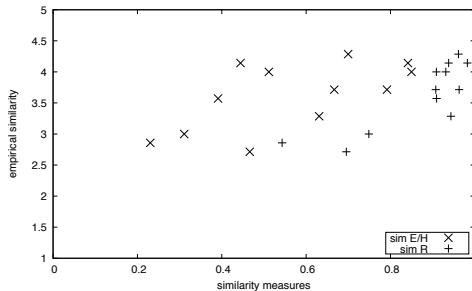


Figure 7.9: Similarity measures and perceived similarity scatterplot

differences between the UIs compared to sim_R , indicating that the Euclidean and Hamming formulations of similarity are a more suitable tool for Web Migration purposes. This can also be seen in fig. 7.9.

Analysis of the empirical data shows rather low values for difficulty, which can be reasoned with the relatively simple user interfaces and the test subjects' proficiency with computers. A moderate correlation between Like and \mathfrak{S} ($r_s = 0.5753, p = 0.0504$) significant at $\alpha = 0.6$ was found, which can imply a preference for familiar interfaces. The Like evaluation expectedly has the largest standard deviation ($\sigma(\text{Like}) = 1.299$) as it is the most subjective criterion. Analyzing the influence of the three distances in pair-wise relations to \mathfrak{S} , the following linear model was found highly significant at $\alpha = 0.001$ with good fit ($p = 0.001, R^2 = 0.670$): $\mathfrak{S} = 3.92 - 2.14\text{dist}_{\text{ori}}$.

For the calibration process in section 7.3.2, a linear model is assumed. We derive the weight factors from multiple linear regression of the three distances and the median of \mathfrak{S} . The resulting linear model in eq. (7.8) is highly significant at $\alpha = 0.001$ and has good fit ($p = 0.0001, R^2 = 0.919$). The F-Test yields $F = 30.26$ which is $F > f_{\text{crit}}$ with

$f_{\text{crit}} = 15.83$ for $p = 0.001$, 3 degrees of freedom of the numerator and 8 degrees of freedom for the denominator, showing an improved model fit over the intercept-only model hypothesis.

$$\mathfrak{S} = 4.61604 - 3.24281 \text{dist}_{\text{ori}} - 1.27447 \text{dist}_{\text{ord}} - 0.888071 \text{dist}_{\text{den}} + e[t] \quad (7.8)$$

Using the factors of the distances as weights w_i , the similarity measures can be re-calculated as shown in table 7.4. The calibrated similarity measures $\text{sim}_{E,\text{cal}}$ and $\text{sim}_{H,\text{cal}}$ improve the correlation with median \mathfrak{S} to $r_s = 0.8383$, $p = 0.0007$, highly significant at $\alpha = 0.001$. Figure 7.10 shows the results of residuals analysis with good overall alignment. All variance inflation factors are less than $\text{VIF}_{\text{den}} < 1.12$, showing a low multicollinearity of the distances' factors. Table 7.4 shows the calibrated similarity measures, and fig. 7.11 visualizes the calibrated measures and their corresponding residuals QQ plots.

Table 7.4: Calibrated sim measures, maxima highlighted in bold

UI	$\text{sim}_{E,\text{cal}}$	$\text{sim}_{H,\text{cal}}$	$\bar{\mathfrak{S}}$
u'_1	0,9088	0,8999	4
$u''_{1,\text{ori}}$	0,5430	0,4726	2.8571
$u''_{1,\text{ord}}$	0,9611	0,945	4.2857
$u''_{1,\text{den}}$	0,9074	0,8906	3.7143
u'_2	0,9822	0,9822	4.1429
$u''_{2,\text{ori}}$	0,7487	0,6904	3
$u''_{2,\text{ord}}$	0,9431	0,9282	3.2857
$u''_{2,\text{den}}$	0,9093	0,8719	3.5714
u'_3	0,9627	0,9627	3.7143
$u''_{3,\text{ori}}$	0,6952	0,6602	2.7143
$u''_{3,\text{ord}}$	0,9311	0,9048	4
$u''_{3,\text{den}}$	0,9378	0,9378	4.1429

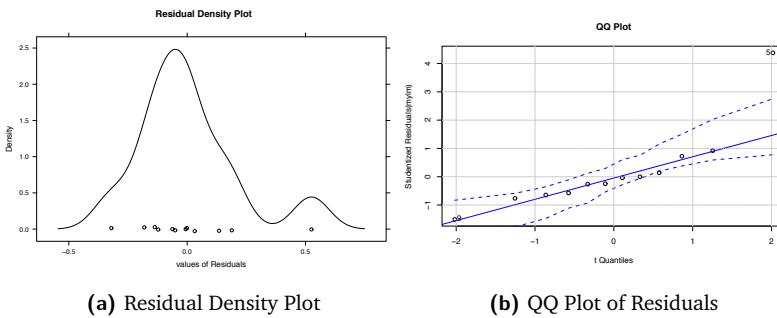


Figure 7.10: Residual Plots, created using (Wessa, 2017)

Threats to Validity. Construct validity of this experiment is threatened by possible variations in the evaluation runs. This was addressed by providing a fixed environment: All sessions were performed on the same PC and screen in order to ensure consistent visual representation and interaction across all test subjects. The task list T_i ensured that all test subjects solved the same tasks on the user interfaces and thus had the same understanding of the underlying Task and Behavior models. The tasks were designed as small and self-contained interactions that did not allow for alternative interaction paths. All test subjects received the same introduction and demonstration of the tasks on the legacy user interface. The second important aspect affecting construct validity is the analysis of Likert items in the questionnaire. In the context of discussions about limitations of valid statistical measures on ordinal-scaled Likert items, table 7.3 follows the practical approach commonly used in the HCI field and user research to apply mean as it better supports smaller sample sizes (Sauro and J. R. Lewis, 2016). However, all interpretations are restricted to ordinal statements; no interval or ratio statements are derived. The correlation analysis was conducted on the median values using Spearman's r_s instead of Pearson's ρ for the same reason.

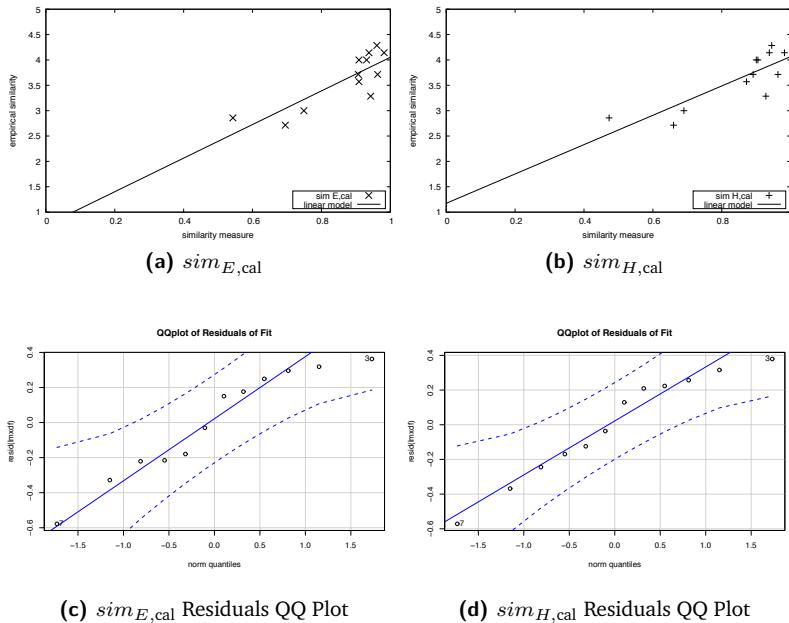


Figure 7.11: Calibrated Similarity Measures and Residuals QQ plots,
created using (Wessa, 2017)

Internal validity of this experiment is threatened through potential subjective biases. In order to avoid participants being biased towards giving higher ratings to the “original” WUIs from U_W in comparison to the variations in U_V , we assigned identifiers not disclosing this information ($u'_1 \hat{=} A3$, $u'_2 \hat{=} B2$, and $u'_3 \hat{=} C2$). Also, the order in which Web versions of the same legacy UI were evaluated was randomized to avoid bias through test subjects adjusting to the same sequence of interfaces. The test subjects’ preference for similar interfaces observed in the Like evaluations may have been affected by the experiment, hinting that similar equals good. However, we measured this data only as additional in-

formation and did not perform further detailed analysis, as the overall preference for similar interfaces through migration has already been well-researched in previous studies. It can be argued that the similarity evaluations of the test subjects are highly subjective. However, this is intended since the measured item, the perceived similarity of UIs, is a subjective notion. This is addressed by the model creation and calibration process, and the good model fit has shown applicability.

External validity of the experiment is threatened through limitations in the generalizability of results. Evidently, the evaluation results cannot be used as a generalized ready-made model for UI Similarity in arbitrary Web Migration contexts, as the similarity evaluations of the test subjects are highly subjective. However, AWSM:CI aims at providing a strategy for creating and adapting a similarity measurement model tailored for a specific target user group of a specific Legacy System to be migrated to the Web. In that way, the evaluation experiment has shown that the AWM:CI method can be successfully applied to achieve this result, without making claims with regards to the generalisability of the concrete numerical results and derived findings.

Conclusion of AWM:CI experimentation. The AWM:CI evaluation experiment has shown that the similarity measures based on Tversky distances in the Orientation, Order, and Density dimensions of layout similarity can be effectively used to estimate the perceived similarity of the user interface in Web Migration. Of the three similarity measures, the Euclidean formulation sim_E and the Hamming formulation sim_H are better applicable than the Tversky mode formulation sim_R . Both similarity measures in their initial state (all weights set to one) fulfill the required concordance assumption. It was shown that the AWM:CI calibration process conducted with only limited effort, i.e. seven users from the target group and three user interfaces with three variations, can

be applied to improve the initial similarity measures to an $R^2 = 0.919$ fit on perceived similarity, highly significant with $p = 0.0001$. Different influences of the proposed distances were observed: while Density was found to be the most influenced by changes in other dimensions and having only small influence on similarity, Orientation showed stronger impact. The concrete findings on differences in the influence of the distances have to be seen in the light of a relatively small group of test users and cannot be generalized. However, AWSM:CI calibration can be applied in order to adjust this to specific target user groups and analyze concrete preferences within these groups. The efficiency of AWSM:CI is evident when comparing to the naive approach of a traditional full empirical evaluation of similarity of all different variations of migrated Web UIs with test subjects. Instead, AWSM:CI provides a method for automated computation of similarity measures as approximation of perceived similarity. These measures can be tailored to the target group with a limited effort calibration process on a small group of representatives from the target group.

7.5.3 Research Results

The research objective addressed by this chapter, RO3, to enable control of the impact of user interface changes through Web Migration on customers with limited resources and lack of Web Engineering expertise, is achieved. The effectiveness, applicability, and calibratability of the AWSM:CI Method for measuring Visual UI Similarity of legacy and Web-migrated user interfaces have been demonstrated and detailed through experimentation. Efficiency of AWSM:CI is currently only half-satisfied due to limitations in the quality of fully automated UI Element detection. However, the overall feasibility of the automation approach could be demonstrated. Expertise requirements are limited through the computable similarity measures, automation, and calibration artifacts and strategy presented above. In spite of the half-restricted efficiency

results, RO3 is considered as achieved. AWSM:CI RQ1 was answered through the UI similarity technique introduced in section 7.3. AWSM:CI RQ2 was answered through specification of a vision-based approach for Similarity Analysis, enabling application on both legacy and Web-based user interfaces, as defined in section 7.3.1. AWSM:CI RQ3 was answered through the realization of UI Similarity Analysis by distance-based computable measures in section 7.3.3 and specifying a solution for automatic detection of UI Elements in section 7.4.

7.6 Summary

This chapter presented the AWSM Customer Impact method, which facilitates the measurement and control of visible change in user interfaces by Web Migration, specifying a novel UI Similarity measurement strategy that can be calibrated to the target user group. The strategy is supported through a computer-vision based UI Element detection and computation of UI layout distances from visual analysis. The conceptual model of the AWSM:CI techniques and their implementation in the Toolsuite have been described. In the evaluation, we have shown the feasibility of the approach and the quality of its measurements with limited effort, improvement of the measurements through calibration, and how to avoid extensive empirical evaluations with only a small number of users. Furthermore, the experiments provided insights on influence factors on perceived similarity contributing to ongoing research in this field. While this research is still in early stages, operationalization of the proposed similarity measure can help controlling customer impact in Web Migration.

Evaluation

8

This chapter evaluates the AWSM approach according to the requirements stated in section 2.3. This assessment is followed by a comparison to the state of the art in terms of benefits and weaknesses. Finally, the application of AWSM in the context of the thesis scenario is outlined.

8.1 Requirements Evaluation

In the following, the AWSM approach is assessed with regard to the scope and stakeholder requirements of this thesis. To conduct this evaluation, each requirement is considered separately against the solutions provided by the AWSM Methodology and their support through the Toolsuite. The same criteria that were used for the assessment of the state of the art of Web Migration approaches in section 3.2 are applied to AWSM. Where applicable, the assessment of requirements refers to the experimentation results presented for each AWSM Method and its corresponding Tools in order to further support the assessment.

8.1.1 Scope Requirements

The following scope requirements are evaluated to determine to what extent AWSM address the scope set in section 1.4, i.e. to what extent it is a Web Migration approach supporting the initial phase of a Web Migration and having Web Applications as target architecture.

S1 Initial Phase Support

The Methodology and Toolsuite provided by AWSM dedicatedly addresses the initial phase of Web Migration prior to actual Transformation and beyond knowledge recovery. Table 4.2 presented a mapping of AWSM Methods onto the phases of the ReMiP reference model, placing AWSM:RE and AWSM:RM in the first phase. While AWSM:RE focuses on knowledge recovery, AWSM:RM goes beyond this. The risk management method based on Rapid Web Migration Prototyping specifies a set of activities that are to be executed even before the final decision whether to migrate to the Web or not has been made. In this early pre-migration phase, AWSM:RM addresses the communication of necessity and benefits of a Web Migration through transfer of the Rapid Prototyping paradigm into the Web Migration domain. In this way, the AWSM:RM method and ReWaMP/RWMPA and UI Transformer toolchain facilitate the creation of web migration prototypes as concrete, tangible demonstrative products contributing to making the migration business case and as a vehicle for communication across stakeholders serving as a basis for decision making. Furthermore, the S2DCS Tool presented in section 4.4.1 facilitates Web Migration strategy selection, an essential activity in the initial phase, based on data from a systematic mapping study comprising scientific publications and software tools. According to the assessment scheme, the requirement is fully satisfied.

S2 Web Application Target

The AWSM Methodology and Toolsuite has Web Applications according to Definition 2 as target architecture. AWM target systems are Web Systems, communicating via HTTP and based on W3C standards like HTML, CSS, WASM, that provide content and services through a Web-based user interface. This can be seen in the architecture of AWM:RM prototypes presented in fig. 6.1 and the comparative analysis of Web user interfaces and their corresponding legacy versions enabled through

AWSM:CI as presented in section 7.3. The target Web-based user interfaces created through AWM:RM are not mere wrappers, but real WUIs, implemented in HTML, CSS, JS, and are run independently of their legacy counterparts. They feature essential Web paradigms, like asynchronous request-response communication, spatial and technological client-server separation, and URL-based navigation of resources. As Web Applications are created based on existing non-Web Legacy Systems in Web Migration, AWM:CI does support joint analysis of non-Web and Web-based user interface through a computer-vision based approach operating on visual features at a cross-platform level of abstraction, enabling consideration of target Web Applications concerning UI similarity with their legacy counterparts. Furthermore, also the tools of the AWM Toolsuite itself are implemented based on open Web standards as reasoned in the design decision of principle P1. According to the assessment scheme, the requirement is fully satisfied.

8.1.2 Stakeholder Requirements

The following stakeholder requirements are evaluated to assess appropriateness of the AWM Methodology and Toolsuite for the characteristics of an independent software vendor, as detailed in section 2.1.

C1 Risk Management

The AWM Methodology and Toolsuite provide risk management methods beyond basic feasibility-centric approaches through its AWM:RE and AWM:RM methods. AWM:RE enables a concept-assignment-based knowledge recovery approach, which addresses the risk of losing knowledge implicitly represented by the legacy source code during Web Migration (cf. chapter 5). The systematic application of AWM:RE Reverse Engineering extracts knowledge from the source code and sustains it for further management and usage in an interoperable way based on the SCKM formalism and a queryable Web standards-based

knowledge representation. The experiment in section 5.6 has shown that ISVs are enabled to conduct this activity through crowdsourcing. Following AWSM principle P4, AWM:RM specifies a rapid-prototyping-based method for creating demonstrative web migration prototypes that allows for identification of migration process and migration result risks (cf. chapter 6). The produced web migration prototypes and gained migration experience not only provide insights on the technical feasibility, but they also represent a concrete and tangible contribution towards the Web Migration business case: they allow assessing the plausibility and thus desirability of a potential Web-based version of the Legacy System which enables an informed balancing of potential business value and cost. The experiments in section 6.6 have shown that these web migration prototypes can be created from base material of the existing Legacy System enabled through the AWM Tools for Rapid Web Migration Prototyping. According to the assessment scheme, the requirement is fully satisfied.

C2 Reuse of Legacy Assets

The AWM Methodology and Toolsuite addresses reuse of legacy assets at three different levels, covering the conceptual model, view and controller layers of an application: AWM:RE focuses on reuse on the model and requirement level, AWM:RM on the business logic and UI level and AWM:CI on the view and user interaction level. In this way, the three AWM Methods establish continuity of functionality (AWM:RE and AWM:RM) and user interaction (AWM:RM and AWM:CI) across legacy and target Web System. The reuse of legacy models and requirements is achieved through AWM:RE by reverse-engineering these assets implicitly represented in the codebase into explicit artifacts, thus enabling their reuse in both model-driven and non-model-driven subsequent processing. The reuse of business logic and the user interface is achieved through AWM:RM by semi-automatic (ReWaMP) / automatic (UI Transformer) Transformation based on legacy code artifacts,

as shown in the experiments in section 6.6. The reuse of view layout and user interaction is achieved through AWSM:CI by enabling a joint analysis of legacy and Web user interfaces based on corresponding user interface artifacts. The experiment in section 7.5 has shown that the proposed similarity computations can act as approximations of perceived similarity, ensuring continuity between legacy and Web user interfaces. The KDM-based Legacy System, SCKM, and Legacy User Interface formalisms provide the conceptual foundation of reuse in AWSM. According to the assessment scheme, the requirement is fully satisfied.

C3 Expertise & Tool Support

The AWSM Methodology and Toolsuite address expertise and tool support requirements through a method design integrating the non-functional expertise requirement for all three AWSM Methods and the tools of the AWSM Toolsuite, respectively. AWSM:RE addresses the expertise requirement by reusing program comprehension results from ongoing Forward Engineering and by automatic decomposition of Concept Assignment into microtasks solved using crowd expertise. The experiment in section 5.6 has demonstrated that the crowd worker's expertise and results quality is sufficient to reduce effort and expertise requirements on ISV staff by outsourcing it to the crowd. AWSM:RE tool support comprises the Annotation Platform, its CSRE extension, and the tools for integration with ongoing development. AWSM:RM addresses the expertise requirement by enabling business logic reuse based on existing staff expertise in the legacy platform supported by the ReWaMP toolchain, lowering the required Web Engineering and migration expertise demand through its semi-automatic process supported by the extensive RWMPA guidance and avoiding Web Engineering requirements for UI creation through the fully automatic UI Transformer without manual interventions. The experiments in section 6.6 have demonstrated feasibility of AWM:RM with limited migration and Web

Engineering expertise. AWSM:CI addresses the expertise requirement by specifying calibratable similarity measures that can be computed and empirically adjusted by existing ISV staff with fundamental statistics expertise based on automated visual UI Element detection and with non-expert test subjects form the target group as demonstrated by the experiment in section 7.5. The AWSM Toolsuite provides tool support for all three AWSM Methods enabling semi-automatic or automatic processes, fulfilling the tool aspect of the requirement. As AWSM cannot entirely avoid expertise demand in new areas but is still feasible with existing staff, the expertise aspect of the requirement is conditionally met. Thus, the overall requirement is assessed as half satisfied, acknowledging that Web Migration activities can hardly be conducted without any additional expertise requirements.

C4 Agile Development Process Integration

The AWSM Methodology and Toolsuite address development process integration through a method design specifying the conceptual integration of each AWSM method into ongoing development activities and the integration support tools of the AWSM Toolsuite. AWM principle P2 avoids specification of a stand-alone process: the three AWM Methods targeting the initial phase can be integrated easier than full-coverage Web Migration approaches. AWM principle P3 facilitates integration with a variety of different model-driven or non-model-driven development processes. AWM:RE integrates with ongoing development through continuous Reverse Engineering, embedding Concept Assignment with Forward Engineering in the comprehension phase, reusing the mental representation. This is supported by the IDE integration tool. AWM:RE integrates Web Migration project management with software project management via the migration package/migration backlog formalism and the corresponding project management tool integration. AWM:RM integrates with ongoing development through transfer of the Rapid Prototyping paradigm into the Web Migration domain, embedding

AWSM:RM activities with ongoing explorative or experimental Prototyping, with resulting web migration prototypes as product increments achievable within one Scrum sprint. AWSM:CI integrates with ongoing development in the context of software quality measurements and user interface design activities within ongoing Forward Engineering UIX analysis and approval activities of a dedicated team of UIX experts. As AWSM specifies integration with ongoing development for all three methods but introduces activities and artifacts that, depending on the maturity of the ongoing forward development process, can be new, the development process integration is conditionally met. Thus, the overall requirement is assessed as half satisfied, acknowledging that Web Migration activities can hardly be entirely integrated with ongoing forward development.

Table 8.1 shows the overall evaluation results of AWSM.

Table 8.1: AWSM Evaluation

S1 Initial	S2 Web	C1 Risk	C2 Reuse	C3 Exp	C4 Agile
●	●	●	●	○	○

8.2 Comparison with State of the Art

Table 8.2 shows AWSM in comparison to the approaches assessed in section 3.2. Concerning support of the initial phase, AWSM is among the very few to fully address this requirement. Similar to AWS Migration and SMART, AWSM has a dedicated focus on the phases prior to migration, but it is not limited to planning, providing concrete solutions for Legacy and Requirements Analysis, Target Design and Implementation covering Configuration & Change Management, Migration Environment and Staff Qualification (cf. table 4.2). Unlike SMART, AWSM addresses

the communication of benefits of Web Migration through demonstration of desirability and unlike ARTIST with tangible means. All three AWSM Methods contribute to the migration decision point specified by various Web Migration approaches like AWS Migration, ARTIST, REMICS, SMART and SAPIENSA. AWSM's S2DCS supports the Strategy Selection, which is an essential step in the initial ReMiP phase with a faceted search interface over a database of 122 approaches and tools. This is only addressed in the meta-approaches of AWS Migration and SMART. S2DCS provides concrete decision support in contrast to the abstract guidelines in AWSM Migration and SMART.

Table 8.2: Evaluation of AWSM in Comparison to State of the Art, requirements S1 Initial phase, S2 Web Application Target, C1 Risk management, C2 Reuse of legacy assets, C3 Expertise & Tool Support, C4 Agile Development Process Integration

Approach	Target	Method	S1	S2	C1	C2	C3	C4
SMART	SOA	N/A	○	○	●	○	●	○
SAPIENSA	SOA	Trans	○	○	○	○	○	○
serviciFi	SOA	ReEng	○	○	●	○	○	○
Marchetto2008	SOA	Encaps	○	○	○	●	●	○
SOAMIG	SOA	Trans	○	○	○	○	○	○
Gaps2Ws	SOA	Encaps	○	○	○	○	●	○
PRECISO	SOA	Encaps	○	○	○	○	●	○
AWS Migration	Cloud	N/A	●	○	○	○	○	○
REMICS	Cloud	Trans	○	●	○	○	○	●
ARTIST	Cloud	Trans	●	○	●	○	○	○
CloudMIG	Cloud	Trans	○	●	○	○	○	○
IC4	Cloud	ReEng	○	●	○	○	○	○
AMS	Cloud	Encaps	○	○	○	●	●	○
NCHC	Cloud	Encaps	○	○	○	●	●	○
L2CMH	Cloud	Trans	○	○	○	○	○	○

Approach	Target	Method	S1	S2	C1	C2	C3	C4
MIGRARIA	WSE	Trans	○	●	○	●	○	○
MigraSOA	WSE	Trans	○	●	○	●	○	○
MELIS	Web	Encaps	●	○	●	●	○	○
TUIMigrate	Web	Trans	○	○	○	●	●	○
M&S SW	Web	Encaps	○	○	○	●	●	○
UWA/UWAT+	Web	ReEng	●	●	○	●	○	○
CelLEST	Web	Encaps	○	○	○	●	○	○
DAS	Web	Encaps	○	○	○	●	●	○
AWSM	Web	ReEng	●	●	●	●	○	●

AWSM belongs to the group of Web Migration approaches with a target architecture of a Web Application, covering all three layers of a typical three-tier-architecture, in particular including consideration of the Web-based user interface. This consideration is also found in REMICS, CloudMIG, IC4, Migraria, MigraSOA, and UWA/UWAT+. REMICS, CloudMIG and IC4 focus on a cloud-based backend, Migraria and MigraSOA are WSE approaches where the source system is already a Web System. REMICS, CloudMIG, and UWA/UWAT+ are model-driven approaches with user-interaction-focused Transformations of the presentation model, IC4 targeting SaaS implies a Web-based UI but does not provide concrete technique descriptions. In contrast, AWSM is model-driven agnostic (cf. principle P3), and defined for migration of non-Web Legacy Systems towards Web Applications according to Definition 2, which can be cloud-based SaaS systems, but also distributedly hosted intra-net applications based on Web technologies and hybrid cloud applications.

Regarding risk management, the web migration prototypes of AWSM:RM fulfill a similar function like migration pilots in several approaches like AWS Migration, REMICS, SMART, IC4, however, AWSM emphasizes the value of demonstration of desirability in addition to

technical feasibility, and unlike these approaches, AWSM:RM provides concrete and comprehensive techniques for the rapid creation of web migration prototypes. Risk management is only fully addressed by three other Web Migration approaches: SMART, serviciFi and ARTIST. Similar to AWM, the SMART methodology comprises several techniques for risk management, including feasibility assessment, an incremental process model, and migration pilots, so the complementary risk management techniques of AWM:RM can be easily integrated with SMART. The serviciFi approach manages risk through a combined analysis of technical feasibility and economic viability using portfolio analysis. The cost-benefit map is based on return on investment estimates considering maintenance cost and business value in relation to market needs. In contrast, AWM:RM provides not only a proof of technical feasibility but also a more concrete demonstration of business value and plausibility of a Web-based solution which, combined with information recovered through AWM:RE, can be used to enhance the portfolio analysis technique of serviciFi. ARTIST's technical and economic feasibility assessment can benefit from the integration of AWM:RM in a similar way. Like ARTIST, AWM uses extensive knowledge recovery to address the risk of knowledge loss. While AWM is agnostic to the specific nature of target knowledge representations, enabled by AWM's KDM-based SCKM and the queryable knowledge representation of AWM:RE, ARTIST's extensive model-driven knowledge recovery can be supported by AWM:RE as described in section 5.3.

Reuse of functionality is expectedly high across all Web Migration approaches — all approaches but AWS Migration and PRECISO feature functionality reuse— as this is a definitive property of any software migration approach. The distinctive property is reuse-based continuity of user interaction. AWM addresses reuse on all three application layers as outlined above in order to maintain functionality and user

interaction. A similarly high level of reuse is observed mainly in Encapsulation approaches (Marchetto2008, AMS, NCHC, MELIS, M&S SW, CelLest, DAS) due to their Web Migration without modernization nature (cf. section 3.3) and WSE approaches (MIGRARIA, MigraSOA) which already start with a Web-based source system. Only one other Transformation approach, TUIMigrate, and one Reengineering approach, UWA/UWAT+, also achieves a full rating for reuse. TUIMigrate, in contrast to AWSM, focuses on text-based terminal user interfaces. Thus achieving continuity in user interaction and a similar layout is simpler but comes at the cost of emulating an outdated terminal-based interaction in the Web browser. AWSM, in contrast, addresses user interface continuity in AWSM:RM and AWSM:CI for graphical user interfaces both on the source and target side of the migration. The conceptual user-centered modeling of UWA/UWAT+ is the closest to AWSM in its emphasis on user interface continuity. While UWA/UWAT+ focuses on similarity in the Task and Behaviour dimension, based on model-based hypermedia re-design of business process tasks into content navigation models, AWSM focuses on the Layout dimension. Thus, UWA/UWAT+ and AWSM are complementary and can be integrated into the context of Web Migration towards a model-driven UWA-based architecture. The other comprehensive Web Migration approaches like REMICS and ARTIST are falling behind with regard to reuse, as they only consider reuse of functionality, but can be easily extended with AWSM techniques due to their structure as methodological frameworks.

Concerning expertise and tool support, AWSM only receives a half rating, conceding that in spite of the extensive tool support of the AWSM Toolsuite, the Web Migration activities of the AWSM Methods can hardly be conducted without any additional expertise requirements. In contrast, nine Web Migration approaches received a full rating. However, Marchetto2008, Gaps2Ws, PRECISO, AMS, NCHC, M&S SW, and DAS

are Encapsulation approaches that cannot be considered equivalent full Web Migration approaches targeting real Web Applications independent of the source system. Thus the mainly tool-based wrapper approaches pose low expertise requirements. The same holds for TUMigrate due to its focus on terminal user interface migration. The SMART methodology outperforms AWSM in terms of expertise requirements due to its simple, well-defined process and activities along with comprehensive templates, guidance documentation, and tools. This advantage, however, is to be seen in the light of SMART's different scope limiting its applicability as described for other requirements above. Expectedly, AWSM outperforms the comprehensive Web Migration approaches like REMICS, ARTIST and UWA/UWAT+ with regard to expertise requirements. This, however, has to be considered with the same fairness regarding their significantly broader and more complete scope: while AWSM aims at providing a methodology and toolsuite addressing shortcomings of existing Web Migration approaches, REMICS, ARTIST and UWA/UWAT+ aim at specifying complete Web Migration approaches at the extent of comprehensive EU-funded research projects. Interestingly, the IC4 approach was designed for a very similar SME-sized ISV stakeholder like AWSM, but according to the assessment scheme is rated lower than AWSM due to the required manual Reengineering and lack of tool support.

Integration of Web Migration activities and artifacts into ISV's ongoing agile development was observed the requirement hardest to achieve in section 3.2. While 22 out of 23 assessed approaches do not address this requirement, defining Web Migration activities and artifacts in an isolated, stand-alone process manner, only REMICS agile extensions define a mapping of REMICS to Scrum that, even though not explicitly mentioned, facilitates integration with ongoing agile development. AWSM consistently specifies integration for all three methods of the AWSM Methodology, and the AWSM Toolsuite provides several dedicated in-

tegration tools. Yet, the development process integration requirement is only conditionally met as the measurement activities and artifacts of AWSM:CI can only be integrated straight-forward if the ongoing agile development process already makes use of differential software quality measurements between user interfaces. Due to this restriction, AWSM is rated equal to REMICS, but higher than all 22 other approaches, which ignore the integration aspect as important factor of facilitating Web Migration initiation for SME-sized ISVs.

As shown in table 8.2, despite its limitations in two of the four stakeholder requirements with SHOULD-priority, AWSM's overall assessment is higher than any existing Web Migration approach. This holds for both the summative and average scoring¹²⁵, where AWSM ($\Sigma = 5$, $\bar{S} = 0.833$) is rated higher than the top-ranking approaches in section 3.2 SMART ($\Sigma = 3.5$, $\bar{S} = 0.583$), REMICS ($\Sigma = 3$, $\bar{S} = 0.5$), ARTIST ($\Sigma = 3$, $\bar{S} = 0.5$) and UWA/UWAT+ ($\Sigma = 2.5$, $\bar{S} = 0.417$). AWM, however, does not directly compete with complete migration processes as defined for instance by REMICS, ARTIST or UWA/UWAT+. Instead, similar to SMART, AWM is a *complementary methodology* providing a set of principles, formalisms, and methods supported by the tools of the AWM Toolsuite, which address aspects of Web Migration that have received little to no attention in existing complete approaches. It is therefore meant to be used in conjunction with one of the complete Web Migration approaches as suitable for the concrete migration situation and environment (specific target architecture, available resources, model-driven or non-model-driven development process, etc.), selected through S2DCS, which defines the overall migration procedure (cf. fig. 4.3).

¹²⁵based on a mapping S of the three possible ratings as follows: $\circlearrowleft \mapsto 0$, $\bullet \mapsto 0.5$, $\bullet\bullet \mapsto 1$

8.3 Application Scenarios

The applicability of AWSM has been tested in the context of the scenario described in section 2.1. Below, three application scenarios are described, relating to the three methods of AWSM, where AWM techniques and tools were applied in the ISV scenario context. The first one describes the application of the Reverse Engineering model SCKM and the Annotation Platform on the production codebase of medatixx. Application scenario two presents the use of Rapid Web Migration Prototyping for the medical appointment scheduling scenario application. The last application scenario reports on the application of Layout Similarity measurements and the calibration process used for UI Similarity Analysis between user interfaces of the Desktop Application Patient Management System and Web versions of the same user interfaces.

8.3.1 Reverse Engineering and Annotation Platform in ISV Production Codebase

For testing of the AWM Reverse Engineering Method, we had access to the production codebase of the ISV medatixx. Using static analysis tools, we analyzed the characteristics and structure of the codebase, as described in section 2.1.2. The SCKM model was applied to code samples of the scenario codebase to verify its capability to model arbitrary knowledge in legacy source code artifacts and the modeling of knowledge types, as shown in the left sub-tree of fig. C.2, was iteratively tested and improved on the codebase. The resulting fine-grained distinction of eight detailed knowledge types has shown to be exhaustive on the scenario codebase for describing valuable knowledge implicitly represented by it.

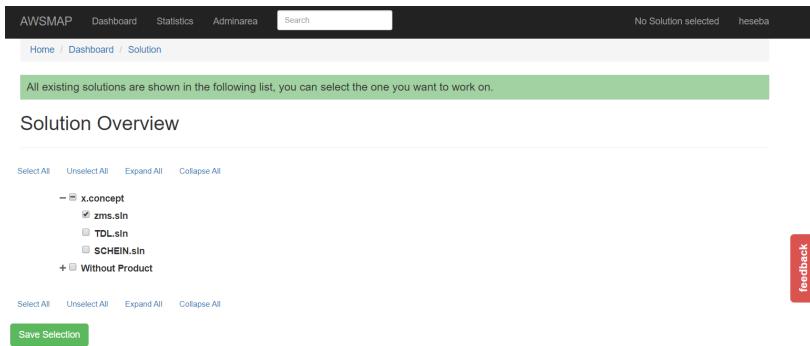


Figure 8.1: Solution Picker View in medatixx Version of Annotation Platform

A fork of the Annotation Platform described in section 5.4 was created with specific adaptions to the environment of medatixx. This version was deployed at medatixx and connected to its MS Team Foundation Server instance. From this TFS, the codebase was loaded into the Annotation Platform through a dedicated batch import implementation created to accelerate loading of the large amount of software artifacts contained in the production code. Another adaption in the medatixx-specific fork of the Annotation Platform is the product/solution switcher. It allows to temporarily limit the scope of the software artifacts represented in the Annotation Platform to specific software products or MS Visual Studio solutions. The medatixx instance of the Annotation Platform was tested by volunteers from the software development department and the software architect. Figure 8.1 shows the solution picker view of the medatixx-specific version of the Annotation Platform. The products and solutions listed can be used to filter the artifacts shown in the Annotation Platform. The red button on the right was implemented to support feedback, e.g. on bugs or feature requests, and keeps track of the viewing context in which the feedback was sent. The logged-in user shown on the top right of the screen is authenticated against an Active Directory Domain controller.

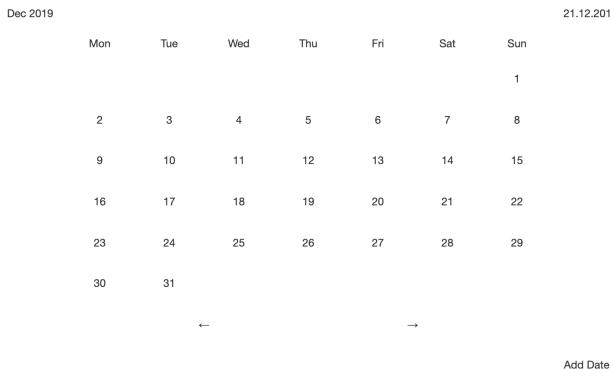


Figure 8.2: Calendar View of Web Migration Prototype with WASM-based Business Logic reused from Legacy System

8.3.2 Rapid Web Migration Prototyping of Medical Appointment Scheduling

The testing of AWSM:RM was based on the medical appointment scheduling scenario described in section 2.1.2. As outlined in table 2.1, it provides basic functionality for management of appointments in resident doctor's offices, including the creation, modification, deletion, and viewing of existing appointments in different calendar views and the search for free timeslots to allocate new appointments into. To create a web migration prototype of the core functionality of the scenario application, we applied the WASM-based AWSM:RM business logic reuse technique, building a demonstrative prototype that provides calendar view, appointment entry and free timeslot search functionality executing legacy business logic modified through and supported by the ReWaMP tool. Figure 8.2 shows a screenshot of a web migration prototype, featuring a calendar view with WASM-enabled business logic taken from the medical appointment scheduling scenario application.

The important characteristic of this prototype is the reuse of parts of the legacy business logic. As shown in the example in listing 8.1 these parts are exported into JavaScript from the C++ code, which is compiled to WASM and then, as shown in listing 8.2 they can be bound in JavaScript to handle the click events in the Web user interface, redirecting to the compiled WASM module.

Listing 8.1: Example of C++ Code Added by ReWaMP to Export Legacy Functionality to JavaScript

```
EMSCRIPTEN_BINDINGS(rwmp_binds) {
    using namespace emscripten;
    class_ <Cal01_View>("Cal01_View")
        .constructor()
        .function("OnOK", &Cal01_View::OnOK)
        .function("OnBnClickedButtonAdddate",
                  &Cal01_View::OnBnClickedButtonAdddate);
}
```

Listing 8.2: JavaScript Binding EventHandler of Button to Exported Legacy Functionality

```
function rwmp_init() {
    rwmp = new Module.Cal01_View();
    rwmp.OnInitDialog();
    ...
    document.getElementById("IDC_BUTTON_ADDDATE")
        .addEventListener("click", function() {
            rwmp.OnBnClickedButtonAdddate();
        });
}
```

The UI Transformer tool was tested on MFC-based legacy user interfaces from the scenario application to automatically create Web versions of the user interfaces. Based on these UI prototypes, additional CSS styling resulted in user interfaces like the calendar shown in fig. 8.3.

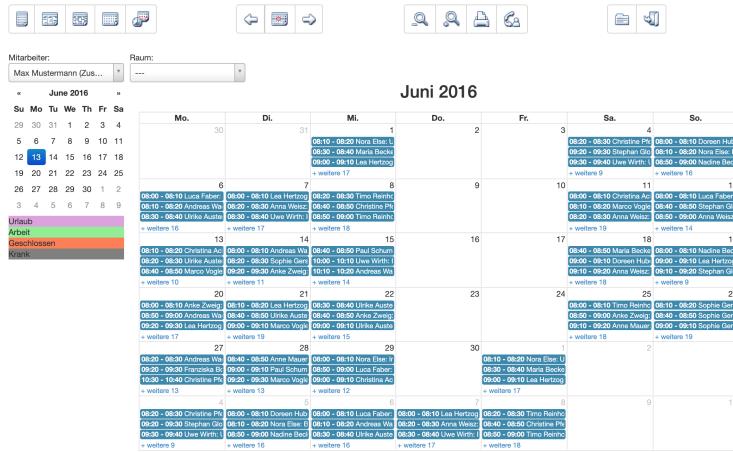


Figure 8.3: Web Version of Calendar User Interface of Scenario Application

8.3.3 Layout Similarity Measurement in x.concept

The AWSM:CI Layout Similarity Measurement was tested on user interfaces of the medatixx x.concept Desktop application described in section 2.1.2. For three different views – a graphical shift schedule, a calendar view, and a patient data form – Web versions of the user interfaces were systematically altered in layout dimensions as described in section 7.3.2. The test subjects were acquainted with the original desktop user interfaces of x.concept and then evaluated the similarity with the different variations of their Web versions. The results of this empirical experiment were compared with the objective similarity measures proposed in section 7.3.3 to verify their applicability for comparisons of

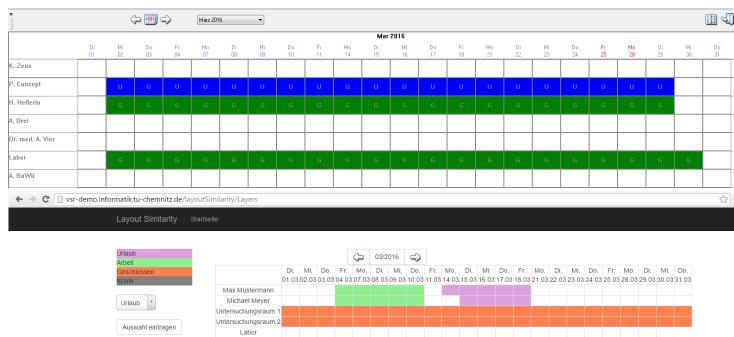


Figure 8.4: Legacy Desktop (top) and Web (bottom) user interface from ZMS, used in Similarity Measurement Experiments

real legacy user interfaces and their Web versions and to demonstrate the calibration process. Figure 8.4 shows the user interface of the graphical shift schedule. The legacy desktop user interface shown on top was analyzed comparatively against different Web-based versions like the one shown on the bottom, and the results of similarity computation were compared with empirical similarity evaluation results.

8.4 Summary

While previous sections evaluated the AWSM Methods and techniques in isolation, this section analyzed the extent to which the AWSM Methodology and Toolsuite as a whole satisfy the requirements for a Web Migration solution addressing the shortcomings of existing approaches as elicited in section 3.4. The analysis showed that all scope requirements and all stakeholder requirements except Expertise and Process Integration are fully satisfied. The latter two requirements are partially satisfied acknowledging the specificity of Web Migration and its activities, which cannot be conducted entirely without additional expertise requirements and completely integrated with ongoing development

except for simple Encapsulation approaches or approaches with limited scope. The chapter has compared AWSM with the state of the art in the Web Migration field, showing that AWSM Methods and tools successfully address gaps in existing comprehensive Web Migration approaches. A brief overview of application of AWSM techniques and tools in the context of the ISV scenario was given.

Conclusion and Outlook

9

Concluding the thesis, this section summarizes the results and contributions, reflects on problems that were not addressed or require further investigation and points out open research challenges.

9.1 Thesis Summary

Initially motivated by the observed difficulties of several ISVs to commence Web Migration in the context of industrial research collaboration projects, this thesis systematically investigated the problem domain. It outlined the reasons for Web Migration on the one hand, and the complexity of the initial legacy situation on the other hand, identifying effort and risk as the two main obstacles and deriving three corresponding research questions that define the scope of this thesis. We detailed the situation of medatixx as representative sample of a capable modern medium-sized ISV struggling to bring its non-Web Legacy Systems to the Web, analyzing characteristics of company, development, and Legacy Systems as well as migration objectives. Based on a systematic field research and analysis process, requirements for an appropriate solution have been elicited, which detail the research questions and represent the problem analysis findings from the ISV stakeholder context. Analysis of the state of the art based on a systematic mapping study revealed a lack of approaches supporting the initial phase of Web Migration to address concerns about effort and risk that are tailored to the char-

acteristics of SME-sized ISVs, despite academia having shifted focus away from Web Migration. These shortcomings were further detailed as three main gaps in Web Migration research. To close these gaps, an HCD ideation method was applied to create three specific research objectives under the overarching goal to support ISV in Web Migration initiation with limited resources and expertise.

The proposed solution is AWSM, which provides a methodology for supporting ISVs to commence Web Migration. AWSM specifies principles, formalisms, methods, and tools which are complementary to existing comprehensive Web Migration approaches and address their identified shortcomings. The AWSM principles base it on open Web standards, shape it as a methodology for integration with Web Migration approaches on different degrees of model-driven adoption, and advocate the use of Rapid Prototyping. The AWSM formalisms provide the conceptual basis ensuring interoperability through mathematical modeling and consistent mapping to the KDM OMG standard. The three AWSM Methods each address the gaps of existing approaches that contribute to ISV's doubts about feasibility and desirability:

- AWSM:RE allows to identify and maintain existing valuable knowledge through crowdsourced Concept Assignment supported by a Web-based annotation platform.
- AWSM:RM minimizes risk through migration pilots and demonstrates desirability and feasibility of a potential Web-based version of the Legacy System applying the Rapid Prototyping paradigm to Web Migration.
- AWSM:CI allows to control the impact of Web Migration on customers through measuring visible changes in the user interface.

The AWSM Toolsuite comprises tools that support these methods and the overall AWSM approach through automated or semi-automated

Transformations, analysis, integration with existing development and management software, and providing queryable standards-based representations. In particular, the AWSM Strategy Selection Decision Support System facilitates ISV's specification of an overall Web Migration strategy through faceted scenario-based search based on the data from our comprehensive systematic mapping study comprising 122 published Web Migration approaches and software tools.

The AWSM Reverse Engineering Method achieves the research objective of identification and management of existing knowledge in legacy source code with limited resources and a lack of Web Engineering expertise by facilitating recovery of problem and solution domain knowledge from the legacy codebase for ISVs through a novel crowdsourced Concept Assignment strategy. AWSM:RE integrates with ongoing development as well as with other Web Migration methods leveraging a queryable open Web standards-based knowledge representation. The AWSM Risk Management Method achieves the research objective of demonstration of desirability and feasibility of a potential Web-based version of the Legacy System with limited resources and lack of Web Engineering expertise by enabling the creation of a Web-based version of the Legacy System through a novel Rapid Web Migration Prototyping strategy. The AWSM Toolsuite supports this strategy with a WebAssembly-based toolchain and a guidance and automation system. The AWSM Customer Impact method achieves the research objective of control of the impact of user interface changes through Web Migration on customers with limited resources and lack of Web Engineering expertise by enabling the measurement of visible change in user interfaces resulting from Web Migration through a novel UI Similarity measurement strategy. The measurements can be calibrated to the target user group and are supported through a computer-vision based UI Element detection and computation of UI layout distances from visual analysis.

Effectiveness and applicability of the AWSM Methodology and Toolsuite has been evaluated in five different experiments combining empirical and objective data. The experiments demonstrated the feasibility of the proposed techniques with limited resources and limited Web Engineering and migration expertise, as well as the quality of the achievable results. The support tools of the AWSM Toolsuite were consistently found useful by the test subjects. The main problems identified under experimentation were task complexity, addressed by an additional guidance and automation system, and the non-mature nature of understanding perceived similarity of user interfaces, which is still a young field with ongoing research. Analysis of the AWSM Methodology and Toolsuite in the context of stated requirements confirmed that the original objectives have been achieved.

9.2 Lessons Learned

The research conducted for this PhD thesis yielded findings that are not specific to the AWSM Methodology and Toolsuite but relevant for research in the Web Migration field and related areas in general:

- In academia, the interest in core Web Migration topics has significantly dropped, with the majority of published approaches being of limited relevance due to outdated target Web technologies (cf. Heil and Gaedke, 2017). Recently, the research focus shifted towards more modern technologies like Cloud. However, many ISVs still struggle with basic Web Migration and cannot target the Cloud. The lack of research interest and resulting unavailability of solutions intensifies this problem.
- A reconsideration of Web Migration in the light of new technologies like WebAssembly can be fruitful, as they allow different

migration paths due to opening up the formerly restricted technological environment of the Web towards various source technologies. (cf. Heil, Siegert, et al., 2018)

- The transfer of paradigms that have been successful in forward software and Web Engineering to Web Migration, such as Crowd-sourcing (Heil, Siegert, et al., 2019; Heil, Förster, et al., 2018), Rapid Prototyping (Heil, Siegert, et al., 2018) and Visual UI Analysis (Heil, Bakaev, et al., 2016) can provide new impetus.
- Despite fake contributions even in a limited group of crowdworkers, suitable quality control measures can effectively ensure result quality and some crowdworkers exhibit an unexpected degree of active commitment and investment of time to provide good result quality (Heil, Siegert, et al., 2019; Heil, Förster, et al., 2018).
- The effort for complex Web Migration activities can be reduced through improved process guidance, partial automation, and heuristic suggestions, which is measurable in both required time and empirical evaluation (cf. Section 6.6.3).
- Perceived similarity of user interfaces is important for Web Migration and influence factors like Order, Orientation and Density constitute a very basic, applicable model (Bakaev, Heil, Khvorostov, et al., 2019; Heil, Bakaev, et al., 2016), but research in this field is still very young and ongoing.
- While a high degree of integration of Web Migration activities with ongoing agile development is a crucial feasibility factor for ISVs, it has to be acknowledged that full integration is an ideal that cannot be achieved (cf. Section 8.1.2). As a requirement for Web Migration, however, it is valuable since it has not been considered in research yet.

9.3 Detailed Contributions

This thesis produced models, methods, techniques, architectures, and tools that support ISVs to commence Web Migration. The following list provides a more detailed view of the contributions outlined in section 1.5 by summarizing all research contributions of the thesis:

- Specification of a dedicated Web Migration initiation methodology for SME-sized ISVs with non-Web Legacy Systems and large user bases.
- Systematic problem analysis of the main stakeholder situation and identification of research objectives through a field-research-, HCD- and LFA-based research process.
- Systematic mapping study of both academic publications and software tools in the Web Migration field with a focus on the SME perspective.
- Development of a Decision Support System facilitating Web Migration strategy selection for SMEs through scenario-based, faceted search based on the systematic mapping data.
- Definition of a role and process model for Reverse Engineering through Concept Assignment integrated into ongoing development and specification of a support toolsuite architecture integrated with ISV's software infrastructure.
- Definition of a queryable open Web standards-based representation of knowledge in legacy codebases through ontological modelling and specification of a storage and querying knowledge base architecture.
- Specification of a novel Crowdsourced Reverse Engineering strategy through transfer of the Crowdsourcing paradigm to the Reverse Engineering domain by re-formulation of Concept Assignment as classification problem.

- Specification of a novel Rapid Web Migration Prototyping strategy through transfer of the Rapid Prototyping paradigm to the Web Migration domain by leveraging current open Web standards.
- Specification of a novel calibratable UI similarity measurement strategy through modeling computable similarity measures by calculation of visual layout distances between non-Web and Web user interfaces.
- Insights on perceived similarity and its objective and subjective impact factors in the context of ongoing research on visual UI similarity perception.
- Systematic empirical experimentation with the proposed methods and tools combining both subjective and objective measures for analysis of effectiveness, applicability, and result quality.

9.4 Ongoing and Future Work

The AWSM Methodology and Toolsuite addressed three crucial challenges related to initiation of Web Migration for ISVs. The proposed methods and tools aimed at lowering the initial barrier originating in effort and risk related to Web Migration by dedicatedly addressing doubts about feasibility and desirability. Legacy knowledge recovery, demonstration of feasibility, desirability and plausibility of a Web-based version of the Legacy System and customer impact control through UI similarity measurements are enabled by various AWSM techniques. The tools of the AWSM Toolsuite enable ISVs with limited resources and limited Web Engineering and Web Migration expertise to successfully apply these techniques. Future work should focus on enhancing the efficiency and scope of the proposed mechanisms, explore other issues under the same motivation that are not addressed by AWSM, and further investigate the perception of user interfaces.

9.4.1 Methodology and Toolsuite Improvements

The AWSM Methodology and Toolsuite can be improved with regard to its functional scope and efficiency in several ways. The AWSM Reverse Engineering Method based on Concept Assignment specifies a generic and queryable representation of knowledge $k = (t, r)$ (cf. eq. (4.2)) and its location l in the codebase facilitated by the SCKM Ontology and SPARQL Endpoint. Due to principle P2 and P3 no restrictions or further specifications are made for the internal representation r . Increasing the scope of AWSM:RE would allow putting more emphasis on the extraction process from extension l to intension k , and a more detailed specification of r . This can be achieved in different ways: Combining AWSM:RE Concept Assignment with automatic Reverse Engineering methods can make use of the knowledge type information t to run dedicated knowledge extractors for the specific knowledge type on the related extensions, benefiting from AWSM:RE Concept Assignment similar to the classifiers running on previously detected ROIs in section 7.4.2. Integration with comprehensive Web Migration approaches specifies r according to the specific models required, e.g. UWA models, and adding model-specific extraction processes. The third option is to extend our experiments on Crowdsourced Reverse Engineering towards the extraction of specific problem and solution domain knowledge representations, e.g. UML diagrams of persistence models or BPMN diagrams of business processes, by the crowd, for instance through microtasking with a more comprehensive classification ontology specific to Legacy System \mathcal{L} . This requires more research to transfer the benefits observed in CSRE and adapt the quality control measures to the new activity type.

Within CSRE, balancing controlled disclosure with readability is a challenge. Algorithm 5.1 presents a simple anonymization technique, but AWSM:RE would benefit from a more sophisticated algorithm that im-

proves crowdworkers' code comprehension while maintaining the three anonymization properties defined in section 5.5.2. For Crowdsourced Reverse Engineering, quality control is crucial. We used majority consensus to aggregate results, treating individual crowdworker results as votes. To analyze agreement, Entropy E and normalized Herfindahl dispersion measure H^* were used. Agreement measures can approximate result confidence and thus filter/flag crowd results with low agreement across crowdworkers. A research challenge is to identify agreement measures that handle split votes¹²⁶ better, e.g. using Fleiss Kappa, and define an extended confidence-based quality control mechanism.

Visualizations as external representation of Knowledgebase \mathbb{K}_B as described in section 5.4 facilitate achieving the high level of code comprehension required for Web Migration. AWSMAP provides different progress statistics as shown in fig. 5.8 and visualizations of concept relationships: Sankey diagrams, as shown in fig. 5.9, Force Graphs and Chord diagrams, which represent the interlacing between artifacts and features or features and features. Improvement of these visualizations requires further research, including empirical studies on their usability and specification of better visualizations including also domain knowledge types in addition to features, integration of the visualizations into developer tools, and specification of context-adaptable visualizations capable of handling high volumes of artifacts and knowledge in production-grade knowledge bases.

The Rapid Web Migration Prototyping technique of AWSM:RM is based on the emerging WebAssembly standard. With Mozilla's ongoing development of the *WebAssembly System Interface* (WASI)¹²⁷, WASM is

¹²⁶ E and H^* rate a distribution like $(4, 1, 1, 1, 1)$ worse than $(4, 4, 0, 0, 0)$; both have $f_e = 0.5$, but the agreement is relatively higher for the first distribution

¹²⁷ <https://wasi.dev/> Retrieved: 6.12.2019

becoming an increasingly relevant technology no longer limited to the client side¹²⁸. For the WASM-based infrastructure of AWSM:RM, support for POSIX-like system functions in the browser¹²⁹ can be used to increase the capabilities of ReWaMP prototypes and further reduce the manual code adaption effort, since fewer dependencies *Dep* of Legacy System \mathcal{L} will require expert changes due to higher availability of libraries/system functions. To take advantage of the flourishing WASM environment, a review of the ReWaMP process and RWMPA services is required once a first stable version of WASI becomes available.

For the rapid creation of grid-based Web user interfaces from legacy non-Web desktop user interfaces, the index approximations of eq. (6.2) have been shown to produce reasonable quality with high performance in section 6.6.4. The combined objective functions were rated lower and represent significant computational complexity. To improve AWSM:RM, new objective functions and their combinations need to be investigated in order to provide a higher result quality. At the same time, the problem of computational complexity must be further addressed, leveraging parallel computation and intermediate results sharing and the improvement potential through low-level programming languages demonstrated in section 6.6.4.

The visual UI similarity process presented in fig. 7.1 invites improvements in three parts. The visual UI Element detection provides reasonable results for atomic UI Elements, but for productive use, improved detection of composite elements and improved overall detection rate is required. Our ongoing research thus focuses on enabling the use of deep neural networks, in particular using the Faster R-CNN architecture (Ren et al., 2017), through synthesis of sufficiently large datasets and

¹²⁸cf. <https://github.com/CraneStation/wasmtime/> Retrieved: 6.12.2019

¹²⁹currently available as polyfill: <https://wasi.dev/polyfill/> Retrieved: 6.12.2019

improving detection rate through application of the multi-agent system (MAS) paradigm to combine DOM-based with computer-vision-based detectors. Object detection for UI Elements, however, is still a research challenge due to the distinct characteristics compared to object detection in general, as described in section 7.4.2, with applications beyond Web Migration. The second potential improvement is the similarity approximation, which is currently focused on Layout, but could be extended towards consideration of Material and integrated with existing works on Task and Behavior similarity. The similarity measures will also benefit from new insights in perceived UI similarity, which is still not well-understood in research. The third relevant improvement opportunity is the provision of concrete guidance for the refinements based on the analysis results. An improved solution could combine the measurement with detailed reporting and resolution advice similar to the USF platform for usability smells in Web user interfaces (Grigera et al., 2017).

9.4.2 Open Questions

This section briefly outlines research questions that have not been addressed and should be explored in future work.

As observed in our systematic mapping study (Heil and Gaedke, 2017), consideration of Web Migration for SME-sized ISVs in academia is very low. While this thesis provided several contributions to address this gap, but there are still many research opportunities. Specifically, the integration of Web Migration activities with ongoing agile development, which was shown to be the requirement with almost zero consideration in section 3.3, poses a significant challenge. The AWSM Methodology and Toolsuite aimed for integration in the early phases of Web Migration and achieved a partial fulfillment of the requirement. Thus, a challenging open research question is how to improve integration of Web Migration

activities with ongoing agile development, not only for activities from migration initiation but also for the migration phase such as target architecture specification, Transformation or migration project monitoring.

Crowdsourcing has shown to be effective for the Reverse Engineering activity of Concept Assignment. An open research question is the application of Crowdsourcing in other Reverse Engineering areas. The reformulation of Concept Assignment as classification problem, its assessment in the eight foundational dimensions of Crowdsourcing (Latoza and Hoek, 2016), and mapping to the microtasking model described in section 5.5 can serve as a blueprint strategy for the identification of suitable Crowdsourcing models for other Reverse Engineering activities.

AWSM:RE with its OA-based ontological external knowledge representation is in line with recent efforts addressing standards-based data portability e.g. for research data (Díaz et al., 2019) and sharing platforms like hypothes.is¹³⁰. An interesting research challenge is to investigate how Reverse Engineering using the W3C Web Annotation standard and OWL can benefit from the foundational knowledge sharing paradigm of LOD (Linked Open Data), the mature semantic Web technology stack and the increasing research interest and active community in the context of the Solid¹³¹ project.

WebAssembly at its current stage¹³² was used as a technological enabler for Rapid Prototyping in the Web Migration domain based on the ideal of reuse and the available ISV staff expertise. With the recent upswing of WASM and its increasingly rich environment – including package

¹³⁰<https://hypothes.is> Retrieved: 6.12.2019

¹³¹<https://solid.mit.edu/> Retrieved: 6.12.2019

¹³²our research on web migration prototyping started in early 2016, one year before WebAssembly reached cross-browser consensus and in 2017, to the best of our knowledge, we were the first to consider WASM in the general Web Migration context

management¹³³, development environment¹³⁴, server-side runtimes¹³⁵ and integration with existing technologies like Microsoft's Razor Engine¹³⁶ – the maturity level and acceptance of WASM as Web technology brings up the research challenge of considering WASM beyond Prototyping use cases as primary Web Migration target and explore resulting opportunities for reuse based Web Migration.

The successful transfer of paradigms from other domains into the software migration domain presented in this thesis, e.g. Crowdsourcing from Software Engineering to Reverse Engineering and Rapid Prototyping from Agile Development and HCD to Web Migration, and in previous theses, e.g. Knowledge Management (Razavian, 2013) and Method Engineering (Khadka, 2016) to SOA Migration, show the potential of reviewing Web Migration from the perspective of paradigms successful in other domains. As observed in section 3.2, despite its success in Software Engineering, methods from Agile Development have hardly been considered for Web Migration with the exception of REMICS agile extensions (Krasteva et al., 2013) and the integration of Concept Assignment into ongoing agile development outlined in section 5.3. Thus, an important research challenge is to investigate advances in Web Migration through transfer of knowledge from other domains, in particular forward software engineering.

Understanding and analyzing the visual perception of user interfaces, their similarity, and complexity is a relatively new research area the scope of which exceeds Web Migration and which enables various use cases including UI refactoring, usability and interaction quality prediction without extensive interaction tracing, targeted UI optimization,

¹³³<https://wapm.io/> Retrieved: 6.12.2019

¹³⁴<https://webassembly.studio/> Retrieved: 6.12.2019

¹³⁵<https://wasmer.io/> Retrieved: 6.12.2019

¹³⁶<https://blazor.net/> Retrieved: 6.12.2019

design search and design transfer (Bakaev, Heil, Khvorostov, et al., 2019). The research challenge lies in the identification of appropriate objective, computable metrics that represent subjective visual perception of the user interface and to define methods and tools which make use of these insights to address the aforementioned use cases. Our ongoing research in this field in the context of collaboration¹³⁷ with colleagues of NSTU Novosibirsk focuses on visual complexity (Bakaev, Laricheva, et al., 2018), application of Kansei Engineering for subjective similarity features (Bakaev, Khvorostov, et al., 2017a), metrics integration (Bakaev, Heil, Perminov, et al., 2019) and computer-vision based UI mining (Bakaev, Heil, Khvorostov, et al., 2018). The Aalto Interface Metrics (AIM) (Oulasvirta et al., 2018) project¹³⁸ follows a similar objective and agenda. Understanding and analyzing visual perception of user interfaces is in its early stages and provides various research opportunities.

¹³⁷<http://www.wuikb.info/en/platform/about> Retrieved: 8.12.2019

¹³⁸<https://interfacemetrics.aalto.fi/> Retrieved: 6.12.2019

Scenario Materials

A

Table A.1: ZMS Scenario Application Quantitative Analysis Results

Criterion	Number
directories	31
source files	603
LOC	131,505
SLOC	106,469
comment lines	14,340
mean LOC per source file	218
mean SLOC per source file	177
header files	234
mean header LOC	69
implementation files	198
mean implementation LOC	398
classes	211
mean methods per class	16
functions	3362

Problem Analysis Materials

B

Table B.1: Stakeholder Map

Stakeholder	Interest/how affected	Capacity & Motivation	Possible Actions to Address Stakeholders
Software/ Migration Engineers	- new knowledge - increased workload through migration	- key role, defining available expertise and requirements for migration activities- limited motivation improvement of product and lower effort for updates and maintenance	- presentation of technology demonstrators etc. - reduce workload through tailored and integrated migration methods and tools

Stakeholder	Interest/how affected	Capacity & Motivation	Possible Actions to Address Stakeholders
Managers	<ul style="list-style-type: none"> - higher sales - reduced costs for software distribution and maintenance - increased risk and workload through migration 	<ul style="list-style-type: none"> - decision taker role - higher competitiveness due to improved product portfolio and reduced effort 	<ul style="list-style-type: none"> - risk management - communicate benefits through prototypes
Doctor's Offices	<ul style="list-style-type: none"> - innovative features & improved usability - changes in existing work patterns- training effort- new hardware - security and data privacy 	<ul style="list-style-type: none"> - customers - almost no intrinsic motivation 	<ul style="list-style-type: none"> - user training
Patients	<ul style="list-style-type: none"> - innovative features & improved usability- more interactivity 	<ul style="list-style-type: none"> - customers of customers 	<ul style="list-style-type: none"> - information about new features

Stakeholder	Interest/how affected	Capacity & Motivation	Possible Actions to Address Stakeholders
Competitors	<ul style="list-style-type: none"> - idea transfer to own products - competitive disadvantage 	<ul style="list-style-type: none"> - market share 	N/A
Companies with similar situation	<ul style="list-style-type: none"> - solution 	<ul style="list-style-type: none"> - improved situation 	<ul style="list-style-type: none"> - dissemination of Results

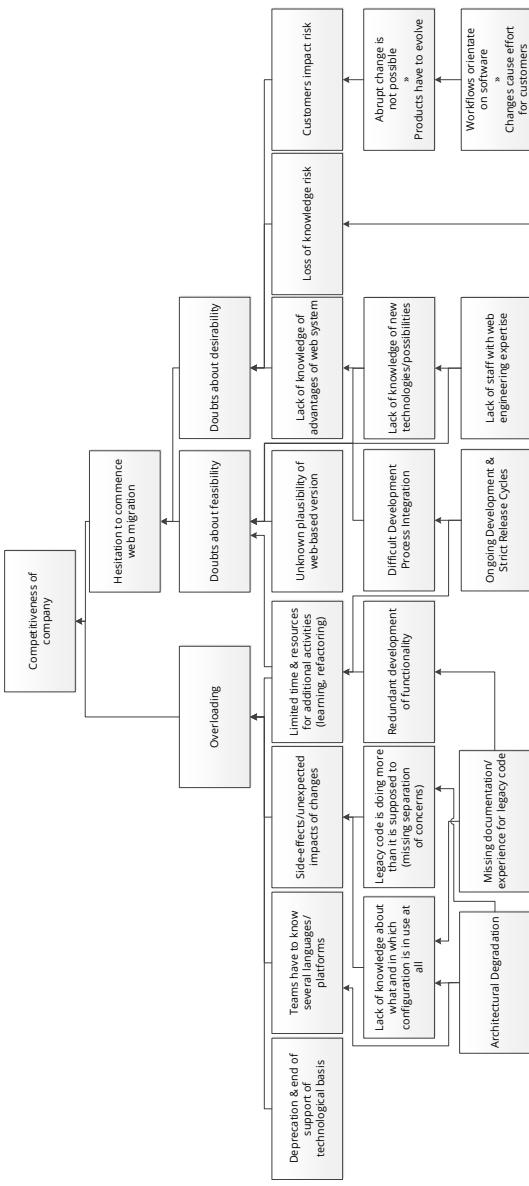


Figure B.1: Hierarchical Representation of Problem Domain as LFA Problem Tree

Solution Materials

C

Auswahl	EF	Ansatz	Autor	Typ	Gesamtignung KMU	
<input type="checkbox"/>	100%.	Rich-IDM: Transforming the user experience of legacy applications	Paiano, Pandurino, Mainetti	Ansatz	Gut	Details
<input type="checkbox"/>	75%.	Incubating services in legacy systems for architectural migration	Zhang, Yang	Ansatz	Schlecht	Details
<input type="checkbox"/>	55%	Migrating legacy spreadsheets-based systems to Web MVC architecture	Amalfitano, Fasolino, Tramontana, Di Mare	Ansatz	Durchschnittlich	Details
<input type="checkbox"/>	25,67%	Migrating legacy systems to the Web: an experience report	Aversano, Canfora, Cimilie, De Lucia	Ansatz	Durchschnittlich	Details
<input type="checkbox"/>	25%	Software modernization by recovering Web services from legacy databases	Pérez-Castillo, de Guzmán, Caballero	Ansatz	Schlecht	Details
<input type="checkbox"/>	20%	PRECISO: a reengineering process and a tool for database modernisation	del Castillo, García-Rodríguez, Caballero	Ansatz	Durchschnittlich	Details
<input type="checkbox"/>	16,67%	Migration of Relational Database to Document-Oriented Database	Kamitis, Arnicans	Ansatz	Durchschnittlich	Details
<input type="checkbox"/>	8,33%	Migration of a Legacy Procedural System to Service-Oriented	Milham	Ansatz	Durchschnittlich	Details

Figure C.1: S2DCS Results View, Showing Ranked List of Matching Approaches, EF Indicates Degree of Suitability to Migration Situation (German)

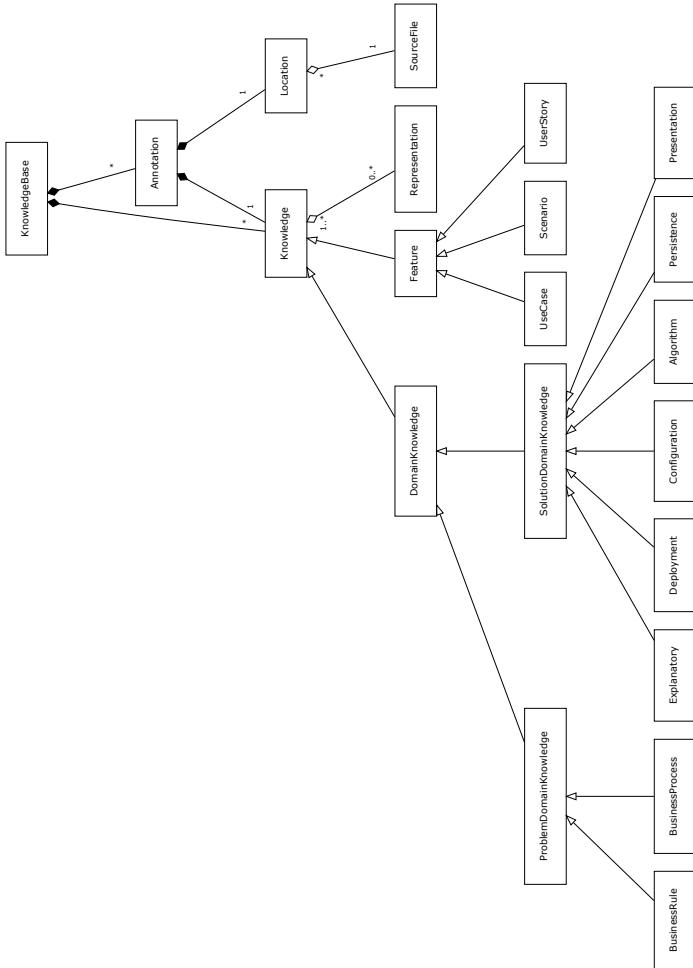


Figure C.2: SCKM Formalism

D.1 SCKM Ontology

The following is an extract from the SCKM Ontology¹³⁹ in Turtle notation. Since SWRL-Rules are very verbose in Turtle, they are shown separately in the following section.

Listing D.1: SCKM Ontology Extract

```
@prefix : <https://vsr.informatik.tu-chemnitz.de/
    ↳ ontologies/source_knowledge#> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix kdm: <http://schema.omg.org/spec/KDM/1.2/kdm> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    ↳ .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sckm: <https://vsr.informatik.tu-chemnitz.de/
    ↳ ontologies/source_knowledge#> .
```

¹³⁹available from the AWSM project page <https://vsr.informatik.tu-chemnitz.de/projects/2016/awsm/>

```

@base <https://vsr.informatik.tu-chemnitz.de/ontologies/
    ↵ source_knowledge> .

<https://vsr.informatik.tu-chemnitz.de/ontologies/
    ↵ source_knowledge> rdf:type owl:Ontology.

## Object Properties ##
oa:hasBody rdf:type owl:ObjectProperty ;
            rdfs:domain oa:Annotation ;
            rdfs:range sckm:DomainKnowledge ,
                        sckm:Feature .

oa:hasSelector rdf:type owl:ObjectProperty ;
            rdfs:domain oa:SpecificResource ;
            rdfs:range oa:TextPositionSelector ;
            rdfs:seeAlso <kdm:SourceRef> .

oa:hasTarget rdf:type owl:ObjectProperty ;
            rdfs:domain oa:Annotation ;
            rdfs:range oa:SpecificResource ;
            rdfs:seeAlso <kdm:SourceFile> .

sckm:influences rdf:type owl:ObjectProperty ;
            owl:inverseOf sckm:requires ;
            rdfs:domain sckm:DomainKnowledge ;
            rdfs:range sckm:Feature .

sckm:requires rdf:type owl:ObjectProperty ;
            rdfs:domain sckm:Feature ;
            rdfs:range sckm:DomainKnowledge .

```

```

sckm:within rdf:type owl:ObjectProperty ;
    rdfs:domain oa:Annotation ;
    rdfs:range sckm:Area .

## Data properties ##
oa:end rdf:type owl:DatatypeProperty ;
    rdfs:domain oa:TextPositionSelector ;
    rdfs:range xsd:integer .

oa:hasSource rdf:type owl:DatatypeProperty ;
    rdfs:domain oa:SpecificResource ;
    rdfs:range xsd:anyURI .

oa:start rdf:type owl:DatatypeProperty ;
    rdfs:domain oa:TextPositionSelector ;
    rdfs:range xsd:integer .

sckm:name rdf:type owl:DatatypeProperty ;
    rdfs:domain sckm:DomainKnowledge ,
        sckm:Feature ;
    rdfs:range xsd:string .

## Classes ##
oa:Annotation rdf:type owl:Class ;
    rdfs:seeAlso <kdm:Annotation> .

oa:SpecificResource rdf:type owl:Class ;
    owl:equivalentClass sckm:Area ;
    rdfs:seeAlso <kdm:SourceFile> ,
        <kdm:SourceRef> .

```

```
oa:TextPositionSelector rdf:type owl:Class ;
    rdfs:seeAlso <kdm:SourceRegion> .

sckm:Algorithm rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge ;
    rdfs:seeAlso <kdm:BehaviorUnit> .

sckm:Area rdf:type owl:Class .

sckm:BusinessProccess rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge ;
    rdfs:seeAlso <kdm:ConceptualFlow> .

sckm:Configuration rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge ;
    rdfs:seeAlso <kdm:ConfigFile> .

sckm:Deployment rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge .

sckm:DomainKnowledge rdf:type owl:Class .

sckm:Explanatory rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge ;
    rdfs:seeAlso <kdm:CommentUnit> .

sckm:Feature rdf:type owl:Class .

sckm:Persistence rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge .
```

```

sckm:Presentation rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge .

sckm:Rule rdf:type owl:Class ;
    rdfs:subClassOf sckm:DomainKnowledge ;
    rdfs:seeAlso <kdm:RuleUnit> .

```

D.2 SCKM Ontology SWRL Rules

The following is an extract from the SCKM Ontology comprising its SWRL-Rules in SWRL Human Readable Syntax¹⁴⁰.

Listing D.2: SCKM Ontology SWRL Rules

```

oa:Annotation(?AN) ^ sckm:Area(?AREA) ^ oa:hasTarget(?AN,
    ↳ ?SR) ^ oa:SpecificResource(?SR) ^ oa:hasSource(?SR,
    ↳ ?SOURCE) ^ oa:hasSource(?AREA, ?SOURCE) ^ oa:
    ↳ TextPositionSelector(?TAN) ^ oa:
    ↳ TextPositionSelector(?TAR) ^ oa:start(?TAN, ?san) ^
    ↳ oa:start(?TAR, ?sar) ^ oa:end(?TAN, ?ean) ^ oa:end
    ↳ (?TAR, ?ear) ^ swrlb:lessThanOrEqual(?sar, ?san) ^
    ↳ swrlb:greaterThanOrEqual(?ear, ?ean) -> sckm:within
    ↳ (?AN, ?AREA)

oa:start(?TR, ?sr) ^ oa:start(?TF, ?sf) ^ oa:hasSource(?SF,
    ↳ ?S) ^ oa:SpecificResource(?SR) ^ oa:
    ↳ TextPositionSelector(?TF) ^ oa:SpecificResource(?SF
    ↳ ) ^ swrlb:lessThanOrEqual(?er, ?ef) ^ oa:Annotation
    ↳ (?AR) ^ swrlb:greaterThanOrEqual(?sr, ?sf) ^ oa:
    ↳ hasBody(?AF, ?F) ^ oa:hasBody(?AR, ?R) ^ oa:
    ↳ TextPositionSelector(?TR) ^ oa:hasSelector(?SF, ?TF

```

¹⁴⁰cf. <https://www.w3.org/Submission/SWRL/>

```

    ↳ ) ^ oa:hasSelector(?SR, ?TR) ^ sckm:Feature(?F) ^
    ↳ oa:end(?TR, ?er) ^ oa:end(?TF, ?ef) ^ oa:hasTarget(
    ↳ ?AF, ?SF) ^ oa:hasTarget(?AR, ?SR) ^ oa:hasSource(
    ↳ ?SR, ?S) ^ sckm:DomainKnowledge(?R) ^ oa:Annotation
    ↳ (?AF) -> sckm:requires(?F, ?R) ^ sckm:influences(?R,
    ↳ ?F)

oa:start(?TR, ?sr) ^ oa:start(?TF, ?sf) ^ oa:
    ↳ SpecificResource(?SR) ^ oa:TextPositionSelector(?TF
    ↳ ) ^ oa:SpecificResource(?SF) ^ oa:Annotation(?AR) ^
    ↳ oa:hasBody(?AF, ?F) ^ oa:hasBody(?AR, ?R) ^ oa:
    ↳ TextPositionSelector(?TR) ^ oa:hasSelector(?SF, ?TF
    ↳ ) ^ oa:hasSelector(?SR, ?TR) ^ swrlb:lessThan(?sf,
    ↳ ?sr) ^ sckm:Feature(?F) ^ oa:end(?TR, ?er) ^ oa:end
    ↳ (?TF, ?ef) ^ oa:hasTarget(?AF, ?SF) ^ oa:hasTarget(
    ↳ ?AR, ?SR) ^ swrlb:greaterThanOrEqual(?ef, ?sr) ^
    ↳ swrlb:greaterThan(?er, ?ef) ^ sckm:DomainKnowledge(
    ↳ ?R) ^ oa:Annotation(?AF) -> sckm:requires(?F, ?R) ^
    ↳ sckm:influences(?R, ?F)

oa:start(?TR, ?sr) ^ oa:start(?TF, ?sf) ^ oa:
    ↳ SpecificResource(?SR) ^ oa:TextPositionSelector(?TF
    ↳ ) ^ oa:SpecificResource(?SF) ^ swrlb:
    ↳ greaterThanOrEqual(?ef, ?er) ^ oa:Annotation(?AR) ^
    ↳ oa:hasBody(?AF, ?F) ^ oa:hasBody(?AR, ?R) ^ oa:
    ↳ TextPositionSelector(?TR) ^ oa:hasSelector(?SF, ?TF
    ↳ ) ^ oa:hasSelector(?SR, ?TR) ^ sckm:Feature(?F) ^
    ↳ oa:end(?TR, ?er) ^ oa:end(?TF, ?ef) ^ oa:hasTarget(
    ↳ ?AF, ?SF) ^ oa:hasTarget(?AR, ?SR) ^ swrlb:

```

```

    ↳ lessThanOrEqual(?sf, ?sr) ^ sckm:DomainKnowledge(?R
    ↳ ) ^ oa:Annotation(?AF) -> sckm:requires(?F, ?R) ^
    ↳ sckm:influences(?R, ?F)

oa:start(?TR, ?sr) ^ oa:start(?TF, ?sf) ^ swrlb:
    ↳ greaterThanOrEqual(?er, ?ef) ^ oa:SpecificResource(
    ↳ ?SR) ^ oa:TextPositionSelector(?TF) ^ oa:
    ↳ SpecificResource(?SF) ^ oa:Annotation(?AR) ^ oa:
    ↳ hasBody(?AF, ?F) ^ oa:hasBody(?AR, ?R) ^ oa:
    ↳ TextPositionSelector(?TR) ^ oa:hasSelector(?SF, ?TF
    ↳ ) ^ oa:hasSelector(?SR, ?TR) ^ swrlb:
    ↳ lessThanOrEqual(?sr, ?sf) ^ sckm:Feature(?F) ^ oa:
    ↳ end(?TR, ?er) ^ oa:end(?TF, ?ef) ^ oa:hasTarget(?AF,
    ↳ ?SF) ^ oa:hasTarget(?AR, ?SR) ^ sckm:
    ↳ DomainKnowledge(?R) ^ oa:Annotation(?AF) -> sckm:
    ↳ requires(?F, ?R) ^ sckm:influences(?R, ?F)

oa:start(?TR, ?sr) ^ oa:start(?TF, ?sf) ^ oa:hasSource(?SF,
    ↳ ?S) ^ oa:SpecificResource(?SR) ^ oa:
    ↳ TextPositionSelector(?TF) ^ oa:SpecificResource(?SF
    ↳ ) ^ swrlb:greaterThanOrEqual(?ef, ?er) ^ oa:
    ↳ Annotation(?AR) ^ swrlb:lessThan(?sr, ?sf) ^ oa:
    ↳ hasBody(?AF, ?F) ^ oa:hasBody(?AR, ?R) ^ oa:
    ↳ TextPositionSelector(?TR) ^ oa:hasSelector(?SF, ?TF
    ↳ ) ^ oa:hasSelector(?SR, ?TR) ^ swrlb:
    ↳ greaterThanOrEqual(?er, ?sf) ^ sckm:Feature(?F) ^
    ↳ oa:end(?TR, ?er) ^ oa:end(?TF, ?ef) ^ oa:hasTarget(
    ↳ ?AF, ?SF) ^ oa:hasTarget(?AR, ?SR) ^ oa:hasSource(
    ↳ ?SR, ?S) ^ sckm:DomainKnowledge(?R) ^ oa:Annotation
    ↳ (?AF) -> sckm:requires(?F, ?R) ^ sckm:influences(?R,
    ↳ ?F)

```


AWSM:RM Materials

E

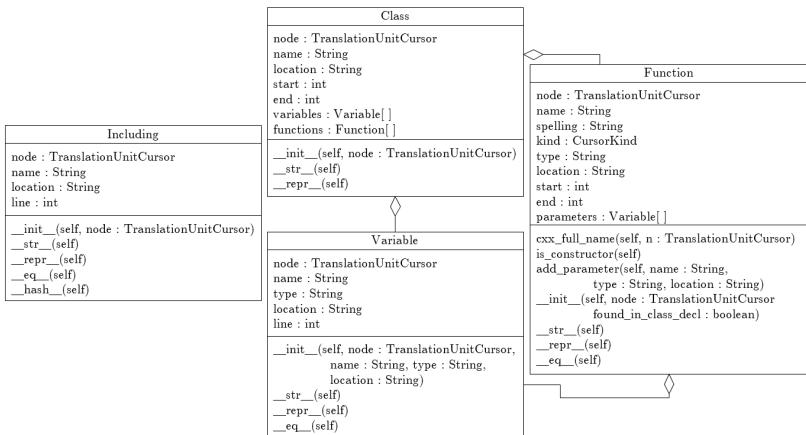


Figure E.1: ReWaMP C++ parser classes

Table E.1: ReWaMP Assumptions

Assumption	Description
Function Body	All functions start with one line as the header. Next, one line is possible starting with ':', but then the body starts. The first line of the body is only '{' and the last only '}'
1 Return	All functions have only one return value. This infers that no changed parameter is needed after function's execution.
UI IDs	The Web UI uses the old MFC ids. Not only the string versions, but also the integers in <i>data-rwmpId</i> .
File Size	All extracted files are relatively small and the number of extracted files is also manageable.
Main Space	To find the name space of the view, the main file's name can be used. Searching all class names if including this name will match only at one class.
Class Methods	All class functions are only declared in the class definition, but not defined.
1 Name/Class	All class methods have a unique name inside of one class. Overriding functions by parameter lists is not present.
No long long	No variables of the type long long or long double need to be exchanged with JS.
Collections	Any collection as vector or map does not contain pointers of non-fundamental types and need be exchanged with JS.
Pointers & Vectors	If a vector is needs to be transferred from or to JS, there is only one pointer in the type. Types as 'vector<int*>*' thus are not needed at exchange.

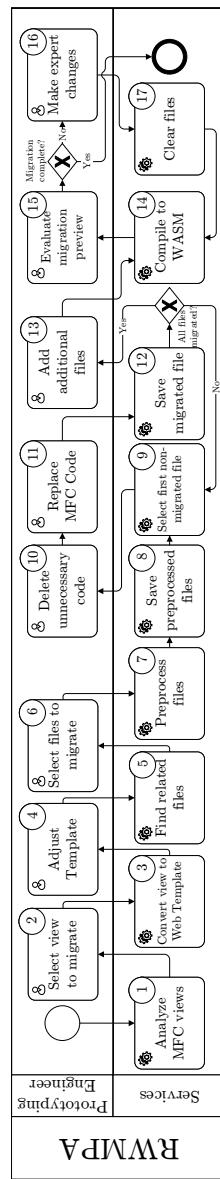


Figure E.2: RWMPA Process

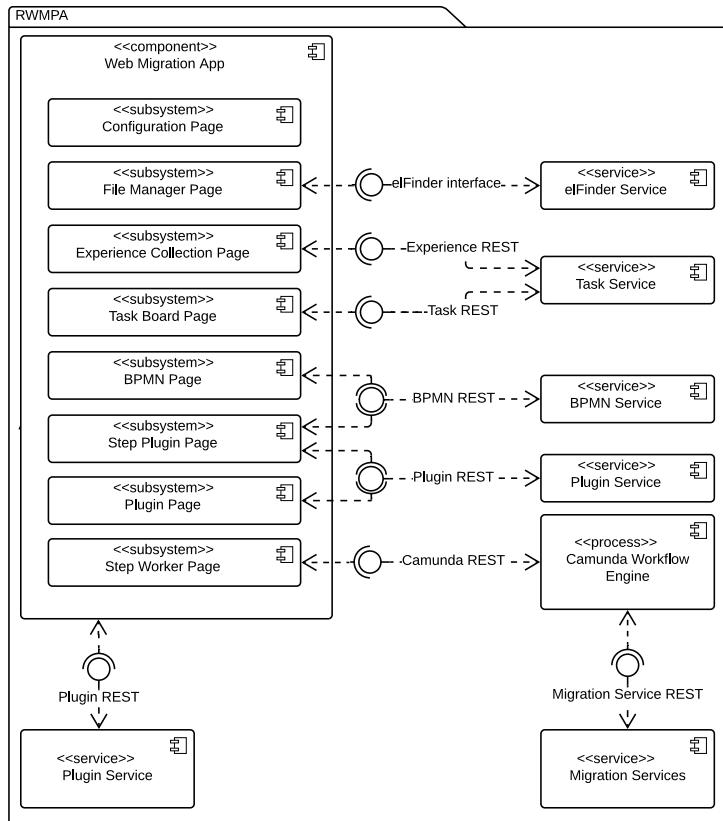


Figure E.3: RWMPA Architecture

ReWaMP Evaluation Materials

F

This appendix contains additional materials used in the ReWaMP evaluation experiments of this thesis.

Table F.1: ReWaMP Evaluation Guidelines

Guideline	Description
Questions	Answer all questions of the test subject, but stay as neutral and objective as possible. The answer should not influence the test subject.
Syntax Errors	Point out all apparent syntax errors a test subject made. The note can be given to the test subject as soon as it continues in another line.
Misdirection	Evince the test subject if it is apparently on the wrong way or performs useless actions. So for example if the test subject reads in details about the data persistence implementation in action 1 instead of concentrating upon the task.
Expertise	Help the test subject at missing programming experience or other programming issues. Thereby it does not play a role if it is related to C++, MFC or Web development. Only point out common structures, the support should not influence the test subject.

F.1 ReWaMP Questionnaire

The following questionnaire was used in the experimental evaluation of ReWaMP in section 6.6.2. Its structure and order of items represents the Google forms questionnaire as seen by the test subjects. The specific questions used in the results analysis in section 6.6.2 are prefixed with Q1, Q2, Q3 etc. All statements are likert items on a 5 level scale of agreement from 1 - not at all to 5 - exactly. Items shown with indented subitems are corresponding to multiple-choice items, the last question of WASM-T group is a single-choice item.

Self Assessment

1. How many years of experience in programming do you have?
2. I have very much experience in web development.
3. I have very much experience in programming C++.
4. I have very much experience in using MFC for creating a GUI.

Functionality of Legacy System and ReWaMP Prototype

1. In the legacy I am able to ...
 - browse the calendar
 - search all appointments of one patient
 - see all appointments at a specific date
 - find a new appointment by choosing a predefined variety
 - delete an existing appointment
2. In the resulting prototype I am able to ...
 - browse the calendar
 - search all appointments of one patient
 - see all appointments at a specific date
 - find a new appointment by choosing a predefined variety
 - delete an existing appointment

Quality of Legacy System and ReWaMP Prototype

1. In my opinion the legacy is performing just like I expected initially.
2. In my opinion the resulting prototype is performing just like I expected initially.
3. In my opinion the legacy is easy to use.
4. In my opinion the resulting prototype is easy to use.
5. In my opinion it is easy to learn how to use the legacy.
6. In my opinion it is easy to learn how to use the resulting prototype.

WASM-T

1. **Q1** It was very easy to understand, what I have to do in the ReWaMP process.
2. WASM-T was very easy to use.
3. **Q2** I had to do a lot of manual work at the ReWaMP process.
4. **Q3** WASM-T carried off much work for me in the ReWaMP process.
5. **Q4** Without WASM-T the ReWaMP process would generate much more work for me.
6. **Q5** The ReWaMP process took me only a short amount of time.
7. **Q6** Without WASM-T the ReWaMP process would take me much longer.
8. WASM-T supported me very good at finding code fragments, which need an expert change.
9. WASM-T supported me at writing code fragments by offering ReWaMP flags.
10. WASM-T supported me very good at exporting functions to the server.
11. The most difficult task was for me...

Extract view related behavior files

Configure WASM-T

Make expert changes

Write expert file

F.2 ReWaMP Evaluation Data

Table F.2 contains the raw evaluation data of the empirical evaluation of ReWaMP, table F.3 the raw evaluation data from time measurements and table F.4 from effort measurements.

Table F.2: ReWaMP Full Empirical Evaluation Data

Item	Sub 1	Sub 2	Sub 3	Sub 4	Sub 5	Sub 6
S1	0	10	0	6	9	9
S2	1	5	2	4	1	4
S3	2	3	1	3	3	2
S4	1	3	1	1	1	1
F11	Y	Y	Y	Y	Y	Y
F12	N	N	N	N	N	N
F13	Y	Y	Y	Y	Y	Y
F14	Y	Y	Y	Y	Y	Y
F15	N	N	N	N	N	N
F21	Y	Y	Y	Y	Y	Y
F22	N	N	N	N	N	N
F23	Y	Y	Y	Y	Y	Y
F24	Y	Y	Y	Y	Y	Y
F25	N	N	N	N	N	N
Q1	4	2	3	5	3	2
Q2	4	3	4	5	3	2
Q3	4	2	2	4	4	2
Q4	5	2	4	4	4	3
Q5	5	3	2	5	5	4
Q6	5	4	4	5	5	4
W1	2	2	1	4	3	2
W2	5	3	4	3	5	3

Item	Sub 1	Sub 2	Sub 3	Sub 4	Sub 5	Sub 6
W3	3	2	3	3	2	3
W4	5	4	4	4	5	4
W5	4	4	4	5	5	4
W6	3	4	2	2	3	4
W7	5	4	4	5	5	4
W8	5	3	4	5	4	1
W9	4	3	4	4	3	4
W10	5	4	4	5	5	5
W11	4	4	4	4	4	4

Questionnaire Responses, S1-S4 Self Assessment, F11-F25 Functionality, Q1-Q6 Quality, W1-W11 WASM-T, Abbreviations: Y - Yes, N - No

Table F.3: ReWaMP Full Time Evaluation Data

t1	t2	t31	t41	t51	t61	t71	t32	t42	t52	t62	t72
2319	230	17	2402	105	10	9	18	1924	531	21	11
673	106	18	1506	98	12	11	18	1536	456	21	12
1686	100	18	2425	159	12	18	18	2243	518	21	12
934	152	19	1788	70	13	14	18	1355	437	21	13
1560	166	18	2314	272	14	15	18	1314	415	21	11
1013	86	18	1949	132	14	13	19	1427	476	22	11

Measured times in seconds for tasks 1-7, tij indicates time for task i on view j

Table F.4: ReWaMP Full Effort Evaluation Data

e_{31}^u	e_{31}^d	e_{32}^c	e_{32}^u	e_{32}^d	e_{41}^c	e_{41}^u	e_{41}^d	e_{42}^c	e_{42}^u	e_{42}^d	e_{51}^c	e_{52}^c	e_{61}^c	e_{62}^c	e_{62}^d
81	60	12	32	43	3	11	45	1	33	94	9	60	273	164	4
81	60	12	32	43	4	10	43	2	30	96	6	57	263	154	4
81	60	12	32	43	2	10	37	0	36	93	9	60	273	164	4
81	60	12	32	43	1	14	37	0	36	89	6	57	263	154	4
81	60	12	32	43	2	14	35	0	30	88	6	57	263	154	4
81	60	12	32	43	2	6	35	0	32	90	6	60	263	164	4

Measured efforts in SLOC for tasks 3-6, e_{ij}^a indicates effort for action a on task i on view j, actions are c - create, u - update, d - delete, left out: $e_{51}^u, e_{51}^d, e_{52}^u, e_{52}^d, e_{61}^u, e_{61}^d$ (all values 0), e_{61}^u (all values 1), e_{61}^d (all values 9)

RWM^A Evaluation Materials

G

This appendix contains additional materials used in the RWM^A evaluation experiments of this thesis.

Table G.1: RWM^A Evaluation Guidelines

Situation	Interaction
Test subject does not understand the task at hand	The researcher points the test subject to the help section. If the test subject still cannot understand the task, the researcher takes a note and explains the task.
Test subject writes a line with an apparent syntax error	When the test subject continues to another line, the researcher points out the syntax errors to the test subject.
Test subjects wants to leave the RWM ^A workflow before successful completion	The researcher points out the problem to the test subject. This situation can only occur in the final task of the RWM ^A workflow: <i>evaluate migration using a preview</i> .

Situation	Interaction
Test subject gets stuck due to missing expertise.	The researcher helps the test subject by providing knowledge on common structures, while not influence the decisions of the test subject.
Test subject runs the correction cycle at least twice because of ignoring the compiler errors	The researcher points the test subject to the compiler errors and the possibility to reset the code to its original state.

G.1 RWMPA Questionnaire

The following questionnaire was used in the experimental evaluation of RWMPA in section 6.6.3. Its structure and order of items represents the Google forms questionnaire as seen by the test subjects. The specific questions used in the results analysis in section 6.6.3 are prefixed with Q1, Q2, Q3 etc. All statements are likert items on a 5 level scale of agreement from 1 - not at all to 5 - exactly. Items shown with indented subitems are corresponding to single-choice items. The original questionnaire was in German language, the following text is a translation into English.

Self Assessment

1. How many years of programming experience do you have?
2. I am very experienced in web development.
3. I am very experienced in development with C++.
4. I have already realized many projects using MFC.
5. My profession is ...

Workflow

1. **Q1** The workflow is easy to understand.
2. **Q2** I had to perform many manual edits.
3. **Q3** The workflow has reduced my work effort a lot
4. I required only little time for the workflow.
5. The workflow ran as I had expected.
6. **Q7** The tools required manual configuration.
7. **Q4** The workflow guidance supported me in understanding the workflow.
8. I was always aware of where I am in the workflow.
9. The most difficult task was for me...
 - Select view to migrate
 - Adjust the template
 - Select files to migrate
 - Delete unnecessary source code from file
 - Replace MFC Code
 - Add additional files
 - Evaluate migration using a preview
 - Make expert changes
 - I do not remember
10. The number of tasks is...
 - Too high
 - Adequate
 - Too low

Tasks

1. **Q5** The task guidance supported me in understanding the tasks.
2. **Q6** I was constantly aware of what is to be done.
3. The GUI is very easy to use.

4. I had to refer to the help pages often.
5. I had to go back one task often.
6. I had to reset a task often.

G.2 RWMPA Evaluation Data

Table G.2 contains the raw evaluation data of the empirical evaluation of RWMPA, table G.4 the raw data of the time measurements.

Table G.2: RWMPA Full Empirical Evaluation Data

S1	S2	S3	S4	S5	W1	W2	W3	W4	W5	W6	W7	W8
7	2	2	1	Student	5	2	4	5	4	1	5	4
10	2	3	1	Student	4	3	4	3	4	2	5	5
9	2	3	2	Researcher	3	3	4	4	4	1	4	3
15	3	2	1	SW Architect	4	3	3	4	4	3	2	5
15	5	2	1	SW Engineer	3	4	3	3	3	1	4	3
9	2	2	1	Student	5	2	5	4	5	1	5	5
7	3	1	1	Student	4	3	5	4	4	2	4	4

Questionnaire responses, S1-S5 Self Assessment, W1-W8 Workflow

Table G.3: RWMPA Full Empirical Evaluation Data (cont.)

W9	W10	T1	T2	T3	T4	T5	T6
16	Adq.	5	4	5	1	2	1
11	Adq	5	4	4	1	2	2
11	Adq	3	3	3	4	3	1
16	TH	3	3	5	1	2	1
16	Adq	4	4	5	2	2	2
11	Adq	5	4	5	2	1	1
16	Adq	4	4	5	3	2	2

Questionnaire responses, W9-W10 Workflow, T1-T6 Tasks, Abbreviations: SW - Software, ADQ - Adequate, TH - Too high

Table G.4: RWMPA Full Time Evaluation Data

t2.1	t4.1	t6.1	t10.1	t11.1	t13.1	t15.1	t16.1	t2.2	t4.2
255	131	90	273	1298	150	284	121	11	37
274	20	110	125	1525	41	155	97	3	2
275	58	36	187	1153	50	129	118	4	58
334	94	86	123	1093	66	258	105	N/A	N/A
230	270	183	333	1150	97	435	141	N/A	N/A
283	135	127	280	1703	241	284	249	13	22
227	123	114	179	1510	283	210	352	3	21

Times for manual tasks in seconds, $t_{i,j}$ - time for task i on view j, view 1: u_{cal} , view 2: u_{new} , N/A - not available**Table G.5:** RWMPA Full Time Evaluation Data (cont.)

t6.2	t10.2	t11.2	t13.2	t15.2	t16.2
17	34	775	22	190	704
56	56	503	25	212	745
63	29	378	16	110	803
N/A	N/A	N/A	N/A	N/A	N/A
N/A	N/A	N/A	N/A	N/A	N/A
13	12	697	10	280	959
82	8	651	17	207	1045

Times for manual tasks in seconds, $t_{i,j}$ - time for task i on view j, view 1: u_{cal} , view 2: u_{new} , N/A - not available

UI Transformer Evaluation Materials

H

This appendix contains additional materials used in the UI Transformer evaluation experiments of this thesis.

H.1 UI Transformer Questionnaire

The following questionnaire was used in the experimental evaluation of RWMPA in section 6.6.4. Its structure and order of items represents the Google forms questionnaire as seen by the test subjects, the numbering of items is equivalent to the numbers used in the results analysis in section 6.6.3. All statements are likert items on a 5 level scale of agreement from 1 - strongly disagree to 5 - strongly agree. Items shown with indented subitems are corresponding to single-choice items. The original questionnaire was in German language, the following text is a translation into English.

Task Description

Task: Compare the graphical user interface A with the graphical user interface B. It is not necessary to look for each small difference; only those which stand out are sufficient.

After the comparison, assess the following statements. There are no right or wrong answers.

Questions

- Q1** The elements in A and B are of equal size.
- Q2** The elements in A are at least as large as the elements in B.
- Q3** Left aligned elements in A are also left aligned in B.
- Q4** Right aligned elements in A are also right aligned in B.
- Q5** The elements in A and B were assigned the same labels.
- Q6** The elements in A and B occur in the same order.
- Q7** A and B are identical.

H.2 UI Transformer Evaluation Data

Table H.1 comprises the raw evaluation data of the performance evaluation and table H.3 of the empirical evaluation of UI Transformer.

Table H.1: UI Transformer Performance Evaluation Data

i	#c	d	t_1	t_2	t_3	t_4	t_5	t_6
1	23	0	52991	68519	104155	109407	141808	39902
2	3	0	16279	15478	28198	36613	45745	12307
3	20	1	54291	65815	100514	103873	124564	39784
4	10	1	34085	33375	51327	56723	83637	25141
5	24	1	70494	93683	136509	138137	225771	49583
6	16	0	43013	46811	72981	89444	132452	31102
7	17	0	45297	52942	74969	77915	108202	32471
8	11	0	34845	36229	57637	63544	89369	24338
9	9	1	30242	36700	60202	66611	108856	23185
10	44	0	90796	120752	172615	164410	280721	72676
11	21	0	54963	68549	90554	104358	171898	38218
12	8	0	27163	26802	41923	50756	73983	19923
13	13	0	37060	39273	61597	64808	116680	26815
14	34	1	89702	126699	174827	182356	293741	66420
15	18	0	47694	53825	77801	83897	133893	34807
16	18	0	47282	55174	76268	86177	127002	34545
17	17	0	45746	53665	81786	83927	134995	33211
18	26	0	66019	80763	117248	121745	179148	46401

i	#c	d	t_1	t_2	t_3	t_4	t_5	t_6
19	15	0	40564	46082	63594	70852	110924	30169
20	45	0	104418	148559	204541	211551	355712	74151
21	25	0	65858	76207	111911	112844	186624	46118
22	17	1	54653	66542	97983	105352	148268	41912
23	9	0	28687	28396	44376	48115	71211	20889
24	15	0	41959	46156	71397	71739	116134	30523
25	24	0	63211	72244	122268	111309	162568	43332
26	12	0	35509	39687	58831	62721	112413	25747
27	7	0	24856	23416	41805	45161	60975	18517
28	23	0	58854	67693	114172	107139	176296	42199
29	2	0	13826	11275	19827	26228	32520	11046
30	18	0	45411	53378	82518	83866	129468	34244
31	4	0	18772	17917	30647	37269	49093	14107
32	13	0	36239	39781	61251	70590	99885	27315
33	15	0	39338	44354	61204	68855	118620	30102
34	5	0	20911	18955	33715	40593	60701	15633
35	33	0	78749	98299	127206	150440	200350	57718
36	33	0	79973	110052	141222	150619	250577	57913
37	44	0	101350	134914	205079	182565	272043	73661
38	26	1	64573	90133	125346	127640	166848	47770

<i>i</i>	#c	d	<i>t</i> ₁	<i>t</i> ₂	<i>t</i> ₃	<i>t</i> ₄	<i>t</i> ₅	<i>t</i> ₆
39	130	0	343834	454761	532608	519752	1040988	312285
40	44	0	99871	139913	182475	191001	331185	72570
41	14	0	39149	42180	65693	74368	114644	28589
42	59	0	132403	196759	265300	274095	484474	101199
43	17	0	45233	52882	80782	87955	134987	33102
44	9	1	32129	35214	54621	62866	85028	23750
45	19	1	60210	84253	125042	123651	200841	44536
46	30	1	85789	122708	179650	178399	302888	62849
47	18	2	56963	75345	106238	114095	191564	43156
48	24	1	70606	101166	135809	145739	158754	54125
49	19	2	63455	86677	136162	140005	241658	47643
50	22	1	66127	79544	109946	111316	181482	49496

i - index of layout l_i , #c - number of controls, d - maximum nested depth, t_j - transformation time for combination
 j in ms

Table H.2: UITransformer Performance Evaluation Data (cont.)

i	t_{6c}	t_7	t_8	t_9	t_{10}
1	935	124686	112758	125425	203952
2	82	37245	27874	36292	57693
3	1217	123189	113321	115981	172955
4	506	68264	54769	65621	100792
5	2040	171170	128068	158620	208725
6	597	97152	74642	88178	142336
7	620	88533	74403	83224	136986
8	361	73976	59159	55261	111850
9	436	77622	49378	75228	119490
10	2647	206337	169370	184588	321439
11	850	114961	102480	101643	178151
12	231	57956	41112	55115	95696
13	419	84170	65159	73083	128390
14	3239	238330	178803	216341	302040
15	689	100739	77360	80813	165599
16	674	101801	78763	89394	142913
17	623	103131	74524	93218	156199
18	1136	146010	116134	122263	224872

<i>i</i>	<i>t_{6c}</i>	<i>t₇</i>	<i>t₈</i>	<i>t₉</i>	<i>t₁₀</i>
19	524	83245	64305	76411	128667
20	2645	269510	194894	227905	374662
21	1088	138377	96956	115449	205943
22	1691	114799	92023	90833	169773
23	273	55916	42943	50188	87356
24	532	89160	63933	73353	144680
25	1051	137168	103691	118977	214945
26	391	72834	57766	75042	128375
27	204	53822	39313	46733	79825
28	984	117695	94551	107629	229665
29	62	26766	20692	24792	42196
30	661	105192	80027	95523	182402
31	107	38928	30270	37001	63937
32	454	77543	59531	67263	119802
33	525	82338	65778	71539	138199
34	126	43591	35030	38422	65649
35	1662	163958	124326	164820	249250
36	1711	196452	144534	167886	287207
37	2756	232212	172857	251890	347843
38	1675	155965	133154	147034	238048

<i>i</i>	<i>t_{6c}</i>	<i>t₇</i>	<i>t₈</i>	<i>t₉</i>	<i>t₁₀</i>
39	15843	745364	635362	663514	1165719
40	2651	242200	186781	209142	355776
41	477	85711	62154	70282	141866
42	4297	370666	258822	307986	499852
43	624	97901	78447	89362	148623
44	450	72674	49610	66375	122605
45	1996	150111	111212	149800	179487
46	3394	220465	175241	202110	255030
47	1759	147362	108226	133777	221927
48	2549	180167	138400	170755	290642
49	2315	178030	142071	164210	247198
50	2261	117016	108540	127767	210481

i - index of layout l_i , #c - number of controls, d - maximum nested depth, t_j - transformation time for combination
j in ms

Table H.3: UI Transformer Empirical Evaluation Data Averaged across Test Subjects

Layout	Combination	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Mean
1	1	3.3	4.5	4.8	4.8	4.7	4.8	3.8	4.39
1	2	3.1	4.4	5.0	4.4	4.8	4.5	3.5	4.24
1	3	3.0	4.3	4.5	4.5	3.4	3.4	2.7	3.69
1	4	3.4	4.6	4.8	4.8	4.8	5.0	4.7	4.59
1	5	2.1	3.8	1.2	1.2	2.1	1.9	1.6	1.99
1	6	3.9	4.6	5.0	5.0	4.9	5.0	4.9	4.76
2	1	3.3	3.7	3.3	3.7	3.8	4.1	3.2	3.59
2	2	2.4	4.0	2.5	2.3	4.4	3.5	2.4	3.07
2	3	2.0	3.3	2.9	3.0	4.1	3.4	2.4	3.01
2	4	2.9	4.1	4.2	4.0	4.5	4.9	3.8	4.06
2	5	2.4	3.5	2.4	2.9	3.9	3.9	2.9	3.13
2	6	3.7	4.4	4.7	4.6	5.0	4.6	4.2	4.46
3	1	3.1	4.1	3.4	3.3	4.2	4.1	3.4	3.66
3	2	1.9	3.8	2.0	2.5	3.2	2.8	2.1	2.61
3	3	3.1	4.3	2.9	3.2	4.0	3.8	3.4	3.53
3	4	2.2	3.9	2.3	1.8	3.4	2.3	2.0	2.56
3	5	2.2	3.5	1.9	1.9	2.3	1.9	1.5	2.17
3	6	3.6	4.2	4.7	4.8	4.5	4.9	4.4	4.44
4	1	2.6	3.9	2.7	2.6	3.6	3.5	2.9	3.11
4	2	2.0	4.0	2.4	2.5	3.4	2.4	2.4	2.73
4	3	2.0	3.9	2.8	2.8	3.4	2.8	2.4	2.87
4	4	2.2	3.9	3.0	2.9	3.9	2.8	2.6	3.04
4	5	2.4	3.8	2.1	2.2	3.2	2.3	2.1	2.59
4	6	2.6	4.0	3.4	3.4	4.4	3.9	3.2	3.56
5	1	2.0	3.8	2.1	2.0	3.0	2.2	2.1	2.46
5	2	1.7	3.2	2.5	2.2	2.9	2.2	2.0	2.39
5	3	1.5	2.8	2.3	2.0	2.5	1.9	1.8	2.11

Layout	Combination	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Mean
5	4	1.8	3.7	1.4	1.2	3.1	2.0	1.6	2.11
5	5	2.3	3.5	2.0	1.9	3.3	2.3	1.6	2.41
5	6	3.1	4.0	4.0	4.3	4.3	3.9	3.8	3.91

AWSM:CI Evaluation Materials

I

This appendix contains additional materials used in the AWM:CI evaluation experiments of this thesis.

I.1 Visual Similarity Questionnaire

The following questionnaire was used in the experimental evaluation of AWM:CI in section 7.5.2. Its structure and order of items represents the printed questionnaire as given to the test subjects. Participant data was captured as text, favorite selection as single-choice with a text comment field, the questions per UI as Likert items on the scales indicated in them and the main difference questions as text.

Participant Data

1. Name
2. Age
3. Gender
4. Occupation

Question per each of the three groups of UIs

1. Which one did you like the most?

A0¹⁴¹

A1

A2

A3

2. Comments

Questions per each of the UIs in each group

1. **Difficult** Knowing how to achieve this task in the old user interface, how difficult was it to achieve the same in this new version of the user interface? [from 1 (not difficult) to 5 (very difficult)]
2. **Like** How much did you like this version of the old user interface? [from 1 (not at all) to 5 (very much)]
3. **⌚** How similar is this version to the old user interface? [from 1 (not similar) to 5 (very similar)]
4. What was the main difference between this UI and the old UI?

I.2 Visual Similarity Task Lists

Task List T_1 for u_1

- $t_{1,1}$ Add vacation for staff Mustermann from April 1 to 8
 $t_{1,2}$ Edit opening hours for April 25: office closed all-day

¹⁴¹this example shows the question for user interface u_1 encoded as A

Task List T_2 for u_2

- $t_{2,1}$ Switch view to month, week, 3-days
- $t_{2,2}$ For doctor Michael Meyer, edit appointment with Sophie Gersten on April 12 at 08:30: extend to 20 minutes
- $t_{2,3}$ For doctor Michael Meyer, move appointment from April 13 08:00 to April 11 08:00
- $t_{2,4}$ For doctor Max Mustermann, add new appointment for Patient Moritz¹⁴² on April 13 09:00
- $t_{2,5}$ For doctor Max Mustermann, cancel appointment for Anke Zweig on April 13 08:00

Task List T_3 for u_3

- $t_{3,1}$ In Stammdaten, add postal code 09111 and city Chemnitz
- $t_{3,2}$ In Stammdaten, edit birthday to 04.05.1978
- $t_{3,3}$ In Kostenträgerdaten, set DMP-Kennzeichen to 4 and besondere Personengruppe to Diabetes mellitus Typ 1
- $t_{3,4}$ In Kostenträgerdaten, assign¹⁴³ Stammarzt Arzt Vier

I.3 Visual Similarity Evaluation Data

Table I.1 contains the raw evaluation data of the evaluation experiment of AWSM:CI.

¹⁴²this task involved searching for the correct patient by name

¹⁴³involves enabling selection by checking checkbox Stammarzt

Table I.1: AWSM:CI Full Empirical Evaluation Data

	$A_0^1 A_0^2 A_0^3 A_1^1 A_1^2 A_1^3 A_2^1 A_2^2 A_2^3 A_3^1 A_3^2 A_3^3 A^* B_0^1 B_0^2 B_0^3 B_1^1 B_1^2 B_1^3 B_2^1 B_2^2 B_2^3 B_3^1 B_3^2 B_3^3 B^*$
3	2 2 3 2 2 4 4 4 1 5 4 0 3 1 2 1 5 4 2 4 2 3 4 0
2	3 4 2 4 2 2 4 4 2 3 4 2 1 4 5 1 2 4 1 3 4 1 2 4 0
1	2 5 1 3 5 1 2 5 1 2 3 1 1 2 5 3 1 2 2 4 1 3 3 3 3
1	4 5 1 3 4 1 4 5 1 5 5 3 1 5 5 1 3 3 1 5 3 2 5 4 2
1	4 4 1 2 3 1 4 4 1 1 3 3 1 5 5 1 1 3 1 4 4 1 3 4 1
2	5 4 4 2 2 3 4 4 2 1 5 3 3 3 5 4 1 3 3 4 4 3 4 4 2
2	4 4 2 2 2 3 4 3 1 2 2 3 3 2 4 1 2 2 4 2 3 2 2 2

Questionnaire responses, A_i^j - rating for variant i of user interface A for group question j , A^* - favorite of group, user interface $u'_1 - A_0$, user interface $u'_2 - B_0$, variations i in order: 1 - ori, 2 - ord, 3 - den

Table I.2: AWSM:CI Full Empirical Evaluation Data (cont.)

	$C_0^1 C_0^2 C_0^3 C_1^1 C_1^2 C_1^3 C_2^1 C_2^2 C_2^3 C_3^1 C_3^2 C_3^3 C^*$
4	2 2 3 1 2 1 5 5 1 4 4 0
2	2 3 2 3 3 2 2 4 2 3 4 1
3	1 5 4 1 2 4 1 4 1 2 4 3
2	4 3 3 2 3 1 4 4 1 5 4 2
1	4 4 1 4 4 2 3 3 1 4 4 2
1	5 5 2 2 3 1 4 5 1 3 5 2
2	4 4 5 1 2 4 2 3 2 4 4 2

Questionnaire responses, C_i^j - rating for variant i of user interface C for group question j , C^* - favorite of group, user interface $u'_3 - C_0$, variations i in order: 1 - ori, 2 - ord, 3 - den

Glossary

- Artifact** “A software artifact is a tangible machine-readable document created during software development. Examples are requirement specification documents, design documents, source code, and executables.” (Object Management Group, 2016a), (cf. *physical asset* ISO/IEEE, 2017b, entry 3.261). 20, 32, 34, 36, 37, 39, 44, 52–54, 68, 94, 101, 113–115, 124, 129, 134, 136, 140, 148, 151–159, 162, 163, 170, 200, 202, 224, 276, 302, 303, 305, 310–313, 327
- Asset** An “item, thing or entity that has potential or actual value to an organization” (ISO/IEEE, 2017b). “A software asset is a description of a partial solution … or knowledge … that engineers use to build or modify software products.” (Object Management Group, 2016a), (cf. *intangible assets* ISO/IEEE, 2017b, entry 3.261). 36, 134
- Business case** A “documented economic feasibility study used to establish validity of the benefits of a selected component lacking sufficient definition and that is used as a basis for the authorization of further project management activities” (ISO/IEEE, 2017b, entry 3.442). 32, 34, 35, 53, 60, 65, 73, 77, 93, 96, 115, 191, 300, 302, 385
- Concept** “Concepts are units of human knowledge that can be processed by the human mind (short-term memory) in one instance.” (V. Rajlich and Wilde, 2002), defined in Definition 7. 143, 148, 153, 154, 165, 172, 176, 179, 181, 380

Concept Assignment A reverse engineering technique that aims at discovering human-oriented Concepts and assigning them to their realizations in the source code (Biggerstaff et al., 1994), cf. section 5.1.3. 14, 15, 113, 114, 142–152, 164–169, 171, 172, 174, 175, 187, 188, 303, 304, 320, 321, 324, 326, 330, 331

Crowdsourcing “The act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call.” (Howe, 2006), defined in Definition 8. 15, 113, 114, 142–145, 165–168, 175, 176, 181, 182, 186, 187, 323, 324, 330, 331

Desktop Application Application software which has a GUI, or, less commonly, a TUI, that is not based on Web technologies in contrast to Web Applications. A comparison can be found in section 1.1. 1, 3–6, 19, 23, 37, 57, 59, 74, 75, 83, 89, 97, 102, 117, 137, 191, 192, 195, 248, 253, 258, 260, 262, 328

Encapsulation Encapsulation approaches are Web Migration approaches, that wrap the unchanged Legacy System or parts of it and expose a new interface which is then integrated with the target system. Defined in section 3.2. 50, 52, 62, 74, 75, 84, 86, 89, 90, 94, 95, 98–100, 103, 195, 203, 309, 310, 318

Forward Engineering The application of Software Engineering to create a new software based on software requirements or on the results of Reverse Engineering, when part of Reengineering.. 42, 43, 55, 59, 62, 67, 76, 95, 115, 141, 149, 150, 175, 190, 195, 200, 224, 275, 276, 303–305, 382

Hybrid Web Application A Web Application that consists of parts of reused legacy code, a transformed Web UI and a generated RESTful API, cf. section 6.3.. 198

Legacy Modernization see Software Modernization. 6, 8, 35, *see also* Software Modernization

Legacy System “Any systems that cannot be modified to adapt to continually changing business requirements and their failure can have a severe impact on business.” (Brodie and Stonebraker, 1995), defined in Definition 3. 1, 6–10, 12–15, 19–21, 23, 24, 28, 29, 31, 32, 35–42, 44–47, 50–57, 59, 60, 62, 63, 66–68, 70–90, 94, 95, 97–99, 101–103, 105–107, 110, 112–115, 117, 118, 122–126, 129–141, 143, 145, 146, 148–150, 152, 154, 159, 161, 162, 169, 174, 187, 189–193, 195–200, 202–204, 207–210, 213, 216, 217, 221, 223, 228, 229, 232–234, 236, 240, 242, 247–249, 253–255, 258–261, 263–266, 269, 270, 273, 277, 279, 284–289, 293–297, 300–303, 307, 312, 314–317, 319–321, 324–326, 328, 380, 382–385

Metamodel A “logical information model that specifies the modeling elements used within another (or the same) modeling notation” (ISO/IEEE, 2017b, entry 3.2433). 44, 46, 62, 80, 87, 100, 130, 201

Migration Engineer A Software Engineer who is performing Web Migration activities. Performing Web Migration activities does not necessarily imply Web Migration expertise. Migration Engineers are to be considered a sub-class of Software Engineers, i.e. all characteristics of Software Engineers apply.. 28, 118, 121, 123, 124, 147, 150, 152, 161, 168, 198, 241, 275

Prototyping . 73, 128, 191, 193, 194, 200, 207, 305, 331, *see also* Software Prototyping

Rapid Prototyping A “type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development

process” (ISO/IEEE, 2017b), defined in Definition 10. 14, 15, 115, 124, 128, 138, 190–197, 200, 235, 241, 300, 304, 320, 323, 325, 330, 331, 382

Rapid Web Migration Prototyping A type of Rapid Prototyping which allows the rapid creation of demonstrative, horizontal prototypes, that represent future Web-based versions of existing non-Web Legacy System allowing to assess and demonstrate their plausibility and desirability for decision making in Web Migration. Defined in Definition 11. 115, 118, 119, 123, 124, 189, 195, 196, 199, 201, 203, 206, 247, 255, 300, 302, 312, 321, 325, 327

Reengineering The “examination and alteration of software to reconstitute it in a new form, including the subsequent implementation of the new form” (ISO/IEEE, 2017b, entry 3.3346), it consists of Reverse Engineering and Forward Engineering. 12, 32, 43, 50, 52, 59, 62, 72, 73, 76, 82, 83, 86, 87, 95, 99, 101, 109, 113, 119, 128, 132, 142, 145, 150, 151, 160–162, 194, 223, 309, 310, 380

Reverse Engineering A “software engineering approach that derives a system’s design or requirements from its code” (ISO/IEEE, 2017b, entry 3.3501). 36, 38, 41, 43, 62, 69, 70, 80–82, 85, 87, 88, 94, 96, 99, 113, 114, 118, 121, 123, 128, 139, 141–145, 148, 150, 151, 165, 167, 174, 175, 187, 188, 224, 277, 301, 304, 312, 324, 326, 327, 330, 331, 380, 382

Risk Management An “organized process for identifying and handling risk factors” (ISO/IEEE, 2017b, entry 3.3528). 27, 34, 35, 55, 60, 67, 72, 77, 82, 93, 96–99, 104, 115, 137, 190, 192, 300, 301, 307, 308, 336

Software Migration modification of software to run in different environments, a software maintenance technique and a part of a software’s lifecycle (IEEE Computer Society, 2014, p. 5-10). 1, 33, 41, 42, 95, 383, 384

Software Modernization The “process of evolving existing software systems by replacing, re-developing, reusing, or migrating the software components and platforms, when traditional maintenance practices can no longer achieve the desired system properties” (Khadka, 2016, p. 6). 6, 8–10, 12, 13, 15, 18, 32, 34–37, 42–44, 57, 67–69, 77, 78, 80, 95, 98, 140, 191, 309, 381, 385

Software Prototyping “Software prototyping is an activity that generally creates incomplete or minimally functional versions of a software application, usually for trying out specific new features, soliciting feedback on software requirements or user interfaces, further exploring software requirements, software design, or implementation options, and/or gaining some other useful insight into the software.” (IEEE Computer Society, 2014), defined in Definition 9. 1

Source System The existing software system from which the target system is created during Software Migration. Typically, source systems are Legacy Systems.. 2, 17, 77, 80, 81, 121, 260, 262, 307, 309, 310, 383

Target Environment The specific environment for which a software is modified in Software Migration. 2, 33, 105, 383

Target System The software system to be created through Software Migration in the intended target environment, based on an existing source system. For Web Migration, target systems are Web Systems.. 32, 50, 55, 60, 61, 63, 71, 74–77, 79, 82–84, 86, 87, 89, 90, 97, 100, 101, 121, 126, 260, 275, 300, 383

Technical Debt “Technical debt is a collection of design or implementation constructs that are expedient in the short term but set up a technical context that can make future changes more costly or impossible.” (Avgeriou et al., 2016). 7, 13, 28, 101

Transformation Transformation approaches are Web Migration approaches that process the legacy source code by a series of automatic steps turning it into the target system's source code. Defined in section 3.2.. 12, 15, 32, 46–48, 50, 52, 54–56, 59–62, 67–72, 76–81, 84, 86, 95, 97–101, 103, 109, 113, 114, 118, 119, 123, 128, 129, 132, 142, 145, 150, 151, 160–162, 189, 197–199, 203, 208, 209, 211, 213, 216–219, 221, 223–226, 228, 232, 233, 247, 248, 250, 251, 254, 269, 300, 302, 307, 309, 321, 330

Web Application “A Web Application is a software system based on technologies and standards of the World Wide Web Consortium (W3C) that provides Web specific resources such as content and services through a user interface, the Web browser.” (Kappel et al., 2006), defined in Definition 2. 2–6, 15, 31, 33, 40, 49, 63, 74, 78–82, 85–87, 97, 98, 104, 116, 176, 198, 203, 210, 214, 233, 254, 299–301, 307, 310, 380

Web Engineering “Web Engineering is the application of systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based applications.” (Deshpande et al., 2002). 2, 15, 18, 29, 37, 41, 42, 106, 107, 114, 115, 117, 123, 124, 127, 128, 139, 145, 188, 189, 191, 192, 195, 196, 199, 232, 236, 241, 242, 244, 254, 257, 260, 263, 264, 296, 303, 321–323, 325

Web Migration Web Migration is a type of Software Migration which transfers a *non-Web source system* to a *Web-based target environment*. It is *adaptive and perfective maintenance* adapting software systems for web-based environments and improving functionality and maintainability. (ISO/IEEE, 2006). 1–4, 6, 8–18, 23, 25, 27–29, 31, 33–35, 37–43, 45–47, 49, 51, 52, 63, 77, 79, 81, 91–95, 97, 98, 100–111, 113–125, 127, 128, 131, 135–138, 140, 145, 147, 149, 155, 160, 161, 163, 175, 188–192, 194–196, 198–200,

214, 232, 233, 240, 255, 257–260, 262–267, 274, 275, 286, 289, 291, 295–297, 299–302, 304–311, 317–327, 329–331, 380–384, 425, 429

Web Migration Prototype A limited Web-based version of the Legacy System which serves as *functional, demonstrative product* (ISO/IEEE, 2017b), showing the plausibility and desirability. Web migration prototypes are *demonstrative prototypes* which communicate the ideas of the user interface and capabilities of the envisioned Web System, allowing decision makers to assess desirability and benefits (Wallmüller, 2001), thus providing a tangible contribution to making the migration business case. Web migration prototypes are *horizontal prototypes* (Wallmüller, 2001) focusing on UI and application logic on the client side. Cf. section 6.3.. 114, 189, 191–193, 198–200, 232, 233, 242, 254, 300, 302, 305, 307, 308, 314

Web System A Web System is a software system based on technologies and standards of the W3C that provides Web specific resources. (adapted from Gaedke, 2000; Kappel et al., 2006), defined in Definition 1. 1–4, 15, 33, 53, 55, 60, 62–64, 74, 75, 77, 80–82, 84, 86, 89, 90, 93, 95, 97, 190, 191, 300, 302, 307, 383, 385

Web Systems Evolution An area of Software Modernization research, where both the target system and also the source system is a Web System. (cf. Kienle and Distante, 2014). 3, 49, 77, 78, 80, 81, 96, 97, 307, 309

Bibliography

Printed References

- Abhervé, Antonin, Andrey Sadovykh, Satish Srirama, et al. (2013). *REMICS Migrate Principles and Methods*. Tech. rep. 257793, pp. 1–50.
- Abrahamsson, Pekka, Outi Salo, and Jussi Ronkainen (2002). *Agile Software Development Methods - Review and Analysis*. VTT. arXiv: 1709.08439.
- Ahmad, Aakash and Muhammad Ali Babar (2014). “A Framework for Architecture-Driven Migration of Legacy Systems to Cloud-Enabled Software”. In: *Proceedings of the First International Conference on Dependable and Secure Cloud Computing Architecture - DASCCA '14*, pp. 1–8.
- Alavi, Maryam (June 1984). “An Assessment of the Prototyping Approach to Information Systems Development”. In: *Communications of the ACM* 27.6, pp. 556–563.
- Albrecht, A.J. and J.E. Gaffney (Nov. 1983). “Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation”. In: *IEEE Transactions on Software Engineering* SE-9.6, pp. 639–648.
- Allahbakhsh, Mohammad, Boualem Benatallah, A. Ignjatovic, et al. (2013). “Quality Control in Crowdsourcing Systems: Issues and Directions”. In: *IEEE Internet Computing* 17.2, pp. 76–81.
- Almonaies, Asil A., James R. Cordy, and Thomas R. Dean (2010). “Legacy System Evolution Towards Service-Oriented Architecture”. In: *International Workshop on SOA Migration and Evolution (SOAME 2010)*, pp. 53–62.
- Amazon Web Services Inc. (2017). *An Overview of the AWS Cloud Adoption Framework - Version 2*. Tech. rep. February. Amazon Web Services, Inc.
- (2018). *AWS Migration Whitepaper*. Tech. rep. March. Amazon Web Services, Inc.

- Anderson, David J (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press.
- Arsanjani, A., S. Ghosh, A. Allam, et al. (2008). “SOMA: A Method for Developing Service-Oriented Solutions”. In: *IBM Systems Journal* 47.3, pp. 377–396.
- Aversano, Lerina, Gerardo Canfora, Aniello Cimitile, and Andrea De Lucia (2001). “Migrating Legacy Systems to the Web: An Experience Report”. In: *Proceedings Fifth European Conference on Software Maintenance and Reengineering*. IEEE Comput. Soc, pp. 148–157.
- Avgeriou, Paris, Philippe Kruchten, Ipek Ozkaya, and Carolyn B Seaman (2016). “Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162).” In: *Dagstuhl Reports* 6.4, pp. 110–138.
- Bakaev, Maxim, Sebastian Heil, Vladimir Khvorostov, and Martin Gaedke (2018). “HCI Vision for Automated Analysis and Mining of Web User Interfaces”. In: *Web Engineering: Proceedings of the 18th International Conference on Web Engineering (ICWE2018)*. Ed. by Tommi Mikkonen, Ralf Klamma, and Juan Hernández, pp. 136–144.
- (2019). “Auto-Extraction and Integration of Metrics for Web User Interfaces”. In: *Journal of Web Engineering* 17.6, pp. 561–590.
- Bakaev, Maxim, Sebastian Heil, Nikita Perminov, and Martin Gaedke (2019). “Integration Platform for Metric-Based Analysis of Web User Interfaces”. In: *Web Engineering: Proceedings of 19th International Conference on Web Engineering (ICWE2019)*. Ed. by Maxim Bakaev, Flavius Frasincar, and In-Young Ko. Daejeon, Korea: Springer, pp. 525–529.
- Bakaev, Maxim, Vladimir Khvorostov, Sebastian Heil, and Martin Gaedke (Sept. 2017a). “Evaluation of User-Subjective Web Interface Similarity With Kansei Engineering-Based ANN”. In: *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, pp. 125–131.
- (2017b). “Web Intelligence Linked Open Data for Website Design Reuse”. In: *Web Engineering: Proceedings of the 17th International Conference on Web Engineering (ICWE2017)*. Ed. by Jordi Cabot, Roberto De Virgilio, and Riccardo Torlone, pp. 370–377.

- Bakaev, Maxim, Tatiana A. Laricheva, Sebastian Heil, and Martin Gaedke (Oct. 2018). "Analysis and Prediction of University Websites Perceptions by Different User Groups". In: *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*. IEEE, pp. 381–385.
- Barbier, Franck, Gaëtan Deltombe, Kamil Rybiński, Sławomir Blatkiewicz, and Michał Śmiałek (2013). *REMICS Recover Toolkit , Final Release*. Tech. rep. 257793, pp. 1–18.
- Barbier, Franck, Alexis Henry, Kamil Rybiński, Sławomir Blatkiewicz, and Michał Smialek (2013). *REMICS Recover Principles and Methods*. Tech. rep., pp. 1–27.
- Batlajery, Belfrit V., Ravi Khadka, Amir M. Saeidi, Slinger Jansen, and Jurriaan Hage (2014). *Industrial Perception of Legacy Software System and Their Modernization*. Tech. rep. September.
- Beedle, Mike and Ken Schwaber (2001). *Agile Software Development With Scrum*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR.
- Benguria, Gorka, Brian Elvesæter, and Sylvia Ilieva (2013). *REMICS Handbook , Final Release*. Tech. rep.
- Bernhart, Mario, Andreas Mauczka, Michael Fiedler, Stefan Strobl, and Thomas Grechenig (Sept. 2012). "Incremental Reengineering and Migration of a 40 Year Old Airport Operations System". In: *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, pp. 503–510.
- Biggerstaff, Ted J., Bharat G. Mitbander, and Dallas E. Webster (1993). "The Concept Assignment Problem in Program Understanding". In: *Proceedings of 1993 15th International Conference on Software Engineering*. Vol. 37. 5. IEEE Comput. Soc. Press, pp. 482–498.
- (May 1994). "Program Understanding and the Concept Assignment Problem". In: *Communications of the ACM* 37.5, pp. 72–82.
- Bing Wu, Deirdre Lawless, Jesus Bisbal, et al. (1997). "The Butterfly Methodology: A Gateway-Free Approach for Migrating Legacy Information Systems". In: *Proceedings. Third IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.97TB100168)*. IEEE Comput. Soc, pp. 200–205.

- Binkley, David (2007). "Source Code Analysis: A Road Map". In: *Future of Software Engineering (FOSE '07)*. IEEE Computer Society, pp. 104–119.
- Bisbal, Jesús, Deirdre Lawless, Bing Wu, and Jane Grimson (1999). "Legacy Information Systems: Issues and Directions". In: *IEEE Software* 16, pp. 103–111.
- Bitkom (2013). *Arbeit 3.0: Arbeiten in Der Digitalen Welt*. Tech. rep. Berlin: BITKOM, pp. 1–36.
- "Algorithms and Theory of Computation Handbook - Levenshtein Distance" (2019). In: *Dictionary of Algorithms and Data Structures*. Ed. by Paul E. Black. CRC Press LLC.
- Bodhuin, Thierry, Enrico Guardabascio, and Maria Tortorella (2002). "Migrating COBOL Systems to the Web by Using the MVC Design Pattern". In: *Ninth Working Conference on Reverse Engineering, 2002. Proceedings*. IEEE Comput. Soc, pp. 329–338.
- (2003). "Migration of Non-Decomposable Software Systems to the Web Using Screen Proxies". In: *10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings*. IEEE, pp. 165–174.
- Bodhuin, Thierry and Maria Tortorella (2004). "Using Grid Technologies for Web-Enabling Legacy Systems". In: *Eleventh Annual International Workshop on Software Technology and Engineering Practice*. IEEE, pp. 186–195.
- Borchers, Jens (1996). "Reengineering-Factory — Erfolgsmechanismen Großer Reengineering-Maßnahmen". In: *Softwarewartung und Reengineering: Erfahrungen und Entwicklungen*. Ed. by Franz Lehner. Wiesbaden: Deutscher Universitätsverlag, pp. 19–29.
- Bozzon, Alessandro, Sara Comai, Piero Fraternali, and Giovanni Toffetti Carughi (2006). "Conceptual Modeling and Code Generation for Rich Internet Applications". In: *Web Engineering: Proceedings of the 6th International Conference on Web Engineering (ICWE2006)*, p. 353.
- Brabham, Daren C. (Feb. 2008). "Crowdsourcing as a Model for Problem Solving". In: *Convergence: The International Journal of Research into New Media Technologies* 14.1, pp. 75–90.

- Bressler, Stacey E. and Charles Grantham (2000). *Communities of Commerce: Building Internet Business Communities to Accelerate Growth, Minimize Risk, and Increase Customer Loyalty*. 1st editio. McGraw-Hill.
- Brodie, M. L. and M. Stonebraker (1995). *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach*. 1st ed. Morgan Kaufmann Publishers Inc.
- Brunelière, Hugo, Jordi Cabot, Grégoire Dupé, and Frédéric Madiot (Aug. 2014). “MoDisco: A Model Driven Reverse Engineering Framework”. In: *Information and Software Technology* 56.8, pp. 1012–1032.
- Brunelière, Hugo, Jordi Cabot, Javier Luis Canovas Izquierdo, et al. (Aug. 2015). “Software Modernization Revisited: Challenges and Prospects”. In: *Computer* 48.8, pp. 76–80.
- Brunelière, Hugo, Javier Cánovas, Guillaume Doux, Oliver Strauß, and Leire Orue-Echevarria (2013). *ARTIST Taxonomy of Legacy Artefacts*. Tech. rep.
- Cagnin, M.I., J.C. Maldonado, and R.D. Penteado (2003). “PARFAIT: Towards a Framework-Based Agile Reengineering Process”. In: *Proceedings of the Agile Development Conference, 2003. ADC 2003*. January. IEEE, pp. 22–31.
- Cai, Deng, Shipeng Yu, Ji-rong Wen, and Wei-ying Ma (2003). *VIPS : A Vision Based Page Segmentation Algorithm*. Tech. rep. Microsoft Research, pp. 1–32.
- Cajas, Viviana, Matías Urbíeta, Yves Rybarczyk, Gustavo Rossi, and César Guevara (2019). “An Approach for Migrating Legacy Applications to Mobile Interfaces”. In: *New Knowledge in Information Systems and Technologies*. Vol. 3, pp. 916–927.
- Canfora, Gerardo, Aniello Cimitile, Andrea De Lucia, and Giuseppe Antonio Di Lucca (Oct. 2000). “Decomposing Legacy Programs: A First Step Towards Migrating to Client–server Platforms”. In: *Journal of Systems and Software* 54.2, pp. 99–110.
- Canfora, Gerardo and Massimiliano Di Penta (May 2007). “New Frontiers of Reverse Engineering”. In: *Future of Software Engineering (FOSE '07)*. Vol. 11. 6. IEEE, pp. 326–341.

- Carughi, Giovanni Toffetti, Sara Comai, Alessandro Bozzon, and Piero Fraternali (2009). “Modeling Distributed Events in Data-Intensive Rich Internet Applications”. In: *Web Information Systems Engineering – WISE 2007*. Vol. 26. 3. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 593–602.
- Chen, Borting, Ho-Pang Hsu, and Yu-Lun Huang (Jan. 2016). “Bringing Desktop Applications to the Web”. In: *IT Professional* 18.1, pp. 34–40.
- Chen, Kunrong and Václav Rajlich (2010). “Case Study of Feature Location Using Dependence Graph, After 10 Years”. In: *IEEE International Conference on Program Comprehension*, pp. 1–3.
- Chinnici, Roberto, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana (2007). “Web Services Description Language (Wsdl) Version 2.0 Part 1: Core Language”. In: *W3C recommendation* 26.1, p. 19.
- Choudhary, Shauvik Roy, Mukul R. Prasad, and Alessandro Orso (May 2013). “X-Pert: Accurate Identification of Cross-Browser Issues in Web Applications”. In: *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, pp. 702–711.
- Chudnovskyy, Olexiy and Martin Gaedke (2010). “Development of Web 2.0 Applications Using WebComposition/Data Grid Service”. In: *The Second International Conferences on Advanced Service Computing (Service Computation 2010)* c, pp. 55–61.
- Coleman, Don, Dan Ash, Bruce Lowther, and Paul Oman (Aug. 1994). “Using Metrics to Evaluate Software System Maintainability”. In: *Computer* 27.8, pp. 44–49.
- Colosimo, Massimo, Andrea De Lucia, Rita Francese, and Giuseppe Scanniello (Oct. 2007). “Assessing Legacy System Migration Technologies Through Controlled Experiments”. In: *2007 IEEE International Conference on Software Maintenance*. IEEE, pp. 365–374.
- Creswell, John W (2014). *Research Design : Qualitative, Quantitative, and Mixed Methods Approaches*. 4. ed., in. Los Angeles: SAGE.
- Curtis, Bill, Michael Muller, Nagaraja S. Adiga, and Lev Lesokhin (2017). *CAST Research on Application Software Health Report - 2017 Global Sample*. Tech. rep. CAST.

- Curtis, Bill, Jay Sappidi, and Alexandra Szynkarski (2012). “Estimating the Principal of an Application’s Technical Debt”. In: *IEEE Software* 29.6, pp. 34–42.
- Daniel, Florian, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh (Jan. 2018). “Quality Control in Crowdsourcing”. In: *ACM Computing Surveys* 51.1, pp. 1–40. arXiv: 1801.02546.
- Deng, Jia Deng Jia, Wei Dong Wei Dong, R. Socher, et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2–9.
- Deshpande, Yogesh, San Murugesan, Athula Ginige, et al. (2002). “Web Engineering”. In: *Journal of Web Engineering* 1.1, pp. 3–17.
- Díaz, Oscar, Haritz Medina, and Felipe I Anfurrutia (2019). “Coding-Data Portability in Systematic Literature Reviews”. In: *Proceedings of the Evaluation and Assessment on Software Engineering - EASE ’19*. New York, New York, USA: ACM Press, pp. 178–187.
- Distante, Damiano, Gerardo Canfora, Scott R. Tilley, and Shihong Huang (2006). “Redesigning Legacy Applications for the Web With UWAT+: A Case Study”. In: *Proceeding of the 28th international conference on Software engineering - ICSE ’06*. New York, New York, USA: ACM Press, pp. 482–491.
- Distante, Damiano, T. Parveen, and Scott R. Tilley (2004). “Towards a Technique for Reverse Engineering Web Transactions From a User’s Perspective”. In: *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004*. IEEE, pp. 142–150.
- Distante, Damiano, V. Perrone, and M.A. Bochicchio (2002). “Migrating to the Web Legacy Application: The Sinfor Project”. In: *Proceedings. Fourth International Workshop on Web Site Evolution*. IEEE Comput. Soc, pp. 85–88.
- Distante, Damiano and Scott R. Tilley (2005). “Conceptual Modeling of Web Application Transactions: Towards a Revised and Extended Version of the UWA Transaction Design Model”. In: *Proceedings of the 11th International Multimedia Modelling Conference, MMM 2005*, pp. 439–445.

Distante, Damiano, Scott R. Tilley, and Gerardo Canfora (2006). "Towards a Holistic Approach to Redesigning Legacy Applications for the Web With UWAT+". In: *Conference on Software Maintenance and Reengineering (CSMR'06)*. IEEE, 5 pp.–299.

EU General Data Protection Regulation (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/Ec*.

European Commission (2004). "Project Cycle Management Guidelines". In: *Aid Delivery Methods* 1, p. 149.

Fahmideh, Mahdi, Farhad Daneshgar, and Fethi Rabhi (2018). "Cloud Migration Methodologies: Preliminary Findings". In: *Advances in Service-Oriented and Cloud Computing*. Vol. 707. February, pp. 112–122.

Felzenszwalb, P. F., Ross B. Girshick, D. McAllester, and D. Ramanan (Sept. 2010). "Object Detection With Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9, pp. 1627–1645.

Fielding, Roy T., Richard N. Taylor, Justin R. Erenkrantz, et al. (2017). "Reflections on the REST Architectural Style and "Principled Design of the Modern Web Architecture" (Impact Paper Award)". In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017*. Vol. Part F1301. New York, New York, USA: ACM Press, pp. 4–14.

Forrester Research (2011). *Application Modernization : Procrastinate at Your Peril !* Tech. rep.

Fowler, Martin and Jim Highsmith (2001). "The Agile Manifesto". In: *Software Development Magazine* 9(8).

Fowley, Frank, Divyaa Manimaran Elango, Hany Magar, and Claus Pahl (Oct. 2017). "Software System Migration to Cloud-Native Architectures for SME-Sized Software Vendors". In: *SOFSEM 2017: Theory and Practice of Computer Science*. IEEE, pp. 498–509.

- (2018). “The Benefits of Using Experimental Exploration for Cloud Migration Analysis and Planning”. In: *Cloud Computing and Service Science. CLOSER 2017. Communications in Computer and Information Science*. Ed. by Muñoz V. Ferguson D., Cardoso J., Helfert M., and Pahl C. Vol. 864. July. Springer, Cham, pp. 157–176.
 - Fowley, Frank and Claus Pahl (2018). “Cloud Migration Architecture and Pricing – Mapping a Licensing Business Model for Software Vendors to a SaaS Business Model”. In: *Advances in Service-Oriented and Cloud Computing*. Vol. 707. February. Springer International Publishing, pp. 91–103.
 - Frey, Sören and Wilhelm Hasselbring (2010). “Model-Based Migration of Legacy Software Systems to Scalable and Resource-Efficient Cloud-Based Applications: The CloudMIG Approach”. In: *Proceedings of the First International Conference on Cloud Computing, GRIDs and Virtualization*, pp. 155–158.
 - (Mar. 2011a). “An Extensible Architecture for Detecting Violations of a Cloud Environment’s Constraints During Legacy Software System Migration”. In: *2011 15th European Conference on Software Maintenance and Reengineering*. IEEE, pp. 269–278.
 - (2011b). “The CloudMIG Approach: Model-Based Migration of Software Systems to Cloud-Optimized Applications”. In: *International Journal on Advances in Software* 4.3, pp. 342–353.
- Frey, Sören, Wilhelm Hasselbring, and Benjamin Schnoor (Oct. 2012). “Automatic Conformance Checking for Migrating Software Systems to Cloud Infrastructures and Platforms”. In: *Journal of Software: Evolution and Process* 25.10, pp. 1089–1115. arXiv: 1408.1293.
- Fuentes-Fernández, Rubén, Juan Pavón, and Francisco Garijo (Mar. 2012). “A Model-Driven Process for the Modernization of Component-Based Systems”. In: *Science of Computer Programming* 77.3, pp. 247–269.
- Fuhr, Andreas, Tassilo Horn, Volker Riediger, and Andreas Winter (Feb. 2013). “Model-Driven Software Migration Into Service-Oriented Architectures”. In: *Computer Science - Research and Development* 28.1, pp. 65–84.

Gaedke, Martin (2000). "Komponententechnik Für Entwicklung Und Evolution Von Anwendungen Im World Wide Web". dissertation. Universität Karlsruhe, p. 225.

Gartner Research (2013). *Insights From the 2013 Gartner CIO Agenda Report: Hunting and Harvesting in a Digital World*. Tech. rep., pp. 1–12.

Gipp, Torsten and Andreas Winter (Oct. 2007). "Applying the ReMiP to Web Site Migration". In: *2007 9th IEEE International Workshop on Web Site Evolution*. IEEE, pp. 9–13.

Gitzel, Ralf, Axel Korthaus, and Martin Schader (2007). "Using Established Web Engineering Knowledge in Model-Driven Approaches". In: *Science of Computer Programming* 66.2, pp. 105–124.

Gold, N. E., Mark. Harman, D. Binkley, and R. M. Hierons (Aug. 2005). "Unifying Program Slicing and Concept Assignment for Higher-Level Executable Source Code Extraction". In: *Software: Practice and Experience* 35.10, pp. 977–1006.

Gordon, V.S. and J.M. Bieman (Jan. 1995). "Rapid Prototyping: Lessons Learned". In: *IEEE Software* 12.1, pp. 85–95.

Grechanik, Mark, Kevin M. Conroy, and Kishore S. Swaminathan (June 2007). "Creating Web Services From GUI-Based Applications". In: *IEEE International Conference on Service-Oriented Computing and Applications (SOCA '07)*. IEEE, pp. 72–79.

Grechanik, Mark, Chi Wu Mao, Ankush Baisal, David Rosenblum, and B.M. Mainul Hossain (July 2018). "Differencing Graphical User Interfaces". In: *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, pp. 203–214.

Grechanik, Mark, Qing Xie, and Chen Fu (Sept. 2009a). "Experimental Assessment of Manual Versus Tool-Based Maintenance of GUI-directed Test Scripts". In: *2009 IEEE International Conference on Software Maintenance*. IEEE, pp. 9–18.

– (2009b). "Maintaining and Evolving GUI-directed Test Scripts". In: *2009 IEEE 31st International Conference on Software Engineering*. IEEE, pp. 408–418.

- Grigera, Julián, Alejandra Garrido, José Matías Rivero, and Gustavo Rossi (2017). “Automatic Detection of Usability Smells in Web Applications”. In: *International Journal of Human Computer Studies* 97. September 2016, pp. 129–148.
- Hansen, Morten T., Nitin Nohria, and Thomas J. Tierney (1999). “What’s Your Strategy for Managing Knowledge?” In: *Harvard Business Review* 77.2, pp. 106–116.
- Heil, Sebastian, Maxim Bakaev, and Martin Gaedke (2016). “Measuring and Ensuring Similarity of User Interfaces: The Impact of Web Layout”. In: *Web Information Systems Engineering – WISE 2016*. Ed. by Wojciech Cellary, Mohamed F. Mokbel, Jianmin Wang, et al. Cham: Springer International Publishing, pp. 252–260.
- Heil, Sebastian, Felix Förster, and Martin Gaedke (2018). “Exploring Crowd-sourced Reverse Engineering”. In: *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*. Funchal, Portugal: SciTePress, pp. 147–158.
- Heil, Sebastian and Martin Gaedke (2016). “AWSM – Agile Web Migration for SMEs”. In: *Proceedings of the 11th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*. Rome, Italy: SciTePress, pp. 189–194.
- (2017). “Web Migration – A Survey Considering the SME Perspective”. In: *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*. Porto, Portugal: SciTePress, pp. 255–262.
- Heil, Sebastian, Valentin Siegert, and Martin Gaedke (2018). “ReWaMP: Rapid Web Migration Prototyping Leveraging WebAssembly”. In: *Web Engineering: Proceedings of 18th International Conference on Web Engineering (ICWE2018)*. Ed. by Tommi Mikkonen, Ralf Klamma, and Juan Hernández. Cáceres, Spain, pp. 84–92.

- Heil, Sebastian, Valentin Siegert, and Martin Gaedke (2019). "Crowdsourced Reverse Engineering: Experiences in Applying Crowdsourcing to Concept Assignment". In: *Evaluation of Novel Approaches to Software Engineering, Revised Selected Papers, Communications in Computer and Information Science*. Ed. by Ernesto Damiani, George Spanoudakis, and Leszek Maciaszek. Springer, pp. 215–239.
- Hopkins, Richard and Kevin Jenkins (2008). *Eating the IT Elephant: Moving From Greenfield Development to Brownfield*. 1st ed. Upper Saddle River, NJ, USA: IBM Press.
- Howe, Jeff (2006). "The Rise of Crowdsourcing". In: *Wired* 14.6.
- IDEO (2015). *The Field Guide to Human-Centered Design*. 1st ed. IDEO.org.
- IEEE Computer Society (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0*. Ed. by P. Bourque and R.E. Fairley. 3rd. Los Alamitos, CA, USA: IEEE Computer Society Press.
- IETF (1997). *RFC 2119 Key Words for Use in RFCs to Indicate Requirement Levels*.
- Jamshidi, Pooyan, Aakash Ahmad, and Claus Pahl (2013). "Cloud Migration Research: A Systematic Review". In: *IEEE Transactions on Cloud Computing* 1.2, pp. 142–157.
- Jamshidi, Pooyan, Claus Pahl, Samuel Chineneze, and Xiaodong Liu (2015). "Cloud Migration Patterns: A Multi-Cloud Service Architecture Perspective". In: ed. by Alex Norta, Walid Gaaloul, G. R. Gangadharan, and Hoa Khanh Dam. Vol. 9586. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 6–19.
- Joachims, Thorsten (1997). "A Probabilistic Analysis of the Rocchio Algorithm With TFIDF for Text Categorization." In: *Fourteenth International Conference on Machine Learning*, pp. 143–151.
- Kan, Stephen H. (1996). *Metrics and Models in Software Quality Engineering*. 3. print. Addison Wesley Longman, Inc.
- Kappel, Gerti, Birgitt Pröll, Siegfried Reich, and Werner Retschitzegger, eds. (2006). *Web Engineering – The Discipline of Systematic Development of Web Applications*. Wiley.

- Karampaglis, Zisis, Anakreon Mantis, Fotios Rafaillidis, Paschalis Tsolakis, and Apostolos Ampatzoglou (2014). “Secure Migration of Legacy Applications to the Web”. In: *International Conference on Software Engineering and Formal Methods*. Ed. by Antonio Cerone, Donatella Persico, Sara Fernandes, et al. Springer Berlin Heidelberg, pp. 229–243.
- Karnitis, Girts and Guntis Arnicans (June 2015). “Migration of Relational Database to Document-Oriented Database: Structure Denormalization and Data Transformation”. In: *2015 7th International Conference on Computational Intelligence, Communication Systems and Networks*. IEEE, pp. 113–118.
- Kassenärztliche Bundesvereinigung (2018). *IT in Der Arztpraxis - Verzeichnis Zertifizierter Software*. Tech. rep. Berlin: Kassenärztliche Bundesvereinigung, Dezernat Digitalisierung und IT.
- Kazman, Rick and Hong-Mei Chen (July 2009). “The Metropolis Model a New Logic for Development of Crowdsourced Systems”. In: *Communications of the ACM* 52.7, pp. 76–84.
- Kazman, Rick, S.G. Woods, and S.J. Carriere (1998). “Requirements for Integrating Software Architecture and Reengineering Models: CORUM II”. In: *Proceedings Fifth Working Conference on Reverse Engineering (Cat. No.98TB100261)*. IEEE Comput. Soc, pp. 154–163.
- Khadka, Ravi (2016). “Revisiting Legacy Software System Modernization”. dissertation. Utrecht University.
- Khadka, Ravi, Belfrit V. Batlajery, Amir M. Saeidi, Slinger Jansen, and Jurriaan Hage (2014). “How Do Professionals Perceive Legacy Systems and Software Modernization?” In: *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. New York, New York, USA: ACM Press, pp. 36–47.
- Khadka, Ravi, Gijs Reijnders, Amir M. Saeidi, Slinger Jansen, and Jurriaan Hage (Sept. 2011). “A Method Engineering Based Legacy to SOA Migration Method”. In: *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, pp. 163–172.

- Khadka, Ravi, Amir M. Saeidi, Andrei Idu, Jurriaan Hage, and Slinger Jansen (2013). “Legacy to SOA Evolution: A Systematic Literature Review”. In: *Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments*. Ed. by Anca Daniela Ionita, Marin Litoiu, and Grace Lewis. IGI Global. Chap. 3, pp. 40–71.
- Khusidman, Vitaly and William M. Ulrich (2007). *Architecture-Driven Modernization: Transforming the Enterprise*. Tech. rep.
- (2008). *Architecture-Driven Modernization: Transforming the Enterprise*. Tech. rep. OMG, pp. 1–7.
- Kienle, Holger M. and Damiano Distante (2014). “Evolution of Web Systems”. In: *Evolving Software Systems*. Ed. by Tom Mens, Alexander Serebrenik, and Anthony Cleve. 1st ed. Springer Berlin Heidelberg. Chap. 7, pp. 201–228.
- Koch, Nora, Alexander Knapp, Gefei Zhang, and Hubert Baumeister (2008). “Uml-Based Web Engineering”. In: *Web Engineering: Modelling and Implementing Web Applications*. Ed. by Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina. Human-Computer Interaction Series January. London: Springer London, pp. 157–191.
- Kong, Jun, Omer Barkol, Ruth Bergman, et al. (2012). “Web Interface Interpretation Using Graph Grammars”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4, pp. 590–602.
- Kong, Lanyan, Eleni Stroulia, and Bruce Matichuk (1999). “Legacy Interface Migration: A Task-Centered Approach”. In: *Proceedings of the 8th International Conference on Human-Computer Interaction*. Munich, pp. 1167–1171.
- Konstanteli, Kleopatra, Leire Orue-Echevarria, Hugo Brunelière, et al. (2015). *ARTIST Methodology Process Framework M30*. Tech. rep., pp. 1–73.
- Krasteva, Iva, Stavros Stavru, and Sylvia Ilieva (2013). “Agile Model-Driven Modernization to the Service Cloud”. In: *Proceedings of The Eighth International Conference on Internet and Web Applications and Services (ICIW 2013)*. Rome, Italy: Xpert Publishing Services, pp. 1–9.

- Kumar, Ranjitha, Arvind Satyanarayan, Cesar Torres, et al. (2013). "Webzeitgeist: Design Mining the Web". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. New York, New York, USA: ACM Press, p. 3083.
- Kumar, Ranjitha, Jerry O. Talton, Salman Ahmad, and Scott R Klemmer (2011). "Bricolage: Example-Based Retargeting for Web Design". In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. New York, New York, USA: ACM Press, p. 2197.
- Latoza, T. D. Thomas D. and André van der Hoek (2016). "Crowdsourcing in Software Engineering : Models , Opportunities , and Challenges". In: *IEEE Software*, pp. 1–13.
- Lewis, Grace, Edwin Morris, Liam O'Brien, Dennis B. Smith, and Lutz Wrage (2005). *SMART: The Service-Oriented Migration and Reuse Technique*. Tech. rep. CMU/SEI-2005-TN-029. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.
- Lewis, Grace, Edwin Morris, Dennis B. Smith, and Soumya Simanta (2008). *SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment*. Tech. rep. CMU/SEI-2008-TN-008. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.
- Li, Weijun, Xu Chen, and Z. M. Ma (July 2014). "Reengineering Fuzzy Nested Relational Databases Into Fuzzy XML Model". In: *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, pp. 1612–1617.
- Li, Zengyang, Paris Avgeriou, and Peng Liang (2015). "A Systematic Mapping Study on Technical Debt and Its Management". In: *Journal of Systems and Software* 101, pp. 193–220.
- Liao, T. Warren, Zhiming Zhang, and Claude R. Mount (June 1998). "Similarity Measures for Retrieval in Case-Based Reasoning Systems". In: *Applied Artificial Intelligence* 12.4, pp. 267–288.
- Liu, Wei, Xiaofeng Meng, and Weiyi Meng (Mar. 2010). "ViDE: A Vision-Based Approach for Deep Web Data Extraction". In: *IEEE Transactions on Knowledge and Data Engineering* 22.3, pp. 447–460.

- Liu, Weibo, Zidong Wang, Xiaohui Liu, et al. (Apr. 2017). “A Survey of Deep Neural Network Architectures and Their Applications”. In: *Neurocomputing* 234, pp. 11–26.
- Lucia, Andrea De, Rita Francese, Giuseppe Scanniello, and Genoveffa Tortora (Nov. 2008). “Developing Legacy System Migration Methods and Tools for Technology Transfer”. In: *Software: Practice and Experience* 38.13, pp. 1333–1364. arXiv: 1008.1900.
- Lucia, Andrea De, Rita Francese, Giuseppe Scanniello, Genoveffa Tortora, and Nicola Vitiello (Sept. 2006). “A Strategy and an Eclipse Based Environment for the Migration of Legacy Systems to Multi-Tier Web-Based Architectures”. In: *2006 22nd IEEE International Conference on Software Maintenance*. IEEE, pp. 438–447.
- Lucia, Andrea De, Massimiliano Di Penta, Filippo Lanobile, and Marco Torchiano (2009). “METAMORPHOS: MEthods and Tools for migrAting Software systeMs Towards Web and Service Oriented aRchitectures: exPerimental Evaluation, Usability, and tecHnOlogy tranSfer”. In: *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE, pp. 301–304.
- Lucia, Andrea De, Giuseppe Scanniello, and Genoveffa Tortora (Sept. 2007). “Identifying Similar Pages in Web Applications Using a Competitive Clustering Algorithm”. In: *Journal of Software Maintenance and Evolution: Research and Practice* 19.5, pp. 281–296. arXiv: 1408.1293.
- MacKenzie, C. Matthew, Ken Laskey, Francis McCabe, et al. (2006). “Reference Model for Service Oriented Architecture”. In: *OASIS standard* 12.
- Maenhaut, Pieter-Jan, Hendrik Moens, Veerle Ongenae, and Filip De Turck (Jan. 2016). “Migrating Legacy Software to the Cloud: Approach and Verification by Means of Two Medical Software Use Cases”. In: *Software: Practice and Experience* 46.1, pp. 31–54. arXiv: 1008.1900.
- Manolescu, Ioana, Marco Brambilla, Stefano Ceri, Sara Comai, and Piero Fraternali (2005). “Model-Driven Design and Deployment of Service-Enabled Web Applications”. In: *ACM Transactions on Internet Technology* 5.3, pp. 439–479.

- Mao, Ke, Licia Capra, Mark Harman, and Yue Jia (2017). “A Survey of the Use of Crowdsourcing in Software Engineering”. In: *Journal of Systems and Software* 126, pp. 57–84.
- Marchetto, Alessandro and Filippo Ricca (Oct. 2008). “Transforming a Java Application in an Equivalent Web-Services Based Application: Toward a Tool Supported Stepwise Approach”. In: *2008 10th International Symposium on Web Site Evolution*. IEEE, pp. 27–36.
- Marcotte, Ethan (2010). “Responsive Web Design”. In: *A List Apart* 306.
- Marcus, Andrian, Andrey Sergeyev, Václav Rajlich, and Jonathan I. Maletic (2004). “An Information Retrieval Approach to Concept Location in Source Code”. In: *Proceedings - Working Conference on Reverse Engineering, WCRE*, pp. 214–223.
- Masak, Dieter (2006). *Legacysoftware*. Xpert.press. Berlin/Heidelberg: Springer-Verlag.
- McCallum, Andrew and Kamal Nigam (1998). “A Comparison of Event Models for Naive Bayes Text Classification”. In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1, pp. 41–48.
- McDonald, Andrew and Ray Welland (2005). “Agile Web Engineering (AWE) Process: Perceptions Within a Fortune 500 Financial Services Company”. In: *Journal of Web Engineering* 4.4, pp. 283–321.
- McKeeman, William M. (1998). “Differential Testing for Software”. In: *Digital Technical Journal* 10.1, pp. 100–107.
- mediatixx GmbH & Co. KG (2018). *Medatixx - Zahlen, Daten, Fakten*.
- Mell, Peter and Timothy Grance (2011). “The NIST Definition of Cloud Computing”. In: *Recommendations of the National Institute of Standards and Technology Special Publication 800-145*.
- Meng, Xin, Jingwei Shi, Xiaowei Liu, Hufeng Liu, and Lian Wang (July 2011). “Legacy Application Migration to Cloud”. In: *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, pp. 750–751.

- Menychtas, Andreas, Kleopatra Konstanteli, Juncal Alonso, et al. (July 2014). “Software Modernization and Cloudification Using the ARTIST Migration Methodology and Framework”. In: *Scalable Computing: Practice and Experience* 15.2, pp. 131–152.
- Menychtas, Andreas, Christina Santzaridou, George Kousiouris, et al. (Sept. 2013). “ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud”. In: *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE, pp. 424–431.
- Merlo, E., P.-Y. Gagne, J.-F. Girard, et al. (1995). “Reengineering User Interfaces”. In: *IEEE Software* 12.1, pp. 64–73.
- Mitra, Nilo, Yves Lafon, et al. (2003). “Soap Version 1.2 Part 0: Primer”. In: *W3C recommendation* 24, p. 12.
- Mohagheghi, Parastoo, Arne J. Berre, Alexis Henry, Franck Barbier, and Andrey Sadovskyh (2010). “REMICS- REuse and Migration of Legacy Applications to Interoperable Cloud Services”. In: *Towards a Service-Based Internet*. Ed. by Witold Abramowicz, Ignacio M. Llorente, Mike Surridge, Andrea Zisman, and Julien Vayssiére. Springer Berlin Heidelberg, pp. 195–196.
- Mohagheghi, Parastoo and Thor Sæther (July 2011). “Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project”. In: *2011 IEEE World Congress on Services*. IEEE, pp. 507–514.
- Moreno, Nathalie, José Raúl Romero, and Antonio Vallecillo (2008). “An Overview of Model-Driven Web Engineering and the MDA”. In: *Web Engineering: Modelling and Implementing Web Applications*. Ed. by Gustavo Rossi, Oscar Pastor, Daniel Schwabe, and Luis Olsina. London: Springer London, pp. 353–382.
- NASCIO (2016). *State CIO Top Ten Priorities for 2017*. Tech. rep. NASCIO.
- Nasr, Khalid Adam, H. Gross, and A. van Deursen (Mar. 2010). “Adopting and Evaluating Service Oriented Architecture in Industry”. In: *2010 14th European Conference on Software Maintenance and Reengineering*. IEEE, pp. 11–20.

- Nebeling, Michael, Stefania Leone, and Moira C. Norrie (2012). "Crowdsourced Web Engineering and Design". In: *Web Engineering*. Ed. by Marco Brambilla, Takehiro Tokuda, and Robert Tolksdorf. Berlin, Germany: Springer Berlin Heidelberg, pp. 31–45.
- Nebeling, Michael and Moira C. Norrie (2013). "Responsive Design and Development: Methods, Technologies and Current Issues". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 510–513.
- Nebeling, Michael, Maximilian Speicher, and Moira C. Norrie (2013). "CrowdAdapt: Enabling Crowdsourced Web Page Adaptation for Individual Viewing Conditions and Preferences". In: *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing system*, pp. 23–32.
- Neumann, Andy, Nuno Laranjeiro, and Jorge Bernardino (2018). "An Analysis of Public REST Web Service APIs". In: *IEEE Transactions on Services Computing* November, pp. 1–1.
- Nguyen, Dinh Khoa, Willem-Jan van den Heuvel, Mike P. Papazoglou, Valeria de Castro, and Esperanza Marcos (July 2009). "Gap Analysis Methodology for Business Service Engineering". In: *2009 IEEE Conference on Commerce and Enterprise Computing*. IEEE, pp. 215–220.
- Nielsen, Jakob and Thomas K. Landauer (1993). "A Mathematical Model of the Finding of Usability Problems". In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '93*. New York, New York, USA: ACM Press, pp. 206–213.
- Nonaka, Ikujiro (2008). *The Knowledge-Creating Company*. Harvard Business Review Press.
- O'Reilly, Tim (2007). "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software". In: *Communications & Strategies* 1.First Quarter, pp. 17–38.
- Oliva, Aude and Antonio Torralba (2001). "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope". In: *International Journal of Computer Vision* 42.3, pp. 145–175.

Orue-Echeverria, Leire, Juncal Alonso, Marisa Escalante, et al. (2014). *ARTIST Methodology M30*. Tech. rep., pp. 15–22.

Oulasvirta, Antti, Aliaksei Miniukovich, Gregorio Palmas, et al. (2018). “Aalto Interface Metrics (AIM)): A Service and Codebase for Computational GUI Evaluation”. In: *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings - UIST ’18 Adjunct*. New York, New York, USA: ACM Press, pp. 16–19.

Pahl, Claus, Huanhuan Xiong, and Ray Walshe (2013). “A Comparison of On-Premise to Cloud Migration Approaches”. In: *Service-Oriented and Cloud Computing*. Ed. by Kung-Kiu Lau, Winfried Lamersdorf, and Ernesto Pimentel. Springer Berlin Heidelberg, pp. 212–226.

Pennington, Nancy (July 1987). “Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs”. In: *Cognitive Psychology* 19.3, pp. 295–341.

Pérez-Castillo, Ricardo, Ignacio García Rodríguez De Guzmán, and Mario Piattini (2011). “Knowledge Discovery Metamodel-Iec 19506: A Standard to Modernize Legacy Systems”. In: *Computer Standards and Interfaces* 33.6, pp. 519–532.

Pérez-Castillo, Ricardo, Ignacio García-Rodríguez de Guzmán, Ismael Caballero, and Mario Piattini (May 2013). “Software Modernization by Recovering Web Services From Legacy Databases”. In: *Journal of Software: Evolution and Process* 25.5, pp. 507–533.

Pérez-Castillo, Ricardo, Ignacio Garcia-Rodriguez de Guzman, Mario Piattini, and Christof Ebert (Nov. 2011). “Reengineering Technologies”. In: *IEEE Software* 28.6, pp. 13–17.

Pérez-Castillo, Ricardo, Ignacio Garcia-Rodríguez de Guzmán, and Ismael Caballero (2009). “PRECISO: A Reengineering Process and a Tool for Database Modernisation Through Web Services”. In: *Proceedings of the 2009 ACM symposium on Applied Computing - SAC ’09*. New York, New York, USA: ACM Press, pp. 2126–2130.

- Pérez-Castillo, Ricardo, Ignacio García-Rodríguez de Guzmán, and Mario Piattini (Oct. 2011). “Business Process Archeology Using MARBLE”. In: *Information and Software Technology* 53.10, pp. 1023–1044.
- Politis, David (2017). *2017 State of the SaaS-Powered Workplace*. Tech. rep. BetterCloud.
- Pols, Axel, Katja Hampe, Franz Grimm, and Christopher Meinecke (2016). *Digital Banking*. Tech. rep. Berlin: Bitkom Research.
- Puder, Arno (2004). “Extending Desktop Applications to the Web”. In: *Proceedings of the 2004 International Symposium on Information and Communication Technologies*. Las Vegas, Nevada, USA: Trinity College Dublin, pp. 8–13.
- Rajlich, V. and N. Wilde (2002). “The Role of Concepts in Program Comprehension”. In: *Proceedings 10th International Workshop on Program Comprehension*. IEEE Comput. Soc, pp. 271–278.
- El-Ramly, Mohammad, Eleni Stroulia, and Paul Sorenson (2002). “Mining System-User Interaction Traces for Use Case Models”. In: *Proceedings - IEEE Workshop on Program Comprehension 2002*-Janua, pp. 21–29.
- Razavian, Maryam (2009). “Knowledge-Driven SOA Migration”. In: *CEUR Workshop Proceedings*, pp. 13–18.
- (2013). “Knowledge-Driven Migration to Services”. dissertation. Vrije Universiteit Amsterdam.
- Razavian, Maryam and Patricia Lago (2010). “Understanding SOA Migration Using a Conceptual Framework”. In: *Journal of Systems Integration*, pp. 33–43.
- (2011). “A Survey of SOA Migration in Industry”. In: *Service-Oriented Computing*. Ed. by Gerti Kappel, Zakaria Maamar, and Hamid R. Motahari-Nezhad. Springer Berlin Heidelberg, pp. 618–626.
- (2012). “A Lean and Mean Strategy for Migration to Services”. In: *Proceedings of the WICSA/ECSA 2012 Companion Volume on - WICSA/ECSA '12*. August. New York, New York, USA: ACM Press, p. 61.
- (2014). “A Lean and Mean Strategy : A Data Migration Industrial Study”. In: *Journal of Software: Evolution and Process* 26.2, pp. 141–171.

- Razavian, Maryam, Dinh Khoa Nguyen, Patricia Lago, and Willem-Jan van den Heuvel (2010). “The SAPIENSA Approach for Service-Enabling Pre-Existing Legacy Assets”. In: *Proceedings of the International Workshop on SOA Migration and Evolution (SOAME) 2010*. Ed. by G. Lewis, F. Ricca, M. Postina, U. Steffens, and A. Winter. August 2014. OFFIS, pp. 21–30.
- Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (June 2017). “Faster R-Cnn: Towards Real-Time Object Detection With Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6, pp. 1137–1149. arXiv: 1506.01497.
- Rivero, José Matías, Sebastian Heil, Julián Grigera, Martin Gaedke, and Gustavo Rossi (2013). “MockAPI: An Agile Approach Supporting API-first Web Application Development”. In: *Web Engineering: Proceedings of the 13th International Conference on Web Engineering (ICWE2013)*. Ed. by Florian Daniel, Peter Dolog, and Quing Li. Aalborg, Denmark: Springer Berlin Heidelberg, pp. 7–21.
- Rivero, José Matías, Sebastian Heil, Julián Grigera, Esteban Robles Luna, and Martin Gaedke (2014). “An Extensible, Model-Driven and End-User Centric Approach for API Building”. In: *Web Engineering: Proceedings of the 14th International Conference (ICWE2014)*. Ed. by S. Casteleyn, G. Rossi, and M. Winckler. Vol. 8541. Toulouse, France: Springer, Cham, pp. 494–497.
- Rivero, José Matías and Gustavo Rossi (2013). “MockupDD: Facilitating Agile Support for Model-Driven Web Engineering”. In: *Current Trends in Web Engineering: ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8–12, 2013. Revised Selected Papers*. Ed. by Quan Z. Sheng and Jesper Kjeldskov. Cham: Springer International Publishing, pp. 325–329.
- Rocha, Leonardo, Fernando Vale, Elder Cirilo, Dárlinton Barbosa, and Fernando Mourão (2015). “A Framework for Migrating Relational Datasets to NoSQL1”. In: *Procedia Computer Science* 51, pp. 2593–2602.

- Rodríguez-Echeverría, Roberto, José María Conejero, Pedro J. Clemente, Juan Carlos Preciado, and Fernando Sánchez-Figueroa (2012). “Modernization of Legacy Web Applications Into Rich Internet Applications”. In: *7th Model-Driven Web Engineering Workshop, in conjunction with International Conference on Web Engineering (ICWE2012)*, pp. 236–250.
- Rodríguez-Echeverría, Roberto, José María Conejero, Marino Linaje, Juan Carlos Preciado, and Fernando Sánchez-Figueroa (2010). “Re-Engineering Legacy Web Applications Into Rich Internet Applications”. In: *Web Engineering*, pp. 189–203.
- Rose, Jeremy, Matthew Jones, and Brent Furneaux (Apr. 2016). “An Integrated Model of Innovation Drivers for Smaller Software Firms”. In: *Information & Management* 53.3, pp. 307–323.
- Roy Choudhary, Shauvik, Mukul R. Prasad, and Alessandro Orso (2014). “X-Pert: A Web Application Testing Tool for Cross-Browser Inconsistency Detection”. In: *Proceedings of the 2014 International Symposium on Software Testing and Analysis - ISSTA 2014*. New York, New York, USA: ACM Press, pp. 417–420.
- Roy Choudhary, Shauvik, Husayn Versee, and Alessandro Orso (Sept. 2010). “A Cross-Browser Web Application Testing Tool”. In: *2010 IEEE International Conference on Software Maintenance*. IEEE, pp. 1–6.
- Saar, Tõnis, Marlon Dumas, Marti Kaljuve, and Nataliia Semenenko (Nov. 2016). “Browserbite: Cross-Browser Testing via Image Processing”. In: *Software: Practice and Experience* 46.11, pp. 1459–1477.
- Sadovykh, Andrey, Christian Hein, Brice Morin, Parastoo Mohagheghi, and Arne J. Berre (2011). “REMICS- REuse and Migration of Legacy Applications to Interoperable Cloud Services”. In: *Towards a Service-Based Internet*. 257793, pp. 315–316.
- Sanoja, Andres and Stephane Gancarski (Apr. 2014). “Block-O-Matic: A Web Page Segmentation Framework”. In: *2014 International Conference on Multimedia Computing and Systems (ICMCS)*. Vol. 0. IEEE, pp. 595–600.
- Sappidi, Jay, Bill Curtis, and Alexandra Szynkarski (2011). *The CRASH Report - 2011/12 Summary of Key Findings*. Tech. rep. CAST Research Labs.

- Satzger, Benjamin, Rostyslav Zabolotnyi, Schahram Dustdar, et al. (2014). “Toward Collaborative Software Engineering Leveraging the Crowd”. In: *Economics-Driven Software Architecture*. Elsevier, pp. 159–182.
- Sauro, Jeff and James R. Lewis (2016). *Quantifying the User Experience: Practical Statistics for User Research*. 2nd ed. Elsevier LTD, Oxford.
- Saxe, Joshua, Rafael Turner, and Kristina Blokhin (Oct. 2014). “CrowdSource: Automated Inference of High Level Malware Functionality From Low-Level Symbols Using a Crowd Trained Machine Learning Model”. In: *2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*. IEEE, pp. 68–75. arXiv: 1605 .08642.
- Schueffel, Patrick (2016). “Taming the Beast: A Scientific Definition of Fintech”. In: *Journal of Innovation Management* 4.4, pp. 32–54.
- Schwabe, Daniel, Gustavo Rossi, and Simone D. J. Barbosa (1996). “Systematic Hypermedia Application Design With OOHDM”. In: *Proceedings of the the seventh ACM conference on Hypertext - HYPERTEXT '96*. New York, New York, USA: ACM Press, pp. 116–128.
- Seacord, Robert C., Grace Lewis, and Daniel Plakosh (2003). *Modernizing Legacy Systems Software Technologies, Engineering Processes, and Business Practices*. Boston: Addison-Wesley.
- Sjoberg, Dag I.K., Aiko Yamashita, Bente C.D. Anda, Audris Mockus, and Tore Dyba (Aug. 2013). “Quantifying the Effect of Code Smells on Maintenance Effort”. In: *IEEE Transactions on Software Engineering* 39.8, pp. 1144–1156.
- Sneed, Harry M (1995). “Planning the Reengineering of Legacy Systems”. In: *IEEE Software* 12.1, pp. 24–34.
- Sneed, Harry M., Ellen Wolf, and Heidi Heilmann (2010a). “Der Reference Migration Process (ReMiP)”. In: *Software-Migration in der Praxis: Übertragung alter Softwaresysteme in eine moderne Umgebung*. 1st ed. dpunkt Verlag. Chap. 4, pp. 85–132.
- (2010b). *Softwaremigration in Der Praxis: Übertragung Alter Softwaresysteme in Eine Moderne Umgebung*. dpunkt Verlag.

- Sosa, Encarna, Pedro J. Clemente, José María Conejero, and Roberto Rodríguez-Echeverría (Sept. 2013). “A Model-Driven Process to Modernize Legacy Web Applications Based on Service Oriented Architectures”. In: *2013 15th IEEE International Symposium on Web Systems Evolution (WSE)*. IEEE, pp. 61–70.
- Sosa-Sánchez, Encarna, Pedro J. Clemente, Alvaro E. Prieto, José María Conejero, and Roberto Rodríguez-Echeverría (2017). “MigraSOA: Migration of Legacy Web Applications to Service Oriented Architectures (SOA)”. In: *IEEE Latin America Transactions* 15.7, pp. 1306–1311.
- Sosa-Sánchez, Encarna, Pedro J. Clemente, Miguel Sanchez-Cabrera, et al. (June 2014). “Service Discovery Using a Semantic Algorithm in a SOA Modernization Process From Legacy Web Applications”. In: *2014 IEEE World Congress on Services*. i. IEEE, pp. 470–477.
- Statista (2018b). *Percentage of All Global Web Pages Served to Mobile Phones From 2009 to 2018*. Tech. rep.
- Stol, Klaas-Jan and Brian Fitzgerald (2014). “Two’s Company, Three’s a Crowd: A Case Study of Crowdsourcing Software Development”. In: *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. New York, New York, USA: ACM Press, pp. 187–198.
- Strauch, Steve, Vasilios Andrikopoulos, Thomas Bachmann, et al. (Dec. 2013). “Decision Support for the Migration of the Application Database Layer to the Cloud”. In: *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. Vol. 1. IEEE, pp. 639–646.
- Stroulia, Eleni and Rohit V. Kapoor (2002). “Reverse Engineering Interaction Plans for Legacy Interface Migration”. In: *Computer-Aided Design of User Interfaces III*. Ed. by C. Kolski and J. Vanderdonckt. Dordrecht: Springer Netherlands, pp. 295–310.
- Stroulia, Eleni, Mohammad El-Ramly, P. Iglinski, and Paul Sorenson (July 2003). “User Interface Reverse Engineering in Support of Interface Migration to the Web”. In: *Automated Software Engineering* 10.3, pp. 271–301.

- Stroulia, Eleni, Mohammad El-Ramly, Lanyan Kong, P. Sorenson, and Bruce Matichuk (1999). “Reverse Engineering Legacy Interfaces: An Interaction-Driven Approach”. In: *Sixth Working Conference on Reverse Engineering (Cat. No.PR00303)*. IEEE Comput. Soc, pp. 292–302.
- Stroulia, Eleni, Mohammad El-Ramly, and Paul Sorenson (2002). “From Legacy to Web Through Interaction Modeling”. In: *International Conference on Software Maintenance, 2002. Proceedings*. IEEE Comput. Soc, pp. 320–329.
- Tak, Byung Chul and Chunqiang Tang (June 2014). “AppCloak: Rapid Migration of Legacy Applications Into Cloud”. In: *2014 IEEE 7th International Conference on Cloud Computing*. IEEE, pp. 810–817.
- Torchiano, Marco, Massimiliano Di Penta, Filippo Ricca, Andrea De Lucia, and Filippo Lanubile (Sept. 2008). “Software Migration Projects in Italian Industry: Preliminary Results From a State of the Practice Survey”. In: *Proceedings of Evol 2008: 4th Intl. ERCIM Workshop on Software Evolution and Evolvability*. 1. IEEE, pp. 35–42.
- Tripp, Steven D. and Barbara Bichelmeyer (Mar. 1990). “Rapid Prototyping: An Alternative Instructional Design Strategy”. In: *Educational Technology Research and Development* 38.1, pp. 31–44.
- Tuch, Alexandre N., Javier A. Bargas-Avila, Klaus Opwis, and Frank H. Wilhelm (Sept. 2009). “Visual Complexity of Websites: Effects on Users’ Experience, Physiology, Performance, and Memory”. In: *International Journal of Human-Computer Studies* 67.9, pp. 703–715.
- Tunkelang, Daniel (Jan. 2009). “Faceted Search”. In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 1.1, pp. 1–80.
- Tzvetkov, Ventzislav and Xiong Wang (2005). “DBXML - Connecting XML With Relational Databases”. In: *The Fifth International Conference on Computer and Information Technology (CIT'05)*. IEEE, pp. 130–135.
- Ulrich, William M. (2011). *Architecture - Driven Modernization 101: Concepts, Strategies & Justification*. Tech. rep.
- Vavliakis, Konstantinos N., Theofanis K. Grollios, and Pericles A. Mitkas (Jan. 2013). “RDATE – Publishing Relational Databases Into the Semantic Web”. In: *Journal of Systems and Software* 86.1, pp. 89–99.

- Vavliakis, Konstantinos N., Andreas L. Symeonidis, Georgios T. Karagiannis, and Pericles A. Mitkas (Apr. 2011). “An Integrated Framework for Enhancing the Semantic Transformation, Editing and Querying of Relational Databases”. In: *Expert Systems with Applications* 38.4, pp. 3844–3856.
- Wagner, Christian (2014). *Model-Driven Software Migration: A Methodology*. Wiesbaden: Springer Vieweg.
- Wallmüller, E (2001). *Software-Qualitätsmanagement in Der Praxis: Software-Qualität Durch Führung Und Verbesserung Von Software-Prozessen*. 2., völlig. München: Hanser.
- Wang, Shuen-tai and Hsi-ya Chang (2014). “Development of Web-Based Remote Desktop to Provide Adaptive User Interfaces in Cloud Platform”. In: *International Journal of Computer, Electrical, Automaton, Control and Information Engineering* 8.8, pp. 1241–1245.
- Warren, Ian (2012). *The Renaissance of Legacy Systems: Method Support for Software-System Evolution*. Springer Science & Business Media.
- Warren, Ian and J. Ransom (2002). “Renaissance: A Method to Support Software System Evolution”. In: *Proceedings 26th Annual International Computer Software and Applications*. IEEE Comput. Soc, pp. 415–420.
- Watanabe, Shinya, Tomoyuki Hiroyasu, and Mitsunori Miki (2002). “NCGA : Neighborhood Cultivation Genetic Algorithm for Multi-Objective Optimization Problems”. In: *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning*. Vol. 6930, pp. 198–202.
- Watanabe, Willian Massami, Giovana Lázaro Amêndola, and Fagner Christian Paes (Mar. 2019). “Layout Cross-Platform and Cross-Browser Incompatibilities Detection Using Classification of DOM Elements”. In: *ACM Transactions on the Web* 13.2, pp. 1–27.
- Weidema, Edgar R.Q., Consuelo López, Sahand Nayebaziz, Fernando Spanghero, and André van der Hoek (2016). “Toward Microtask Crowdsourcing Software Design Work”. In: *Proceedings of the 3rd International Workshop on Crowd-Sourcing in Software Engineering - CSI-SE '16*. New York, New York, USA: ACM Press, pp. 41–44.

Weise, Thomas (2009). *Global Optimization Algorithm: Theory and Application*. 2nd Ed. Vol. 1. Thomas Weise.

Wendland, Marc-Florian, Marco Kranz, Christian Hein, Tom Ritter, and Ana García Flaquer (2013). “Model-Based Testing in Legacy Software Modernization: An Experience Report”. In: *Proceedings of the 2013 International Workshop on Joining AcadeMiA and Industry Contributions to testing Automation - JAMAICA 2013*. New York, New York, USA: ACM Press, pp. 35–40.

Winter, Andreas, C. Zillmann, A. Herget, et al. (2011). “SOAMIG Project: Model-Driven Software Migration Towards Service-Oriented Architectures”. In: *First International Workshop on Model-Driven Software Migration (MDSM 2011)*, pp. 15–16.

Xuan Fan, Pingjian Zhang, and Juanjuan Zhao (June 2010). “Transformation of Relational Database Schema to Semantics Web Model”. In: *2010 Second International Conference on Communication Systems, Networks and Applications*. Vol. 1. IEEE, pp. 379–384.

Yli-Huumo, Jesse, Andrey Maglyas, and Kari Smolander (2016). “How Do Software Development Teams Manage Technical Debt? – An Empirical Study”. In: *Journal of Systems and Software* 120, pp. 195–218.

Zakai, Alon (2011). “Emscripten: An LLVM-to-JavaScript Compiler”. In: *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pp. 301–312.

Zhao, Gansen, Qiaoying Lin, Libo Li, and Zijing Li (Nov. 2014). “Schema Conversion Model of SQL Database to NoSQL”. In: *2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. IEEE, pp. 355–362.

Zillmann, C., Andreas Winter, A. Herget, et al. (Mar. 2011). “The SOAMIG Process Model in Industrial Applications”. In: *2011 15th European Conference on Software Maintenance and Reengineering*. April 2009. IEEE, pp. 339–342.

Standards and Specifications

IEEE Computer Society (1998). *IEEE 1219-1998 Standard for Software Maintenance.*

ISO (2010). *International Standard ISO 9241-210 Ergonomics of Human–system Interaction — Part 210: Human-Centred Design for Interactive Systems.*

ISO/IEC (2009). *International Standard ISO/IEC 20926:2009 IFPUG Functional Size Measurement Method 2009.*

ISO/IEEE (2006). *International Standard ISO/IEC 14764 Software Engineering — Software Life Cycle Processes — Maintenance.*

- (2017a). *ISO/IEC/IEEE International Standard 15939:2017 Systems and Software Engineering – Measurement Process.*
- (Aug. 2017b). *ISO/IEC/IEEE International Standard 24765:2017 Systems and Software Engineering – Vocabulary.*

OASIS (2007). *Web Services Business Process Execution Language Version 2.0.*

Object Management Group (2011). *Architecture-Driven Modernization : Abstract Syntax Tree Metamodel (ASTM) Version 1.0.*

- (2012). *Structured Metrics Metamodel (SMM) Version 1.2.*
- (2013). *Business Process Model and Notation (BPMN) Version 2.0.2.*
- (2014). *Model Driven Architecture (MDA) Version 2.0.*
- (2015). *Interaction Flow Modeling Language Version 1.0.*
- (2016a). *Architecture-Driven Modernization: Knowledge Discovery Meta-Model (KDM) Version 1.4.*
- (2016b). *Query/View/Transformation Specification Version 1.3.*
- (2017). *OMG Unified Modeling Language Version 2.5.1.*

W3C (2015). *W3C Document Object Model, W3C Recommendation, 19 November 2015.*

- (2017a). *CSS Grid Layout Module Level 1, W3C Candidate Recommendation, 14 December 2017.*

W3C (2017b). *Web Annotation Data Model*, W3C Recommendation, 23 February 2017.

- (2018). *Web Components*, W3C Working Group Note, 01 March 2018.
- (2019). *WebAssembly Core Specification*, W3C Recommendation, 5 December 2019.

Online References

Berners-Lee, Tim (1990). *Information Management: A Proposal*. URL: <http://info.cern.ch/Proposal.html> (visited on July 11, 2018).

Chan, Viva (2018). *60 SAAS Statistics That Will Change the Way You Think*. URL: <https://financesonline.com/saas-statistics/> (visited on June 10, 2018).

Cunningham, Ward (1992). *The WyCash Portfolio Management System*. URL: <http://c2.com/doc/oops1a92.html> (visited on Aug. 10, 2018).

Gartner (2014). *High Failure Rates in Insurance Legacy Modernization Challenge CIOs*. URL: <https://www.gartner.com/doc/2653016/high-failure-rate-insurance-legacy> (visited on Dec. 6, 2019).

- (2016). *The Key to Business Transformation Is Culture*. URL: <https://www.gartner.com/smarterwithgartner/the-key-to-business-transformation-is-culture/> (visited on Feb. 11, 2018).

Knorr, Eric (2003). *2004: The Year of Web Services*. URL: <https://www.cio.com/article/2439869/web-services/2004--the-year-of-web-services.html> (visited on July 11, 2018).

Lewis, Grace (2010). *SMART: SOA Migration, Adoption, and Reuse Technique*. URL: https://resources.sei.cmu.edu/library/asset-view.cfm?asset_id=30930 (visited on Dec. 6, 2019).

Pettey, Christy and Rob van der Meulen (2012). *Gartner Survey Shows 71 Percent of Respondents Using SaaS for Less Than Three Years*. URL: <http://www.gartner.com/newsroom/id/2253215> (visited on Oct. 26, 2019).

- (2013). *Gartner Executive Program Survey of More Than 2,000 CIOs Shows Digital Technologies Are Top Priorities in 2013*. URL: <https://www.gartner.com/newsroom/id/2304615> (visited on Aug. 10, 2018).
 - Statista (2018a). *Cloud Software as a Service - Statistics & Facts* | Statista. URL: <https://www.statista.com/topics/3071/cloud-software-as-a-service-saas/> (visited on Dec. 6, 2019).
 - (2018c). *Total Size of the Public Cloud Software as a Service (SaaS) Market From 2008 to 2020 (In Billion U.S. Dollars)*. URL: <https://www.statista.com/statistics/510333/worldwide-public-cloud-software-as-a-service/> (visited on Dec. 6, 2019).
- Wessa, P (2017). *Multiple Regression (V1.0.48)*. URL: https://www.wessa.net/rwasp_multpleregression.wasp (visited on Mar. 31, 2019).

List of Acronyms

ADM	Architecture-Driven Modernization. 43–46, 62, 66, 67, 76, 78, 100, 419, 422, 423
API	Application Programming Interface. 61, 75, 148, 151, 157, 159, 196–198, 200, 201, 204, 207, 210, 233
AST	Abstract Syntax Tree. 45, 46, 211
ASTM	Abstract Syntax Tree Metamodel, OMG ADM standard. 45, 46, 67, 141, 420, 422
ATL	ATLAS Transformation Language. 79–81, 100
AWSM	Agile Web Migration for SMEs. 14, 15, 105, 108–111, 114–116, 118, 119, 121–124, 127–131, 133–138, 142, 143, 147, 151, 175, 188, 192, 199, 255, 260, 277, 297, 299–312, 317, 318, 320–322, 325, 326, 329, 419, 422
AWSMAP	AWSM Annotation Platform. 151, 153–157, 159, 160, 163, 164, 168, 176, 327
BPMN	Business Process Model and Notation, OMG standard. 58, 67, 80, 133, 153, 214, 326
CIM	Computation-Independent Model. 46, 161, 223, 224, 229, 233
CRUD	Create, Read, Update, Delete. 19, 21
CRUDS	Create, Read, Update, Delete, Search. 201
CSS	Cascading Style Sheets. 4, 83, 191, 207, 218, 229, 233, 258, 259, 286, 300, 301

DOM	Document Object Model, W3C standard. 118, 126, 261, 262
DSL	Domain-specific Language. 70
EMP	Eclipse Modeling Project. 100
FHOG	Felzenszwalb histogram of oriented gradients. 280, 281
GASTM	Generic Abstract Syntax Tree Metamodel, part of ASTM specification. 45
GUI	Graphical User Interface. 19, 20, 55, 57, 58, 61–63, 67, 69, 73, 74, 79, 82, 83, 85, 88, 89, 209, 211, 258, 259, 261–263, 265, 266, 269, 270, 277, 279, 281, 287, 380
HCD	Human-Centered Design. 11, 23–27, 106–108, 320, 324, 331
HCI	Human-Computer interaction. 258, 280, 288, 293
HMW	How Might We. 11, 106, 107
HTML	HyperText Markup Language. 4, 20, 73, 75, 77, 82, 84, 85, 90, 191, 207, 229, 230, 233, 262, 286, 300, 301
HTTP	HyperText Transfer Protocol. 3, 4, 51, 191, 300
IaaS	Infrastructure as a Service. 70, 71
IDE	Integrated Development Environment. 123, 150, 157, 158, 304
IEC	International Electrotechnical Commission. 2, 44
IEEE	Institute of Electrical and Electronics Engineers. 2, 50, 275, 276
IFML	The Interaction Flow Modeling Language, OMG standard. 42, 424
ISO	International Organization for Standardization. 2, 24, 44, 50

ISV	Independent Software Vendor. 1, 4, 5, 7–11, 16–19, 22–29, 31, 33–39, 42, 72, 81, 82, 93, 95, 96, 101, 102, 104, 105, 107, 108, 113, 114, 116, 119, 122, 127, 137, 141, 142, 147, 149, 152, 156–159, 165, 170, 171, 174, 175, 186–188, 190–193, 198, 200, 203, 229, 240–242, 258, 260, 264, 266, 268, 275, 285, 302–304, 310–312, 318–325, 329, 330
KDM	Knowledge Discovery Meta-Model, OMG standard. 44–46, 67, 70, 79, 130–132, 134, 138, 141, 152, 162, 209, 211, 278, 303, 308, 320
LFA	Logical Framework Approach. 23–26, 28
LOC	Lines of Code. 7, 9, 20, 46, 176, 178, 233
LOD	Linked Open Data. 161, 330
MDA	Model Driven Architecture, OMG standard. 43, 67
MDE	Model-Driven Engineering. 128
MDWE	Model-Driven Web Engineering. 194, 201
MFC	Microsoft Foundation Class Library. 20, 21, 131, 187, 192, 208, 223, 224, 230, 248, 253
MOF	Meta Object Facility, OMG standard. 44–46, 100
MVC	Model-View-Controller. 57, 77, 80, 81, 83, 85, 86, 203, 208
OA	Web Annotation Data Model, W3C standard. 161
OCR	Optical Character Recognition. 279, 280
OMG	Object Management Group. 43, 44, 67, 70, 76, 100, 127, 130, 132, 141, 320, 419–424
OWL	Web Ontology Language, W3C standard. 113, 161, 163, 175, 330
PaaS	Platform as a Service. 70, 72, 194

PIM	Platform-Independent Model. 46, 62
PMS	Patient Management System. 18, 19, 22, 28, 258
PSM	Platform-Specific Model. 46, 62, 161, 170, 224, 229, 233
QVT	Query/View/Transformation, OMG ADM standard. 46, 47, 62, 79, 100, 163
RDF	Resource Description Framework, W3C standard. 163, 175
ReMiP	Reference Migration Process. 31, 32, 39, 47–49, 61, 63, 74, 75, 90, 101, 110, 111, 300, 306, 425, 430
REST	Representational State Transfer. 51, 157, 159, 197, 198, 201, 210, 233
ReWaMP	Rapid Web Migration Prototyping leveraging WebAssembly. 206–211, 213, 214, 233, 234, 236, 238–243, 245–247, 300, 302, 303, 328
RIA	Rich Internet Application. 78, 79
ROI	Region of Interest. 262, 267, 274, 277, 279–281
RWMPA	Rapid Web Migration Prototyping Assistance. 213–216, 242–244, 246, 247, 300, 303, 328
S2DCS	AWSM Strategy Selection Decision Support System (S2DCS), cf. section 4.4.1. 119–121, 300, 306, 311
SaaS	Software as a Service. 3–6, 63–65, 67–72, 84, 307
SASTM	Specific Abstract Syntax Tree Metamodel, part of ASTM specification. 46
SCKM	Source Code Knowledge Model, cf. section 4.6. 131–134, 146, 148, 149, 161, 162, 176, 301, 303, 308, 312, 326
SLOC	Source Lines of Code. 20, 234, 236, 240

SME	Small and Medium-sized Enterprise. 17–19, 23, 25, 39, 72, 82, 101, 102, 104, 105, 108, 137, 171, 310, 311, 320, 324, 329
SMM	Structured Metrics Metamodel, OMG ADM standard. 46, 70, 276
SOA	Service-oriented Architecture. 3, 13, 33, 49, 51–54, 56, 58, 62, 63, 66–68, 77, 78, 80, 81, 86, 95–97, 331
SOA-MF	SOA Migration Framework. 56, 100
TFS	Microsoft Team Foundation Server. 159
TUI	Text-based User Interface. 81, 84, 86, 89, 94, 103, 263, 380
UI	User Interface. 33, 71–75, 77, 82–90, 95–97, 117, 136, 137, 144, 159, 193, 197, 198, 200, 202, 203, 207–210, 213, 233, 242, 246, 247, 254, 255, 257–268, 270–272, 274, 276–281, 284, 286, 288, 291, 294–297, 301–304, 307, 316, 321, 323, 325, 328, 329, 331, 332
UIX	User Interaction. 19, 275, 276, 305
UML	Unified Modeling Language, OMG standard. 45, 57, 58, 60, 62, 67, 69, 100, 133, 153, 326
VCS	Version Control System. 159
VUE-D	Visual UI Element Detector. 280, 281
W3C	World Wide Web Consortium. 2, 115, 127, 161, 175, 205, 216, 300, 330, 385, 420–424
WASM	WebAssembly, W3C standard. 114, 123, 189, 205–211, 213, 215, 239, 300, 314, 315, 327, 328, 330, 331
WASM-T	WASM Transformator, cf. section 6.4. 208–211, 213, 214, 216, 234, 236–242, 246

Web	World Wide Web. 1–6, 8, 10, 12–15, 19, 22, 23, 26, 28, 29, 33, 37, 39, 41, 42, 52, 53, 56, 58–63, 65, 67, 69, 71–75, 77–84, 86–90, 93, 94, 96–98, 102–107, 113, 114, 116–118, 122–127, 129, 136–138, 142, 151, 152, 159, 161, 163, 175, 185, 188–193, 195–201, 203, 205, 207, 209, 214, 216, 217, 223, 229, 232, 233, 253–255, 258–266, 269, 270, 275, 277, 279, 284, 286–289, 294–297, 300–303, 307–309, 312, 315–317, 319–325, 328–331, 380, 382, 384, 385
WebML	The Web Modeling Language, see also IFML. 42, 78
WMST	Web Migration Support Technology, cf. section 6.4.1. 203–205
WSDL	Web Service Description Language. 51, 58, 60, 62, 80
WUI	Web User Interface. 207, 209, 210, 215, 218, 229, 233, 247, 261–263, 266, 268, 274, 287, 288, 294, 301
XAML	Extensible Application Markup Language. 20, 131
XBI	Cross-Browser Inconsistencies. 261–263
XMI	XML Metadata Interchange, OMG standard. 44
XML	Extensible Markup Language, W3C standard. 12, 44, 84, 89, 163

List of Figures

1.1	Web Migration and Related Topics as Venn Diagram	14
2.1	Research Process for Requirements Analysis	25
2.2	Research Questions and Requirements	31
3.1	Model-Driven Reengineering in the ADM Horseshoe Model	44
3.2	ADM standards in the ADM Horseshoe Model	45
3.3	ReMiP Overview	48
3.4	Concept Map of Relationships between Gaps and Requirements	102
4.1	Relationships between Research Objectives and Gaps	108
4.2	AWSM Solution Overview	109
4.3	AWSM in Context	111
4.4	AWSM:RE Method Overview	112
4.5	AWSM:RM Method Overview	115
4.6	AWSM:CI Method Overview	116
4.7	S2DCS Entry of Migration Situation	120
4.8	Annotation Platform Dashboard	122
4.9	RWMPA View Selection	125
4.10	VUE-D UI Element Detection	126
4.11	SCKM Annotation	135
5.1	AWSM:RE Use Cases	146
5.2	Setup Process	147
5.3	Concept Assignment Process	149
5.4	Management and Usage Process	150

5.5	AWSMAP Architecture	152
5.6	Annotation Platform Editor Screenshot	153
5.7	Annotation Platform Concept View	154
5.8	Annotation Platform Statistics	155
5.9	Annotation Platform Visualizations	156
5.10	AWSMAP Integration Architecture	157
5.11	AWSMAP IDE Integration	158
5.12	TFS Integration: Migration Packages	160
5.13	OWL SCKM Ontology Classes	162
5.14	Crowd-based Concept Assignment compared to Microtasking	166
5.15	Crowdsourcing-based classification process	167
5.16	Crowd Worker View	173
5.17	CSRE Statistics	174
5.18	CSRE Results View	180
5.19	Error Rate and Entropy/normalized Herfindahl measure . .	181
5.20	Classification Time, Explanation Length: 1-dimensional Distributions	183
5.21	Classification Time, Explanation Length	184
6.1	AWSM:RM Prototyping Overview	197
6.2	Web Migration Prototype Screenshot	199
6.3	MockAPI Process Flowchart	201
6.4	Applying MockAPI to a Legacy Desktop User Interface Screenshot	202
6.5	WMST Assessment Process	205
6.6	WebAssembly Overview	206
6.7	Architecture of ReWaMP Prototypes	207
6.8	ReWaMP Process	208
6.9	WASM Transformator Architecture	211
6.10	RWMPA Workflow Start Page explaining Required Steps . .	215
6.11	RWMPA Step Worker Page Example: Merge View	217

6.12	Layout Transformation: Mapping Pixel-based to Grid Layouts	218
6.13	UI Transformation Process	220
6.14	UI Transformer as model-driven reengineering process . . .	224
6.15	Layout Transformation Sample	229
6.16	Test Dataset B_T Views	235
6.17	Time distributions per task and test run	237
6.18	Effort Distributions	238
6.19	ReWaMP Questionnaire Results	239
6.20	RWMPA Questionnaire Results	245
6.21	UI Transformer Questionnaire Results	250
7.1	Visual UI Similarity Analysis Process	267
7.2	Calibration Process	269
7.3	Layout Variations	272
7.4	Orientation, Order and Density Changes	273
7.5	User Interface Concept Map	277
7.6	VUE-D Analysis Process	280
7.7	Visually Segmented Screenshot from VUE-D	281
7.8	VUE-D Architecture	282
7.9	Similarity measures and perceived similarity scatterplot . .	291
7.10	Residual Plots	293
7.11	Calibrated Similarity Measures and Residuals QQ plots . .	294
8.1	Solution Picker View	313
8.2	Calendar View of Web Migration Prototype	314
8.3	Web Version of Calendar User Interface	316
8.4	ZMS Legacy and Web User Interfaces	317
B.1	Hierarchical Representation of Problem Domain as LFA Problem Tree	338
C.1	S2DCS Results View	339

C.2	SCKM Formalism	340
E.1	ReWaMP C++ parser classes	349
E.2	RWMPA Process	351
E.3	RWMPA Architecture	352

List of Tables

2.1	Legacy Software Characteristics & Scenario Instantiation	21
2.2	Methods used in Research phases	27
2.3	Scope and Stakeholder Requirements	30
3.1	Usage of ADM standards in Web Migration methods	47
3.3	Overview of assessed web migration approaches	51
3.5	SMART Evaluation	54
3.6	SAPIENSA Evaluation	56
3.7	serviciFi Evaluation	57
3.8	Marchetto2008 Evaluation	59
3.9	SOAMIG Evaluation	61
3.10	Gaps2Ws Evaluation	62
3.11	PRECISO Evaluation	63
3.12	AWS Migration Evaluation	66
3.13	REMICS Evaluation	68
3.14	ARTIST Evaluation	70
3.15	CloudMIG Evaluation	71
3.16	IC4 Evaluation	73
3.17	AMS Evaluation	74
3.18	NCHC Evaluation	76
3.19	L2CMH Evaluation	77
3.20	MIGRARIA Evaluation	79
3.21	MigraSOA Evaluation	81
3.22	MELIS Evaluation	83
3.23	TUIMigrate Evaluation	85

3.24	M&S SW Evaluation	86
3.25	UWA/UWAT+ Evaluation	88
3.26	CeLLEST Evaluation	89
3.27	DAS Evaluation	90
3.28	Evaluation Overview	91
4.1	How-Might-We-Questions for Research Objective Ideation .	106
4.2	Mapping of AWSM Methods to ReMiP	111
5.1	CSRE Experimental Results	177
5.2	CSRE Experiment Descriptive Statistics	179
6.1	WMST Groups and Implementations	204
6.2	ReWaMP tasks and realization in RWMPA workflow	214
6.3	ReWaMP Time Measurement Statistics	237
6.4	Effort Measurement Statistics	238
6.5	RWMPA Time Measurement Statistics	244
6.6	UITransformer Performance Evaluation	249
6.7	UITransformer Empirical Evaluation	251
7.1	Amount of UI Elements in WUI per hierarchy levels	287
7.2	UI Differences and distances	289
7.3	Computed <i>sim</i> measures, empirical Difficulty, Like, \mathfrak{S}	290
7.4	Calibrated <i>sim</i> measures	292
8.1	AWSM Evaluation	305
8.2	Evaluation of AWSM in Comparison to State of the Art	306
A.1	ZMS Scenario Application Quantitative Analysis Results . .	333
B.1	Stakeholder Map	335
E.1	ReWaMP Assumptions	350

F.1	ReWaMP Evaluation Guidelines	353
F.2	ReWaMP Full Empirical Evaluation Data	356
F.3	ReWaMP Full Time Evaluation Data	358
F.4	ReWaMP Full Effort Evaluation Data	358
G.1	RWMPA Evaluation Guidelines	359
G.2	RWMPA Full Empirical Evaluation Data	363
G.3	RWMPA Full Empirical Evaluation Data (cont.)	363
G.4	RWMPA Full Time Evaluation Data	364
G.5	RWMPA Full Time Evaluation Data (cont.)	364
H.1	UITransformer Performance Evaluation Data	367
H.2	UITransformer Performance Evaluation Data (cont.)	370
H.3	UI Transformer Empirical Evaluation Data	373
I.1	AWSM:CI Full Empirical Evaluation Data	378
I.2	AWSM:CI Full Empirical Evaluation Data (cont.)	378

List of Listings

5.1 Partial SPARQL Containment Query	163
5.2 Partial SPARQL Containment Query using sckm:within	164
5.3 SWRL Rules for sckm:influences	164
6.1 Example of Code Generated by ReWaMP to Handle Passing of Return Value from Function Call	213
6.2 Example of a User Interface Description in MFC Resource File (shortened)	225
6.3 Layout Transformation	226
6.4 Mutation and Crossover	227
6.5 Layout Generation	230
7.1 VUE-D Detection Results (shortened) represented as Pascal VOC model	282
8.1 Example of C + + Code Added by ReWaMP to Export Legacy Functionality to JavaScript	315
8.2 JavaScript Binding EventHandler of Button to Exported Legacy Functionality	315
D.1 SCKM Ontology Extract	341
D.2 SCKM Ontology SWRL Rules	345

List of Algorithms

5.1	CSRE Anonymization Algorithm	171
6.1	ReWaMP Process Algorithm	212

Doctoral Dissertations in Web Engineering and Web Science

- (1) Heinrich, Matthias (2014)
Enriching Web Applications Efficiently with Real-Time Collaboration Capabilities
ISBN 978-3-941003-25-9
Volltext: <https://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-149948>
- (2) Speicher, Maximilian (2016)
Search Interaction Optimization: A Human-Centered Design Approach
ISBN 978-3-944640-99-0
Volltext: <https://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-208102>
- (3) Wild, Stefan (2017)
Enhancing Security in Managing Personal Data by Web Systems
ISBN 978-3-96100-010-4
Volltext: <https://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-217284>
- (4) Tschudnowsky, Alexey (2017)
End-User Development of Web-based Decision Support Systems
ISBN 978-3-96100-014-2
Volltext: <https://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-219823>
- (5) Heil, Sebastian (2021)
Web Migration Revisited: Addressing Effort and Risk Concerns
ISBN 978-3-96100-125-5
Volltext: <https://nbn-resolving.org/urn:nbn:de:bsz:ch1-qucosa2-723455>

