TITLEPAGES.TEX

Inscription

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis

odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Acknowledgment

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis

odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

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:

latex Reihenfolge Namen

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, Hua Wang, et al. Vol. 10041 LNCS. Cham: Springer International Publishing. ISBN: 9783319487397.

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 - Science and Technology Publications, pp. 147–158. ISBN: 978-989-758-300-1.

Heil, Sebastian and Martin Gaedke (2016). "AWSM - Agile Web Migration for SMEs". In: *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*. Enase. Rome, Italy: SCITEPRESS - Science, pp. 189–194. ISBN: 978-989-758-189-2.

– (2017). "Web Migration - A Survey Considering the SME Perspective". In: *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering*. SCITEPRESS - Science and Technology Publications, pp. 255–262. ISBN: 978-989-758-250-9.

Heil, Sebastian, Valentin Siegert, and Martin Gaedke (2018). "ReWaMP : Rapid Web Migration Prototyping leveraging WebAssembly". In: *Proceedings of 18th International Conference on Web Engineering (ICWE2018)*. Vol. 10845 LNCS. Caceres, Spain, pp. 84–92. ISBN: 9783319916613.

– (2019). "Crowdsourced Reverse Engineering: Experiences in Applying Crowdsourcing to Concept Assignment". In: *To appear in: Evaluation of Novel Approaches to Software Engineering, Revised Selected Papers, Communications in Computer and Information Science*. Springer.

Bakaev, Maxim, Sebastian Heil, Vladimir Khvorostov, and Martin Gaedke (2018). "HCI Vision for Automated Analysis and Mining of Web User Interfaces". In:

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 10845 LNCS, pp. 136–144. ISBN: 9783319916613.

Bakaev, Maxim, Sebastian Heil, Vladimir Khvorostov, and Martin Gaedke (2019). "Auto-Extraction and Integration of Metrics for Web User Interfaces". In: *Journal of Web Engineering* 17.6, pp. 561–590. ISSN: 1540-9589.

Bakaev, Maxim, Sebastian Heil, Nikita Perminov, and Martin Gaedke (2019). "Integration Platform for Metric-Based Analysis of Web User Interfaces". In: *To appear in: Proceedings of 19th International Conference on Web Engineering*. Daejeon, Korea: Springer.

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. ISBN: 978-1-5386-3488-2.

– (2017b). "Web Intelligence Linked Open Data for Website Design Reuse". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10360 LNCS, pp. 370–377. ISBN: 9783319601304.

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. ISBN: 978-1-5386-7054-5.

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*. 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, ICWE 2014*. Ed. by S. Casteleyn, G. Rossi, and M. Winckler. Vol. 8541. Toulouse, France: Springer, Cham, pp. 494–497. ISBN: 978-3-319-08244-8.

Contents

1 Introduction

The topic of this thesis is the application of scientific approaches to lower the initial hurdle for Small and Medium-sized Enterprises (SMEs) to move 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 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. *Web Migration* is a particular type of Software Migration - the process of moving an 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 will 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 World Wide Web Consortium (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). 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). Technologies for dynamic behavior on the client and server side led to the introduction of the term *Web Application*, formerly also called Rich Internet Applications (RIAs), indicating that their user experience became similar to that of *desktop applications*[1] (Kienle and Distante, 2014). Increased usage of dynamic content allowed users to use the Web as mass collaboration system in the era referred to as Web 2.0 in the mid-2000s (O'Reilly, 2007).

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). Of the related service models Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS), the *SaaS paradigm* not only influenced the architecture of Web Applications, but also also impacted end users' perception and expectations of software (Politis, 2017; 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).

[1] cf. RIA features defined in (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012)

High popularity and use of mobile devices (Statista, 2018b) in private and work contexts has changed end users' expectations with regard to high levels of interactivity and social interactions (Bitkom, 2013). This influenced Web Applications in two ways: the need for open Application Programming Interfaces (APIs) for integration with third-party systems and the requirement of flexible presentation layers capable of handling the vast amount of different devices giving rise to responsive design frameworks like Bootstrap[2].

The above evolution of Web Systems has brought various advantages over traditional desktop applications, 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 companies 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, as opposed to the platform-wise toolchains (e.g., compilers, IDEs) of desktop applications. *Open Web Standards* like HTTP, HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript form the core of Web Applications. They are standardized, published and continuously updated by international standardization bodies like IEEE and W3C and can be publicly accessed and used without restrictions. Therefore, the risk of technology deprecation is significantly lower than for proprietary desktop application technologies. Development staff needs only be trained on one technological base and can be hired from a potentially more extensive group of developers with experience in Web technologies[3].

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: directly, if the locus of code execution is on the server side, or indirectly, if the locus of code execution is on the client side and the code is distributed via the server. The ability to instantly deploy new versions is of great advantage for companies 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).

From the end users' perspective, Web Applications impose *low requirements for accessing software* compared to desktop applications. They can be used from any

[2] https://getbootstrap.com/
[3] cf. https://insights.stackoverflow.com/survey/2019

system with a Web browser without installation, reducing time and complexity for users. The browser has become a *standard interface* (Aversano et al., 2001) / *standard client* (Gitzel et al., 2007). The resulting high *application portability* (Gitzel et al., 2007) fosters a ubiquity of software that allows for new work patterns (Bitkom, 2013) (cf. *SaaS-Powered Workplace*: 38% of companies almost entirely running on SaaS as of 2017 (Politis, 2017)). The lower complexity and effort gives software vendors the chance to address potentially larger users 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). These are advantageous for companies through higher levels of user engagement (Bressler and Grantham, 2000). Social information sharing is advantageous also in working 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). A survey of 50 Italian companies identified the Web as the main target of migration activities (Torchiano et al., 2008). From a business perspective, cloud computing/SaaS has seen the most interest (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. For instance, online banking is well-established for internet users in Germany (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).

Legacy Systems and legacy modernization have been an interest for research for a long time. Several definitions exist in literature. An overview can be found in (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 company due to the vast amount of *knowledge* about business processes, rules, etc. (Aversano et al., 2001; Harry M. Sneed et al., 2010a; Wagner, 2014) resulting from high investments (Lucia, Penta, et al., 2009) and generate a lot of revenue (considered as "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* and evaluation (Bisbal et al., 1999). Legacy Systems are *poorly documented* or have a complete lack of documentation (Harry M. Sneed et al., 2010a; Ian Warren, 2012; Batlajery et al., 2014; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008). There is a *lack of experienced workforce* in the company which developed the system (Batlajery et al., 2014) due to changes or retirement of the original staff (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 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).

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).

Closely related to the topic of Legacy Systems, *Technical Debt* attracts increasing attention in academia and industry (Z. Li et al., 2015; Yli-Huumo et al., 2016). The Technical Debt metaphor (Cunningham, 1992) describes the negative impacts of Legacy System characteristics in the long term in analogy to financial debts:

Definition 4 Technical Debt (Avgeriou et al., 2016)

> 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.

Studies over large repositories of software from different domains determine the average Technical Debt per line of code at 3.61 USD, meaning that larger systems with more than 300KLOC accrue more than 1 million USD of Technical Debt. (Sappidi et al., 2011) Less conservative estimates on the same repository reach more than 15 USD per line of code (Curtis, Sappidi, et al., 2012). Technical Debt results from intentional and unintentional *violations* of good architectural and coding practices which are likely to cause operational problems like outages, security breaches or which contribute to high costs of ownership, e.g. when implementing changes (Curtis, Sappidi, et al., 2012). It has two components, the cost of fixing these violations - called *principal* - and the continuing costs of leaving violations unfixed - called *interest*. In Legacy Systems, the old technology still in use can be seen as Technical Debt (Yli-Huumo et al., 2016). While Technical Debt is not the focus of this thesis, the metaphor characterizes Legacy Systems and helps to understand better the motivation to modernize legacy software.

1.3 Problem Statement

Making a transition from non-web Legacy Systems to web-based systems is desirable for companies 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, Tortora, De Lucia, et al., 2008).

Commencing a Web Migration is hard, however. The above reports on *Software Migration* indicate that companies are still struggling to modernize their Legacy Systems and there are still many non-web Legacy Systems in existence. Both the business and technical perspective make modernization difficult (Khadka, Batlajery, et al., 2014). *Effort* and *Risk* (Khadka, Batlajery, et al., 2014; Canfora et al., 2000; Bisbal et al., 1999; Heil, Siegert, et al., 2018) are rendering companies hesitant to commence a Web Migration.

Effort. Effort (ISO/IEEE, 2017b) 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., 2010a; 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 one project (112 KLOC, medium-sized, Visual Basic), Bernhart et al. (2012) 18+ PM (250 KLOC COBOL), Aversano et al. (2001) 8PM (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, Scanniello, Tortora, and Vitiello, 2006; Brodie and Stonebraker, 1995). *Resistance from within the organization* (Harry M Sneed et al., 2010b) is another factor influencing effort, with the unwillingness of staff to share knowledge and resistance towards change in general and new technologies, in particular, requiring good communication of the necessity and benefits of modernization (Khadka, Batlajery, et al., 2014). According to Gartner, "Cultural issues are at the root of many failed business transformations" (Gartner, 2016).

Risk. Modernizing Legacy Systems involves high *risk* (ISO/IEEE, 2017b; Khadka, Batlajery, et al., 2014; Canfora et al., 2000; Bisbal et al., 1999; Seacord et al., 2003; Heil, Siegert, et al., 2018). Uncertainty of success makes companies 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., 2010a; 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., 2010a; 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 to commence 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 companies 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 many companies struggling to commence Web Migration.

1.4 Scope

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

Research Question RQ1

How to support software companies to commence Web Migration?

Research Question RQ2

How to support commencing Web Migration with limited effort?

Research Question RQ3

How to address concerns about risk when commencing Web Migration?

The guiding *central research question* (Creswell, 2014) RQ1 and the two specific detailed research questions RQ2 and RQ3 were used for ideation using the *How Might We* method from the Human-Centered Design (HCD) design kit (IDEO, 2015) as described in detail in section section 4.1.1 The re-formulation of RQ1 into "*How might we support companies to commence Web Migration?*" has helped us create a solution consisting of the three core contributions as outlined in the following. The three research questions above 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., 2010a), which is valid for Web Migration. Instead, reengineering (ISO/IEEE, 2017b; Software Engineering Standards Committee of the and 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 XML (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 (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 can be found in (Z. Li et al., 2015).

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

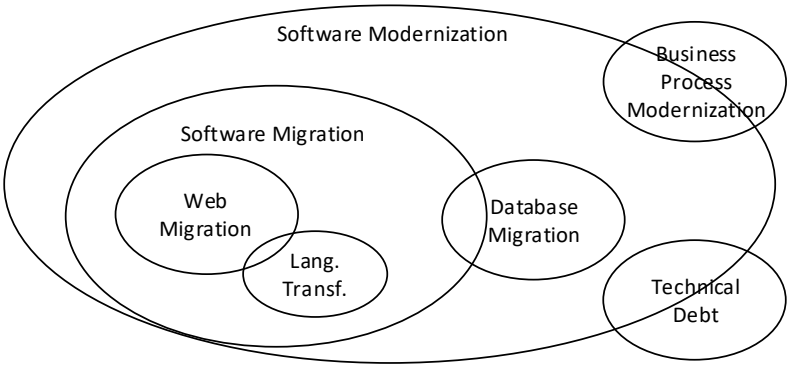


Figure 1.1.: Web Migration and Related Topics as Venn Diagram

1.5 Contributions

To answer RQ1, we define an approach for lowering the initial hurdle for commencing a Web Migration project by providing processes, models, and tools that address the initial fears and resistance outlined above. This approach is Agile Web Migration for SMEs (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 measuring visible changes.

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 it elicits six requirements based on 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 discussed. Chapter 4 reports on the design method and research process of the solution presented in this thesis. Based on systematic, detailed problem analysis, the solution and research design are derived, and an overview of the principles, formalisms, and methods of the AWSM Methodology and tools of the AWSM Platform is given. The following 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 perceptual empirical similarity between legacy and Web user interface tailored to the characteristics of the target user group to control 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 companies 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 software companies to commence Web Migration?", was derived and the scope of this thesis was defined. 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 RQ1.

2 Requirements Analysis

In this chapter, we will further investigate the problem raised in the introduction - concerns about effort and risk that render companies hesitant to commence a Web Migration due to insufficient support in initial phases - by outlining the research context in an industrial 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 definition of requirements in section 2.2 and the detailed problem analysis of the Web Migration approach 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 an SME-sized Independent Software Vendor (ISV)[4] 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

[4] https://www.gartner.com/it-glossary/isv-independent-software-vendor/

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. A previous migration attempt towards a client/server architecture failed due to the complexity and lack of sufficient resources.

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. 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*[5] 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, 168k 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 XAML (122 files, 17.8k SLOC). The main technologies are Visual C++ as programming language, Microsoft Foundation Class (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, in some cases even stand-alone components 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 large[7] Legacy System.

Within x.concept, *ZMS* is an isolated stand-alone component which 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 and 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

[5] https://github.com/AlDanial/cloc
[6] http://msdn.microsoft.com/de-de/library/d06h2x6e.aspx
[7] only 12% of the applications in the Appmarq repository — representing 1.03 billion LOC of 1850 applications — are larger than 1M LOC (Curtis, Muller, et al., 2017)

Abstract Characteristics	Instantiation in ZMS scenario application
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 their 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 company and initiating this process is difficult as outlined in section 1.3 and further analyzed in section 4.1.2.

2.2 Requirements

In this section, requirements for assessing the state of the art are elicited based on the situation and motivation outlined in the introduction and the scenario. 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 existing literature and analysis of the scenario.

To indicate different requirements levels, we follow the definitions of RFC 2119 (IETF, 1997), using

- **MUST** for absolute requirements fulfillment of which is mandatory, and
- **SHOULD** for requirements fulfillment of which is recommended.

Table 2.2 provides an overview of all scope and stakeholder requirements and indicates their significance according to RFC 2119 levels.

Table 2.2.: 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	Re-use of Legacy Assets	SHOULD
C3 Exp	Expertise & Tool Support	SHOULD
C4 Agile	Agile Development Process Integration	SHOULD

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

2.2.1 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.

S1 Initial Phase Support

Companies 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 a software developing company

to overcome the initial resistance. According to the ReMiP (Harry M Sneed et al., 2010b), 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*[8] 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., 2010b). A suitable approach should therefore additionally provide appropriate methods and tools to supply artifacts which can be used as a basis for communicating the necessity and benefits of migration and supporting the decision making.

The satisfaction criterion of this requirement 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.

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

[8] "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)

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 this 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.

2.2.2 Stakeholder Requirements

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

C1 Risk Management

As outlined section 1.3, any Software Modernization involves a high risk, which makes companies hesitant to commence this process (Khadka, Batlajery, et al., 2014; Canfora 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 cut over strategies, called "Big Bang" (Bisbal et al., 1999), have a high risk of failure (Harry M. Sneed et al., 2010a; 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., 2010a) - 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., 2010a). *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 document 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). Any modernization bears the risk of losing valuable tacit knowledge about business processes, rules, etc. (Aversano et al., 2001; Distante, Scott Tilley, and Canfora, 2006; Harry M. Sneed et al., 2010a; 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). A *risk-managed modernization* strategy (Seacord et al., 2003) therefore should provide methods to re-discover, secure and manage this knowledge to make it useable for subsequent modernization activities.

The satisfaction criterion of this requirement is the risk management of the Web Migration, which means that at least one basic risk management practice beyond 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 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 Re-use of legacy assets

TODO:RENAME_continuity_of_functionality_and_UIX?

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., 2010a; 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. Often, 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, Tortora, De Lucia, et al., 2008;

Distante, Perrone, et al., 2002). Existing *legacy artifacts*[9] of an ISV comprise the legacy source code and the running Legacy System itself, whereas *assets*[10] like documentation, requirements or models originally used for production are typically missing. (Wagner, 2014; Bisbal et al., 1999; Harry M. Sneed et al., 2010a; Ian Warren, 2012; Batlajery et al., 2014; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 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, Tortora, De Lucia, et al., 2008). A suitable approach must promote rediscovery, 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., 2010a).

The satisfaction criterion of this requirement is the degree of re-use 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., 2010a; Seacord et al., 2003). While expertise in the source technology base in companies 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., 2010a; Seacord et al., 2003). For Web Migration, additional Web Engineering expertise is required due to the specifics of the target

[9] In 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)

[10] An 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).

environment (cf. *S2 Web*), but typically not existing in ISVs of traditional non-web desktop applications (Fowley, Elango, et al., 2017). For a company 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 outsourcing[11] 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 this 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 2.1.1. 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 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., 2010b; 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.

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

The satisfaction criterion of this 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.

2.3 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. Six requirements were elicited, the two scope requirements represent the problem identified as scope of this thesis in RQ1, i.e., support for the initial phases of Web Migration and Web Applications as target architecture, and the four stakeholder requirements are derived from the problems of the scenario stakeholder such as 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.

3 State of the Art

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 due to the specific characteristics of the Web as target platform of the migration on the one hand side, knowledge of Legacy Systems, migration strategies, reverse engineering and migration planning due to the specific characteristics of Legacy Systems and different nature of migration activities on the other hand side. Thus, isolated consideration of approaches from either only Web Engineering or Software Migration is not sufficient. Dedicated *Web Engineering approaches* such as WebML/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). 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 (I. Warren and Ransom, 2002) fail to address the specific characteristics of the Web as target platform (requirement S2). The following analysis of migration methods is therefore focused on dedicated Web Migration approaches - as introduced in section 1.4, this means methods that move Legacy Systems to a web-based target environment. The selection of approaches assessed is based on their orientation towards the thesis' central research question RQ1 and the availability of published research results within Web Engineering and Software Migration communities. Suitability of the assessed approaches for supporting companies to commence a Web Migration is evaluated employing the requirements and evaluation scheme introduced in the previous chapter.

3.1 Standards and Reference Models

This section provides a brief introduction to relevant standards and reference models from the fields of Software Modernization and software migration that will help

understand the context, structure, and technologies of concrete Web Migration methods analyzed in the following state of the art analysis.

3.1.1 Architecture-Driven Modernization (ADM)

The Object Management Group (OMG)[12] has formed a task force to address Software Modernization. The Architecture-Driven Modernization Task Force (ADMTF)[13] mission is to promote industry consensus on the modernization of existing applications through standardization. According to ADMTF, Architecture-Driven Modernization (ADM) is the process of understanding and evolving existing software assets for several purposes, among them migration, restructuring, refactoring, translation to other languages and porting. ADM considers modernization in two domains: business and IT and on three levels: business architecture, application/data architecture, technical architecture (Vitaly Khusidman and William Ulrich, 2008). The ADMTF advocates a standards-based model-driven methodology for modernizing existing applications on all three architectural levels. This is an extension of OMG's Model Driven Architecture (MDA) (Object Management Group, 2014) principles to Software Modernization. The resulting model is referred to as the ADM Horseshoe[14] Model (Pérez-Castillo, De Guzmán, et al., 2011; Pérez-Castillo, Guzmán, et al., 2011; Khusidman and Ulrich, 2007) (cf. fig. 3.1), consisting of three stages: reverse engineering, restructuring, forward engineering. As of today (late 2018), the ADM family of standards consists of 12 OMG standards[15]. The following paragraphs will briefly summarize the four most relevant 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.

Knowledge Discovery Metamodel (KDM) (Object Management Group, 2016a) is the first and most important of the ADM standards, around which the other ADM standards are defined (Pérez-Castillo, De Guzmán, et al., 2011). It has been adopted as as ISO/IEC 19506:2012 standard[16] in 2012. 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)[17] and XML Metadata Interchange (XMI)[18], allowing integration and interoperability across Software Modernization tools. The KDM *metamodel*[19] 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),

[12] `https://www.omg.org/`
[13] `https://www.omg.org/adm/`
[14] adapted from the original SEI horseshoe reengineering model (R. Kazman et al., 1998)
[15] cf. `https://www.omg.org/spec/category/software-modernization`
[16] `https://www.iso.org/standard/32625.html`
[17] `https://www.omg.org/spec/MOF/`
[18] `https://www.omg.org/spec/XMI/`
[19] metamodel: logical information model that specifies the modeling elements used within another (or the same) modeling notation (ISO/IEEE, 2017b)

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 can be seen as an ontology of Legacy Systems (Pérez-Castillo, De Guzmán, et al., 2011). KDM descriptions describe Legacy Systems at a granularity above procedure level. Throughout this thesis, mappings to KDM will be 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 aims at providing a universal software modeling framework. ASTM allows describing Legacy Systems at a granularity below procedure level by providing a mapping of all code-level programming language statements into low-level software models. To achieve this 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 UML, that allows specifying Platform-Independent Models (PIMs) of ASTs. The Plattform-Specific Models (PSMs) for various programming languages are supported through the Specific Abstract Syntax Tree Metamodel (SASTM) extensions.

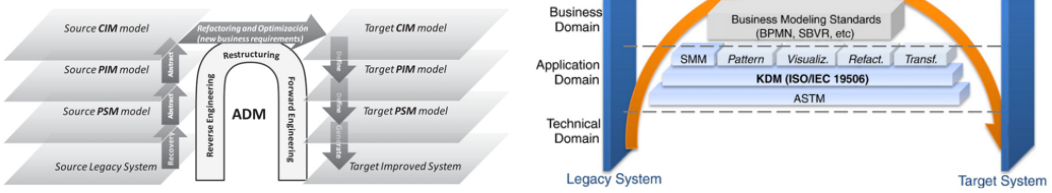
Structured Metrics Metamodel (SMM) (Object Management Group, 2012) specifies a metamodel for representing measurement information related to structured information models, in particular MOF-compliant models. Thus, it is part of the ADM roadmap and addresses the need for metrics in combination with KDM/ASTM. 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).

MOF Query/View/Transformation (QVT) (Object Management Group, 2016b) is an OMG standard addressing model-to-model transformations. Thus, it plays a vital role in the ADM horseshoe connecting 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

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

support bi-directional transformation, whereas QVT-operational are imperative languages for unidirectional transformations. The QVT standard specifies two concrete transformation languages: QVTr and QVTo. QVTr is the QVT Relations language, specifying symmetric relationships between MOF-compliant models and supporting complex object pattern matching in a declarative way. QVTo is the Operational Mappings language defined by QVT and supports transformations through a procedural style of defining mappings from one model to another. The QVT languages employ OCL[20] as expressions language.



(a) Model-Driven Reengineering (Pérez-Castillo, Guzmán, et al., 2011)

(b) ADM standards (Pérez-Castillo, De Guzmán, et al., 2011)

Figure 3.1.: ADM Horseshoe

3.1.2 ReMiP

Sneed et al. introduce a reference model for software migration called Reference Migration Process (ReMiP) (Harry M Sneed et al., 2010b; Gipp and Winter, 2007). It aims at providing a general and adaptable framework for understanding and describing arbitrary software migration processes and is based on a synthesis of existing software evolution and software development methods. 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. 3.2). 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 orga-

[20] https://www.omg.org/spec/OCL/

nizational 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. Each phase is terminated by a milestone which serves as a decision point whether to continue the migration. These four milestones are: project objective & process recommendation, binding master plan, migrated system release and final documentation. 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. The base disciplines are: *configuration & change management*, *project management*, *staff qualification* and *migration environment*. The holistic view of the ReMiP reference model provides a framework for understanding the context of software migration and thus Web Migration. The definition of requirement S2 refers to the ReMiP phases. In section 4.3.3.4 we provide mappings of the solutions provided in this thesis to ReMiP phases and disciplines.

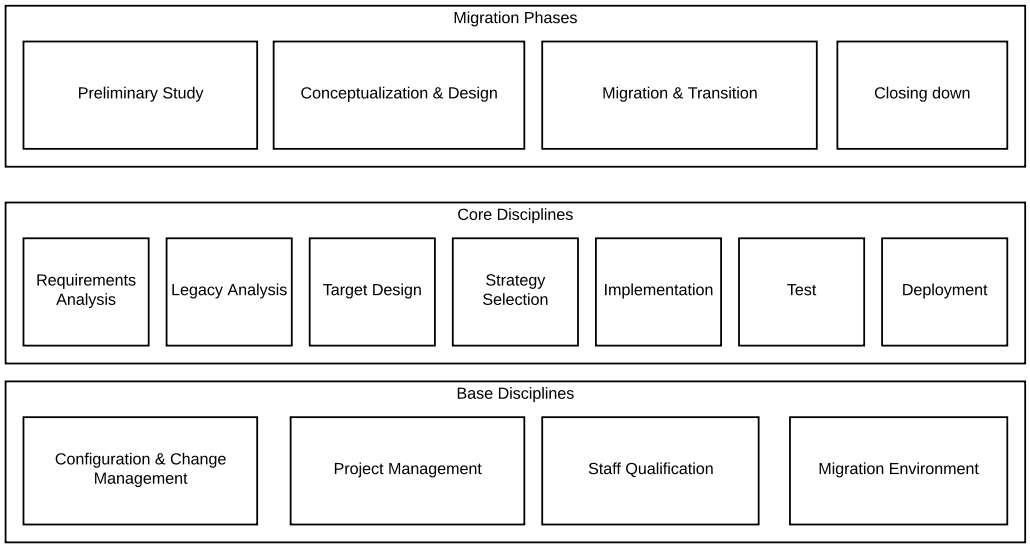


Figure 3.2.: ReMiP Overview (adapted from Harry M Sneed et al., 2010b)

3.1.3 SOA-MF

Razavian and Lago introduced the SOA Migration Framework (SOA-MF) (Razavian, 2013; Razavian and Lago, 2010c; Razavian and Lago, 2010b; Razavian and Lago, 2010a) based on a systematic literature review (SLR) on SOA migration techniques. SOA migration is a research focus within the Web Migration field which has received significant interest (cf. section 3.2.1). SOA-MF considers migration of Legacy Systems

to SOA as a reengineering problem and presents an extended version of SEI's reengineering horseshoe model (R. Kazman et al., 1998), with the three sub-processes reverse engineering, transformation, and forward engineering. In the conceptual model of SOA-MF (cf. fig. 3.3), migration is considered as a transformation of *artifacts* through *activities* supported by *knowledge*, which takes place at different *levels of abstraction* between code and enterprise level. The reverse engineering sub-process analyses and recovers the legacy architecture into higher-level abstract representations, allowing to identify migration candidates and relevant knowledge in the legacy codebase input artifact. Transformation is conducted on representations within the same level of abstraction, altering design elements, architecture, business models and strategies. Forward engineering renovates the selected parts of the Legacy System according to new requirements into a service-based version through service implementation and service composition. The activities of these three sub-processes are supported by code-related, design element-related, composition and business domain knowledge. SOA-MF describes a reference model for a complete SOA migration process; existing migration methods do not necessarily employ all artifacts, activities and knowledge types. It defines a questionnaire-based method for mapping arbitrary SOA migration approaches to SOA-MF according to coverage. Application of this mapping on existing literature has identified eight distinct families of approaches (Razavian, 2013; Razavian and Lago, 2010a). SOA-MF compliments ReMiP's project management focus as a reference model putting more emphasis on technical and business process aspects of migration in ReMiP phases two and three.

3.2 Web Migration Approaches

This section surveys the state of the art of Web Migration approaches. It is organized into four groups: SOA, Cloud, Web Systems Evolution and Web Application. These groups are defined by the target environment (SOA, Cloud, Web Application) and source system (Web Systems Evolution) and represent areas of interest in Web Migration research. We identified 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 on the state of the art of Web Migration in 2018, focusing on comprehensive approaches with high *migration discipline coverage* (cf. (Heil and Gaedke, 2017)), multi-publication approaches and larger-scale research projects. The following sections will briefly introduce the four groups and 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 2.2. The

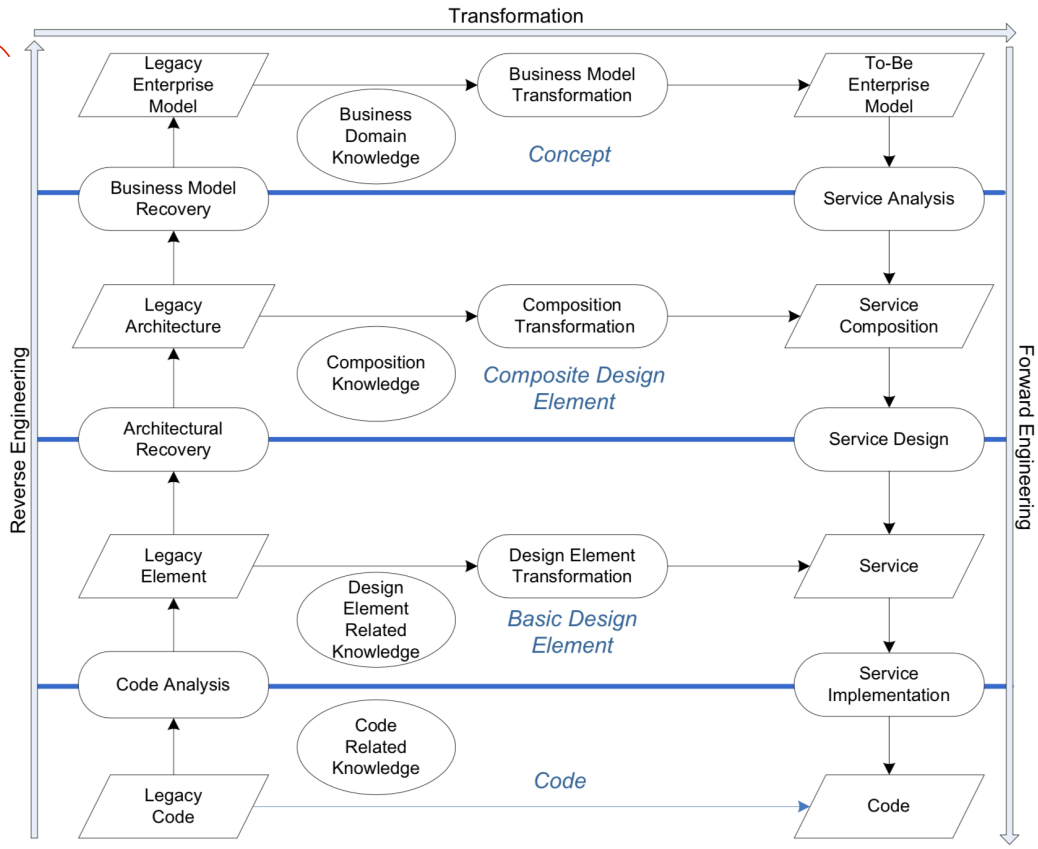


Figure 3.3.: SOA-MF Overview (Razavian and Lago, 2010b)

following symbols represent the ratings according to the three-level assessment scheme introduced in section 2.2: ○: not fulfilled, ◐: partially fulfilled, ●: fulfilled.

3.2.1 Migration to SOA

Service Oriented Architectures are one of the major target environments of Web Migration. SOA is a business-driven approach that abstracts software by decomposition into loosely-coupled services enabling to address rapidly changing business needs through reuse of existing software assets (Gold, Mohan, et al., 2004). A *SOA service* is a "coarse-grained, discoverable, and self- contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model" (G. Lewis, Morris, O'Brien, et al., 2005). While SOA services exist outside the Web environment, the "most common form of SOA implementation is that of *web services*" (G. Lewis, Morris, Smith, et al., 2008), typically with a WSDL service interface description and SOAP over HTTP communication. Therefore, a significant share of Web Migration research is on SOA migration, 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). It is worth noting that SOA migration research focuses on target systems that adhere to the fine-grained SOA notion of *microservices* - i.e. one application composed of a set of small services - than on

integrating several applications through services to build federated applications. However, the term microservices was coined much later, and since it is commonly used in literature, we employ the term *SOA migration* to describe the entire range of migration towards Web services in both the federated SOA and microservices flavor. For more details on SOA migration refer to the dedicated surveys (Khadka, Saeidi, et al., 2013; Razavian and Lago, 2011; Almonaies et al., 2010).

3.2.1.1. 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 version[21], 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

[21] based on the 2013 SMART materials available at `https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508038`

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. It is designed for migration to SOA-based Web Systems, web-based user interfaces are not considered. Risk management is prominently addressed, including basic techniques like feasibility assessment, incremental activities updating artifacts and incremental process as well as migration pilots. 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. Expertise requirements for SMART are not high due to the straightforward, 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.

Table 3.2.: SMART Evaluation

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

3.2.1.2. 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 project[22] 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-CUBE[23]. The SAPIENSA method aims at exploiting architectural knowledge to enable the transformation to services on different SOA abstraction levels[24]. 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 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:

[22] https://www.nwo.nl/en/research-and-results/research-projects/i/19/4019.html
[23] https://cordis.europa.eu/project/reference/215483
[24] cf. to abstraction levels in SOA-MF

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. The target system is a service-based Web System; migration of GUI is not considered. No risk management is proposed. Reuse of legacy assets is employed to retain functionality, but user interaction is not considered. Since the SAPIENSA process is not bound to sophisticated transformation methods or technologies, expertise requirements are not high, but no supporting tools are provided. Integration into ongoing development is not addressed. SAPIENSA contributed to systematizing research on SOA migration by producing the previously introduced conceptual SOA migration reference framework SOA-MF which has contributed to the understanding of migration processes and knowledge in Legacy Systems of this thesis as described in section 4.3.2.

Table 3.3.: SAPIENSA Evaluation

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

3.2.1.3. ServiciFi

The ServiciFi method for SOA Migration presented by Khadka, Reijnders, et al. (2011) and Khadka (2016) is part of the ServiciFi project[25] 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. Portfolio analysis is used to provide a cost-benefit overview and comprises preliminary return on investment in terms of estimated maintenance

[25] https://servicifi.wordpress.com

cost and business value in relation to market needs. The main part of the process is iterative. C1 is fulfilled. While legacy artifacts like documentation, or UML diagrams are analyzed if existing, there is no systematic recovery and re-use 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, Harman, 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 3.4.: serviciFi Evaluation

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

3.2.1.4. Marchetto2008[26]

The approach presented by Marchetto and Ricca (2008) aims at migrating legacy Java GUI desktop applications based on the Model-View-Controller (MVC) pattern[27] 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. Migration to SOA is achieved by replacing method invocations by service invocations in the existing Java desktop GUI. This process is incremental, successively identifying and migrating more services. Knowledge re-discovery is limited to functionality, that is then codified in use case diagrams. The authors indicate the determination of the underlying business process

[26] this identifier was assigned to the approach due to lack of a name by the authors
[27] `http://wiki.c2.com/?ModelViewController`

but without a specified methodology or codification e.g. as BPMN[28] 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 backend. This is achieved through a semi-automatic wrapper approach, employing the Apache Axis2[29] 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. Since the resulting application 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. Integration with ongoing development processes is not considered.

Table 3.5.: Marchetto2008 Evaluation

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

3.2.1.5. SOAMIG

SOAMIG (Fuhr et al., 2013; Winter et al., 2011; Zillmann et al., 2011) is a research project[30] 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, supporting 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

[28] https://www.omg.org/spec/BPMN/
[29] http://axis.apache.org/axis2/java/core
[30] 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

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. The target system is a service-oriented Web System, but SOAMIG does not address web-based user interfaces. Basic risk management methods like technical feasibility study and an iterative process model are present, but the business case is not addressed, extracted knowledge is technically required for the transformation. Reuse focuses on functionality; user interaction is not addressed. Expertise requirements are high due to the complex model-driven methods, tools and querying and transformation languages used. The stand-alone process does not address integration.

Table 3.6.: SOAMIG Evaluation

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

3.2.1.6. 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 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 technical GUI-based wrapper solution without process. Thus phases prior to migration are not addressed. The target system is a service-based Web System without GUI. Risk management and knowledge recovery are not addressed.

Re-use is high for legacy functionality due to the encapsulation approach. Expertise requirements are low since the only required human interaction is simple drag-and-drop service definition supported by a visual programming tool. Integration into ongoing development is not considered.

Table 3.7.: Gaps2Ws Evaluation

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

3.2.1.7. PRECISO

The PRECISO SOA migration method (Pérez-Castillo, García-Rodríguez de Guzmán, et al., 2013; Ricardo Perez-Castillo 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 meta-model. 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 meta-model for the Web service PSM and the Web service implementation is based on SOAP.

PRECISO is a technical SOAP-based database wrapper migration solution, no phases prior to migration are addressed. The target system is a service-based Web System without GUI. Risk management is not addressed; knowledge recovery considers only the data model. Re-use 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. Expertise requirements are low due to the limited scope and high automation achieved in the PRECISO tool. Integration into ongoing development is not considered.

Table 3.8.: 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)

3.2.2.1. AWS Migration

Amazon provides a set of resources[31] 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, 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 tools[32] 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

[31] https://aws.amazon.com/cloud-migration/
[32] https://aws.amazon.com/application-discovery/

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. While it can be used to migrate to SaaS, the AWS migration method addresses more general cloud migration and does therefore not fulfill the target environment Web requirement. 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. No specific re-use 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 further is based on dedicated cloud teams forming a migration factory[33] within the company, targeting large scale companies with appropriate Human Resources and support through external partners from the AWS Migration Acceleration Program[34] (AWS MAP) and AWS Professional Services[35]. Tool support, on the other hand, is extensive and bundled in the AWS Migration Hub[36]. Integration into ongoing development is not addressed.

Table 3.9.: AWS Migration Evaluation

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

3.2.2.2. 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; Sadovykh et al., 2011; Mohagheghi, Berre, et al., 2010) is an EU FP7 project[37] 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

[33] cf. reengineering factory (Borchers, 1996) idea that migration can be factory-like product line way through standardized processes and organization
[34] https://aws.amazon.com/migration-acceleration-program/
[35] https://aws.amazon.com/professional-services/
[36] https://aws.amazon.com/de/migration-hub/
[37] http://www.remics.eu

and verification in the cloud environment. The REMICS modernization method adheres to the ADM horseshoe model (Pérez-Castillo, De Guzmán, et al., 2011; Pérez-Castillo, 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 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. 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. Re-use of functionality is addressed in detail through the SOA migration parts of REMICS, but maintaining user interaction is not part of REMICS. Expertise requirements are high due to the plethora of model-driven techniques and tools employed. 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 this requirement

Table 3.10.: REMICS Evaluation

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

3.2.2.3. ARTIST

ARTIST (Advanced software-based seRvice provisioning and migraTIon of legacy Software) (Brunelière, Cabot, Izquierdo, et al., 2015; Menychtas, Konstanteli, et al.,

2014; Brunelière, Cabot, Dupé, et al., 2014; Menychtas, Santzaridou, et al., 2013; Orue-Echeverria et al., 2014; Konstanteli et al., 2015; Brunelière, Cánovas, et al., 2013) is an EU FP7 research project[38] that aims at providing a comprehensive, tailorable 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.

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 MoDisco[39], 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, however, without tangible means of communication. The target architecture is a cloud-based SaaS system; a web-based GUI is not addressed. Risk management is part of the methodology in terms of technical and economic feasibility assessment, business goals conformance checking and SLA compliance verification, extensive model-driven knowledge recovery is also specified. The model-driven transformation process addresses re-use of functionality, but maintaining user interaction is not part of ARTIST. Expertise requirements are high due to the wide range of model-driven reverse engineering and transformation techniques and tools employed. ARTIST is specified as a stand-alone migration, so integration in ongoing development processes is not considered.

Table 3.11.: ARTIST Evaluation

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

[38] `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/`
[39] part of the preceding European FP6 MODELPLEX project, cf. `https://www.eclipse.org/MoDisco/`

3.2.2.4. 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 PaaS/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 Kieker[40] software monitoring tool. The utilization model represents resource usage and performance characteristics using OMG's Software Metrics Metamodel (SMM)[41]. Target cloud environment selection is enabled through modeling available cloud provider IaaS and PaaS offerings. A subsequent 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 tool[42].

CloudMIG is a cloud migration method that does not address phases prior to migration apart from knowledge recovery. The target system is a cloud-based SaaS application which implies a web-based UI. However, this is not addressed explicitly. Risk management is not considered. Re-use is high for legacy functionality, but user interaction is not regarded. Expertise requirements are high due to the complex MDRE and transformation techniques. Integration into ongoing development is not considered.

Table 3.12.: CloudMIG Evaluation

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

[40] http://kieker-monitoring.net/
[41] https://www.omg.org/spec/SMM/
[42] https://sourceforge.net/projects/cloudmigxpress/

3.2.2.5. 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. Targeting PaaS-supported SaaS applications, a web-based UI is assumed although not explicitly mentioned. 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. Due to the coarse-grain backend component focus, re-use is high for legacy functionality, but user interaction is not regarded. Expertise requirements are high due to the required manual reengineering and lack of tool support. Integration into ongoing development is not considered.

Table 3.13.: IC4 Evaluation

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

3.2.2.6. 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)[43] 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 application-specific desktop virtualization in the browser for multiple users using Sandboxie[44] as a virtualization environment to run several instances of the legacy application simultaneously.

AMS is a technical GUI-based encapsulation solution without process. Thus phases prior to migration are not addressed. The target system is a Web System with a web-based UI mirroring the legacy UI, but not a Web Application. Risk management and knowledge recovery are not addressed. Re-use 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. Expertise requirements are low since the technical focus of AMS does not require manual action. Integration into ongoing development is not considered.

Table 3.14.: AMS Evaluation

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

3.2.2.7. NCHC

Wang and Chang (2014) from Taiwan's National Center for High Performance Computing (NCHC)[45] propose a cloud migration approach for desktop applications based on Apache Guacamole[46] technology. It 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-

[43] https://docs.microsoft.com/en-us/windows/desktop/com/component-object-model--com--portal
[44] https://www.sandboxie.com
[45] http://www.nchc.org.tw/
[46] http://guacamole.apache.org/

based hypervisor through the libvirt 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 technical virtualization solution without process. Thus phases prior to migration 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 etc. Risk management and knowledge recovery are not addressed. Re-use is high both for legacy functionality and user interaction due to the encapsulation approach. Expertise requirements are low since there is no migration process. Integration into ongoing development is not considered.

Table 3.15.: NCHC Evaluation

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

3.2.2.8. 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, 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 SCA[47] 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. The target system is a cloud-based Web System, however, 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. The transformation approach has high re-use for legacy functionality, but user interaction is not

[47] http://oasis-opencsa.org/sca

considered. Expertise requirements are high due to the graph-based modeling and transformation techniques required and lack of concrete tool support. Integration into ongoing development is not considered.

Table 3.16.: 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 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 approaches 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.

3.2.3.1. MIGRARIA

MIGRARIA (Sosa-Sanchez et al., 2014; Sosa et al., 2013; Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Rodríguez-Echeverría, Conejero, Linaje, et al., 2010) is a project[48] that aims at modernization of legacy Web Applications to new Web paradigms: Rich Internet Applications (RIAs) and Service-oriented architecture (SOA). 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 (Sosa-Sanchez et al., 2014; Sosa et al., 2013) propose a different method called MigraSOA and are assessed separately in the following section MigraSOA.

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 WebML[49] to WebML with RIA extensions (Bozzon et al., 2006; Manolescu et al., 2005; Carughi et al., 2009).

[48] `http://www.eweb.unex.es/eweb/migraria/`

[49] the Web Modeling Language, cf. `http://webml.deib.polimi.it/` which has later been standardised by OMG as Interaction Flow Modeling Language (IFML), cf. `https://www.ifml.org`

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 processing and asynchronous communication, later work (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012) formally introduces a more comprehensive list of RIA features. Phases before migration beyond knowledge recovery are not addressed. Both legacy and target system are full Web Applications including web-based GUIs. Risk management 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 WA is stated as a requirement of the MIGRARIA method. 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)[50] or QVT. Integration into ongoing development processes is not addressed. The MIGRARIA method only reports application for small, artificial examples like the Java Pet Store Demo (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012).

Table 3.17.: MIGRARIA Evaluation

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

3.2.3.2. 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 et al., 2017; Sosa-Sanchez 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 et al., 2014). MigraSOA uses the reverse engineering step of the MIGRARIA method to extract an instance of the MIGRARIA MVC meta-model consisting of

[50] cf. https://www.eclipse.org/atl/

data, operations, controlflows with Modisco, services are manually identified in this model, using model-to-model transformations based on ATL rules a Simple-SoaML[51] is derived, and model-to-text transformations using Acceleo[52] 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 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 business processes are then extended to become executable (Sosa Sanchez et al., 2017) by adding the invocation information of the aligned services using Apache Activiti[53]. Phases before migration are not addressed beyond reverse engineering. Being a WSE approach, both the input and the result are Web Applications. Risk management is not addressed; knowledge recovery is conducted only as required for the transformations. Functionality and user interaction are completely re-used since the original Web Application is only internally re-structured to SOA. 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. The proposed process does not address integration into ongoing development activities.

Table 3.18.: 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 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. For more information refer to (Heil and Gaedke, 2017).

[51] https://www.omg.org/spec/SoaML
[52] https://www.eclipse.org/acceleo/
[53] https://www.activiti.org

3.2.4.1. MELIS

Lucia et al. present a migration strategy (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 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. 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. Advanced risk management or recovery of knowledge are not described. Legacy functionality is re-used 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, 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. The process is supported 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 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 3.19.: MELIS Evaluation

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

3.2.4.2. 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. The target system is a Web System including a web-based UI. However, this UI is a representation of a TUI. Risk management is not addressed. Re-use 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. Expertise requirements are low, as the approach is limited to the technical perspective of transforming the UI through providing a transparent middleware. Integration into ongoing development processes 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 3.20.: TUIMigrate Evaluation

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

3.2.4.3. 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. The target system is a Web System including a web-based UI. However, this UI is a representation of a TUI. Risk management is not addressed. Re-use 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. Expertise requirements are low due to the narrow technical focus and near-complete automation. Integration into ongoing development is not considered.

Table 3.21.: M&S SW Evaluation

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

3.2.4.4. 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 Scott Tilley, 2005; Distante, Canfora, et al., 2006; Distante, Scott Tilley, and Canfora, 2006). The focus lies on the definition of a methodological approach based on conceptual user-centered modeling by using the Ubiquitous Web Applications (UWA) framework and its extended transaction design model UWAT+ (Distante and Scott 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 meta-models. 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. The target system is a Web Application including a web-based UI. Risk management is not addressed; knowledge recovery is present. Re-use 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. Expertise requirements are high due to the complexity of UWA/UWAT+ methodology and meta-models and lack of dedicated tools support. Integration into ongoing development is not considered.

Table 3.22.: UWA/UWAT+ Evaluation

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

3.2.4.5. CelLEST

The Canadian CEL[54] Legacy Enhancement Software Technologies (CelLEST) (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[55] 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 CelLEST 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 CelLEST process proposes five steps: execution of the Legacy System in an emulator to collect interaction traces, behavior modeling using state-transition-models, task discovery through pattern analysis, task modeling through information-exchange analysis and Web GUI specifi-

[54] abbreviation for Celcorp, the main industrial sponsor
[55] `http://www.cs.ualberta.ca/~` stroulia/CELLEST

cation based on the identified functionalities. The creation of the state-transition model 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 algorithm (El-Ramly et al., 2002) identifies task execution patterns based on recurring similar state-transition sequences, 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. CelLEST automatically creates an XML specification of a form-based user interface that is executed in runtime interpreters translating to XHTML or WML. 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. Risk management is not addressed; knowledge recovery is present in terms of interaction modeling. Re-use is high both for legacy functionality and user interaction due to the proposed encapsulation approach and explicit modeling of user interactions. 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. Integration into ongoing development is not considered.

Table 3.23.: CelLEST Evaluation

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

3.2.4.6. 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 based on Broadway is proposed which enables translation between GTK+ and HTML5. DAS is a technical virtualization solution without process. Thus phases prior to migration 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 etc. Risk management and knowledge recovery are not addressed. Re-use is high both for legacy functionality and user interaction due to the encapsulation approach.

Expertise requirements are low since there is no migration process. Integration into ongoing development is not considered.

Table 3.24.: DAS Evaluation

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

3.3 Approach Classification

The approaches introduced and assessed above have been assigned into groups according to research areas and target architectures, which represents the historical development of migration research and the classification commonly used in literature (Kienle and Distante, 2014). While some commonalities with regard to the requirements can be identified among the members of these groups, the migration approaches also show unique characteristics that cannot be captured on a per-group basis. Often tailored to a specific context, a high number of parameters determine a specific migration approach (e.g. Migration Type, Migration Discipline, Legacy Environment, Target Environment) (Heil and Gaedke, 2017). These form independent dimensions of description of migration approaches, which are orthogonal to the groups introduced above. In particular, in our Survey (Heil and Gaedke, 2017), we observed that migration approaches are further divided depending on the overall methodology which they use. We identified three main 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; Software Engineering Standards Committee of the and IEEE Computer Society, 1998) and
- **Transformation** approaches, that process the legacy source code by a series of *automatic transformations* turning it into the target system's source code.

Table table 3.25 provides an overview of the approaches in the two grouping schemes *research areas and target architectures* and *methodologies*.

A more comprehensive overview of Reengineering technologies can be found in (R. Perez-Castillo et al., 2011). To discuss the results of the assessment of approaches in section 3.2, findings are reported based on research areas and target architectures, on methodologies and on individual approaches.

Table 3.25.: Overview of web migration approaches wrt. research areas/target architectures and methodologies

	Encapsulation	Reengineering	Transform.	Other
SOA	Marchetto2008, Gaps2Ws, PRECISO	serviciFi	SOAMIG, SAPIENSA	SMART[56]
Cloud	AMS, NCHC	IC4	REMICS, L2CMH, ARTIST, CloudMIG	AWS Migration[57]
WSE			MIGRARIA, MigraSOA	
Web Application	MELIS, M&S SW, CelLEST, DAS	UWA/UWAT+	TUIMigrate	

3.4 Discussion

Since software migration approaches are typically tailored to a specific environment (cf. brownfield software engineering Hopkins and Jenkins, 2008), the Web Migration approaches have been assessed each on its own above. The two orthogonal dimensions - target architecture and methods (Heil and Gaedke, 2017) - both imply common characteristics. Therefore, this section aggregates the analysis results from the Web Migration approaches evaluated above and discusses commonalities in both dimensions as well as observations about all the entire state of the art.

Table 3.26.: Evaluation Overview, requirements S1 Initial phase, S2 Web Application Target, C1 Risk management, C2 Re-use 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	◐	◐	◐	◐	○	○
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	○	◐	○	●	●	○

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 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 re-use of exposed functionality across several systems in a larger enterprise software environment than on re-use 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. 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 employing feasibility studies, portfolio analysis, and pilots.

Cloud 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. 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. 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 re-use of existing assets to maintain both functionality and interaction.

Web Application A poor support of phases before migration (S1 Initial) was observed in the mainly technical 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 technical approaches is to transfer functionality and interaction into a web-based environment. 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)).

Encapsulation Many Web Migration approaches are using wrapping approaches that create a hybrid Legacy System which does not mitigate 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 perfective 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 restructuring are required (Stroulia, El-Ramly, and Sorenson, 2002). Thus, encapsulation approaches reach the highest levels of re-use 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, Tortora, De Lucia, et al., 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006). Technical 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 Transformation-based Web Migration approaches have a high level of re-use 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 transformation 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 (C3 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, Tortora, De Lucia, et al., 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006). Transformation often requires additional prior re-engineering / re-structuring (Nasr et al., 2010).

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, Tortora, De Lucia, et al., 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 re-use 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).

Independent of the groups and methods, the following section describes general characteristics and observations in the state of the art of Web Migration.

The vast majority of approaches follow an incremental and "*feature-driven iterative migration*" (Menychtas, Konstanteli, et al., 2014) process model.

Initial Phases 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);Khadka2016PHD] in term 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.

Risk Management Similarly 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).

Re-use While re-use 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, Tortora, De Lucia, et al., 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, Scott Tilley, and Canfora, 2006).

Low Abstraction Many approaches (eg. MELIS, SoaMIG, M&S SW, Marchetto2008) operate at a low abstraction level close to code (cf. levels of abstraction in ADM and SOA-MF) , "hindering reusability and maintainability of the obtained system" (Sosa-Sanchez 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.

MD 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)[58]: 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 (R. Perez-Castillo et al., 2011). The model-driven objective of achieving re-use 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 companies 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 re-use of interaction-related assets. Many approaches are only demonstrated with small, artificial sample applications (e.g. conference management (Sosa-Sanchez 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 SMEs (cf. section 2.1.1) are neglected. 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 above. The most promising comprehensive approaches in terms of overall rating (e.g. REMICS,

[58] https://www.eclipse.org/modeling/

ARTIST, UWA/UWAT+) all have high expertise requirements. Integration into ongoing, agile development activities is widely overseen with all approaches designed as stand-alone processes. Only REMICS agile extensions - although not explicitly described - could be integrated.

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 labour-expensive category of 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.

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.

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.

Reuse is mainly considered for backend functionality, continuity of the user interface and user interaction is neglected. There is much focus on SOA and Cloud, migration to Web Applications independent of these two paradigms are not frequently considered besides simple encapsulation approaches. Web Migration approaches are evenly split between model-driven and non-model-driven approaches. The characteristics of SME-sized ISVs are ignored by most approaches, defining complex stand-alone processes that require resources and expertise beyond availability. Integration aspects are overseen in research. To conclude, there are many partial solutions in the Web Migration field, but, in particular comprehensive methodologies like ARTIST, REMICS, UWA/UWAT+ exhibit shortcomings in important requirements and would benefit from a dedicated solution addressing these gaps.

3.5 Summary

This chapter presented relevant standards and reference models and assessed existing Web Migration approaches according to the requirements elicited in chapter 2. A two-dimensional classification scheme was introduced and a detailed discussion 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, demonstrating the relevancy and need for the solution presented in the following chapter. The detailed findings have provided input for the creation of the solution concept.

4 Lowering the Initial Hurdle to Commence a Web Migration

To address the problems identified in the state of the art of Web Migration that keep ISVs, in particular SME-sized ISVs, from commencing a Web Migration, this chapter outlines the AWSM (Agile Web Migration for SMEs) (Heil and Gaedke, 2016) approach. Based on the scenario introduced in section 2.1, an *HCD*[59] *ideation process* comprising *LFA*[60]*-based problem analysis* provides a detailed understanding of the stakeholder-specific situation, concretizes the factors of *effort and risk* from the introduction, identifies three main research objectives and derives three corresponding solution opportunities which are described in the context of the AWSM Methodology and Platform.

4.1 Design Method and Considerations

This section describes the research methodology for designing the solution presented in section 4.2. In section 2.1 we introduced the main stakeholder role as SME-sized ISV with legacy, non-web, desktop application products, and a large existing user base.

The research methodology aims at two objectives: to reach a better understanding of the stakeholder problem, i.e. why are these ISVs hesitant to commence a Web Migration, what are the main concerns that create the organizational resistance, and to systematically devise a suitable approach addressing concrete solution opportunities derived from the problem analysis.

In addition to traditional requirements elicitation, methods from two methodologies have been employed: Human-Centered Design and Logical Framework Approach. These are briefly introduced and the application of methods is described in the following.

[59] Human-Centered Design, cf. http://www.designkit.org/

[60] Logical Framework Approach, cf. https://ec.europa.eu/europeaid/multimedia/publications/publications/manuals-tools/t101_en.htm

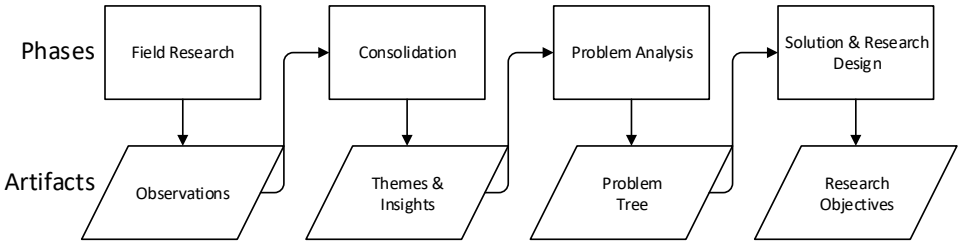


Figure 4.1.: Solution and Research Design Process Flowchart

Human Centered Design (HCD) is a paradigm for designing solutions to problems with a focus on human needs and desires. It has been defined with a technical scope for interactive systems and usability by ISO standard 9241-210 (ISO, 2010). One of the leading proponents and contributors of HCD is the design company IDEO[61] that 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. It is thus a tool for *ideation*, that provides concrete guidelines and methods for conducting *field research* in a design science context, which were applied for this thesis.

Logical Framework Approach (LFA) is a project planning and management methodology employed by the European Commission (European Commission, 2004) and other authorities. It focuses on systematic *problem and stakeholder analysis*, and *objective setting* in the analysis stage and is thus a tool for structuring observations gathered through HCD methods into concrete problem descriptions in this thesis. 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.

4.1.1 Research Process

Figure 4.1 presents the four-phase research process combining methods from HCD and LFA, table 4.1 provides a mapping of these methods on the phases.

Table 4.1.: 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

[61] https://www.ideo.com/

Phase	Methods
Solution & Research Design	HCD How might we, Brainstorm, Get Visual, Prototyping, Feedback integration and iteration

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. Several methods from HCD's Inspire phase were employed: *Recruiting Tools* helped identify people to talk to, 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 *Interviews* in groups and single 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. More than 50 observations of the field research phase were captured and organized using Trello[62] 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. *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*. The results of the problem analysis phase were iteratively improved through a feedback loop with the ISV and are outlined in section 4.1.2.

Solution & Research Design transformed the problem analysis results into a solution design representing a set of research objectives. A sub-tree of relevant problems

[62] https://trello.com/

according to the scope of this thesis section 1.4 was selected and *How Might We* questions were formulated to drive ideation of opportunities (solution ideas) in the collaborative *brainstorm* session. To communicate the solution ideas, *visual* representations (cf. paper prototypes in section 5.3.2.2) and software *prototypes* were created. *Feedback Integration and Iteration* was used to improve the opportunities through stakeholder feedback gathered in a presentation and discussion session at the ISVs headquarters and continued throughout the entire research project. Eventually, partial research questions were derived which define the solution parts on a coarse-grain level. The resulting solution & research design is described in section 4.2.

4.1.2 Problem Analysis Results

As part of the problem 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. Table 4.2 shows the results. The ISV stakeholder is further detailed in 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 processes, 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 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.

Table 4.2.: Stakeholder Map

Stakeholder	Interest/how affected	Capacity & Motivation	Possible Actions to Address Stakeholders
Management	- higher sales - reduced costs for software distribution and maintenance - increased risk and workload through migration	- decision taker role - higher competitiveness due to improved product portfolio and reduced effort	- risk management - communicate benefits through prototypes
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
Doctor's Offices	- innovative features & improved usability - changes in existing work patterns- training effort- new hardware - security and data privacy	- customers - almost no intrinsic motivation	- user training
Patients	- innovative features & improved usability- more interactivity	- customers of customers	- information about new features
Competitors	- idea transfer to own products - competitive disadvantage	- market share	N/A

Stakeholder	Interest/how affected	Capacity & Motivation	Possible Actions to Address Stakeholders
Companies with similar situation	- solution	- improved situation	- dissemination of Results

This 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. 4.2. 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. Effects of this problem such as losing market shares to the competition are not shown for brevity. This competitiveness is jeopardised 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., 2010b). The legacy-characteristic causes of overloading, such as low maintainability due to *side-effects*, *technological deprecation*, *multi-platform environment* and *limited time & resources* are pushing factors in favour 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*. The latter three problems are situational problems that impose constraints on potential solutions as represented in requirements C4 Agile, C3 Exp and C2 Reuse.

4.1.3 Solution & Research Design

The main research question of this thesis, RQ1, directly corresponds to the *hesitation to commence Web Migration* problem and its corresponding sub-tree in fig. 4.2. The detailed problem analysis shows that the *effort and risk* factors described in

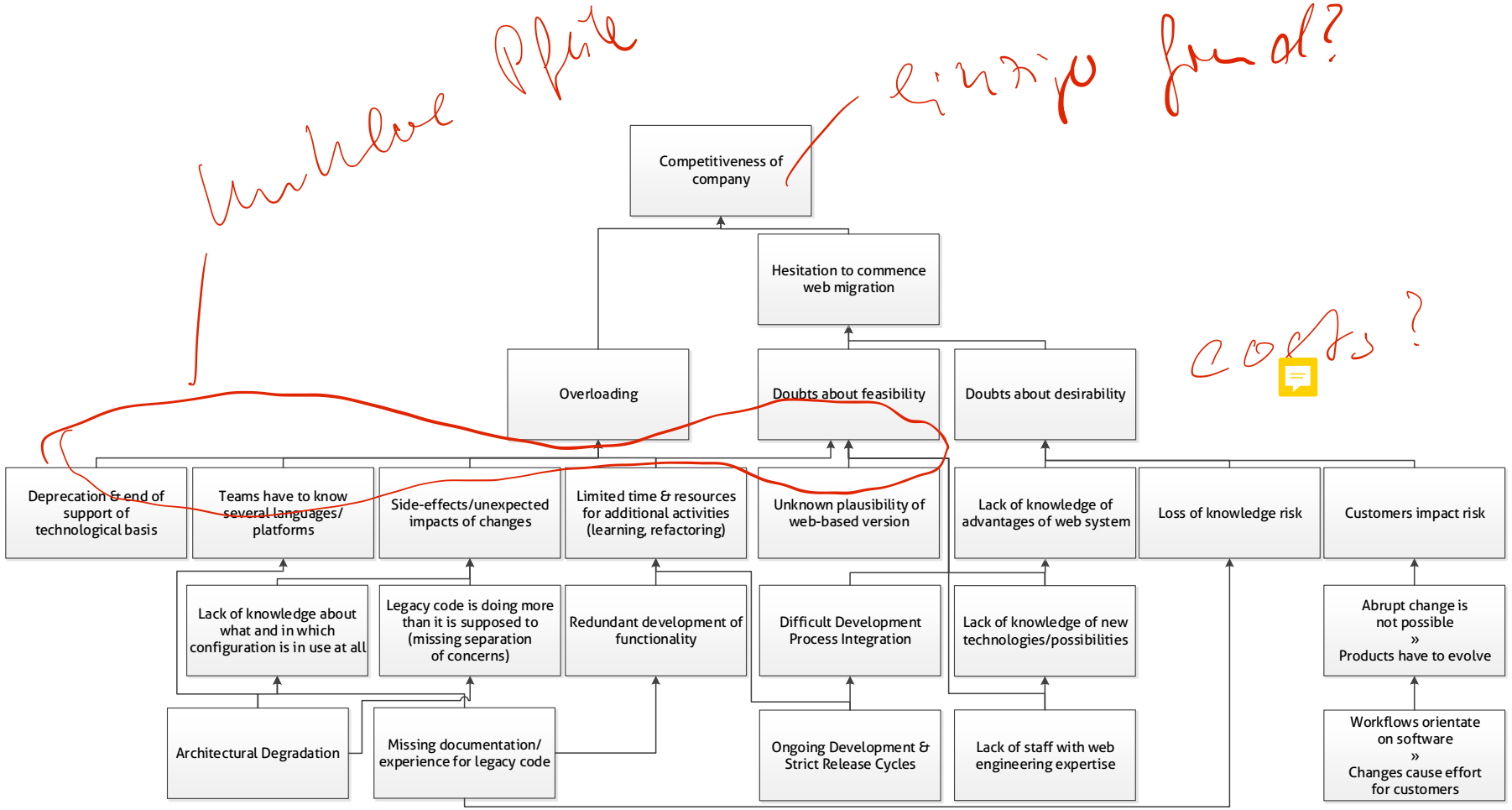


Figure 4.2.: Hierarchical Representation of Problem Domain as LFA Problem Tree

the literature (cf. section 1.3) that make companies hesitant to commence a Web Migration can be concretized into the problems of *doubts about feasibility* and *doubts about desirability* for ISVs with non-webLegacy Systems and large existing user bases. Through reformulation using the How-Might-We method of HCD, the How Might We (HMW) version of the initial research question RQ1

> *How might we support companies to commence Web Migration?*

translates into

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

which defines the solution scope. Aligned with the problem tree, this question is further broken down into the three questions HMW1 - HMW3 in table 4.3.

Table 4.3.: How-Might-We-Questions for Ideation of Research Objectives

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

These questions address the problem sub-trees of feasibility and desirability doubts from the perspective of *functional requirements* (ISO/IEEE, 2017b) in standard requirements engineering. To represent the situation and constraints of the ISV, HMW4 represents the *non-functional requirements* (ISO/IEEE, 2017b) perspective.

Based on RQ1 and HMW4, the research objective of this thesis is:

Research Objective RO1

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 objective, three specific research objectives were identified from RO2 - RO4 employing HCD-based opportunity ideation described in section 4.1.1. These objectives are:

Research Objective RO2

To provide a reverse engineering method, models and tools that allow to identify and manage existing knowledge in legacy source code with limited resources and lack of Web Engineering expertise.

Research Objective RO3

To provide a risk management method, models and tools that demonstrate desirability and feasibility of a potential web-based version of the Legacy System with limited resources and lack of Web Engineering expertise.

Research Objective RO4

To provide an Human-computer interaction (HCI) method, models and tools to control the impact of Web Migration on customers with limited resources and lack of Web Engineering expertise.

These research objectives define the solution specified in the following section.

4.2 Agile Web Migration for SMEs

Our solution to address RO1 to RO4 is *AWSM (Agile Web Migration for SMEs)* (Heil and Gaedke, 2016). Figure fig. 4.3 shows the architecture of AWSM. It consists of the *AWSM Methodology* which is supported by the *AWSM Platform* and leverages three solution components corresponding to research objectives RO2, RO3 and RO4: the AWSM reverse engineering method, the AWSM risk management method and the AWSM customer impact control method. AWSM is structured according to the constructive elements of software engineering for quality ensurance (Wallmüller, 2001) into Principles, Formalisms, Methods, and Tools. The AWSM Methodology describes principles, formalisms, and methods. Corresponding support Tools constitute the AWSM Platform. *Principles* are at the foundation of AWSM and drive design decisions of the AWSM Methodology and Platform in addition to the identified requirements. *Formalisms* are the conceptual basis for the methods and tools. They specify the underlying mathematical model and semantics for knowledge in legacy source code and the definition and algorithm for measuring the similarity between original and migrated user interfaces.

4.3 AWSM Methodology

According to RO1, AWSM addresses shortcomings of existing Web Migration approaches in initial phases of migration projects. Similar to SMART (G. Lewis, Morris, Smith, et al., 2008; G. Lewis, Morris, O'Brien, et al., 2005), it does not impose a specific migration process. Thus, the AWSM Methodology focuses on principles,

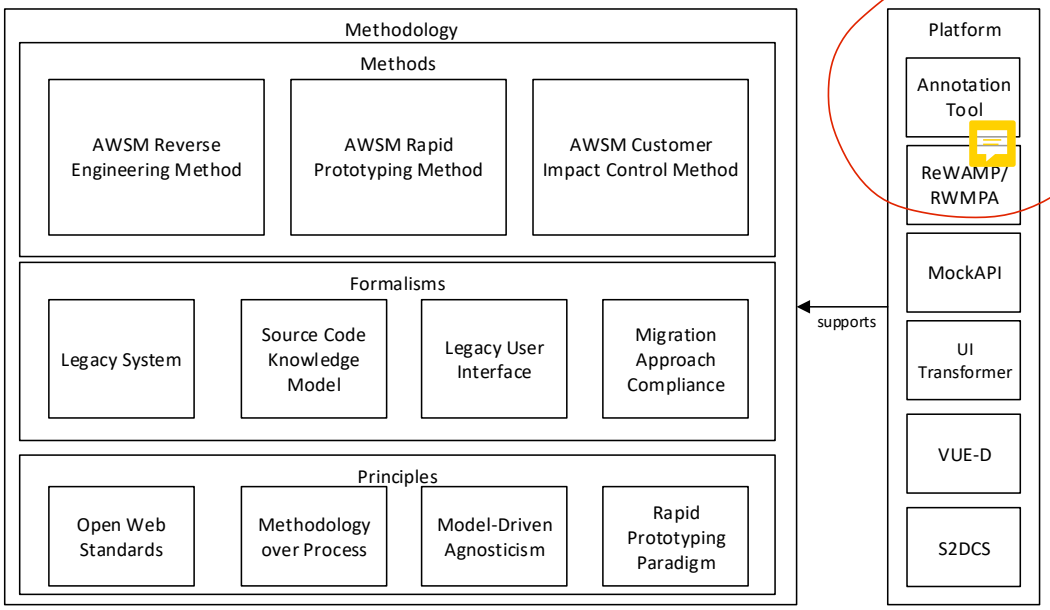


Figure 4.3.: AWSM Overview

formalisms, and methods for Web Migration initiation and integration with existing comprehensive migration methods. It can be combined with incremental reengineering and transformation processes. AWSM addresses a package-oriented (Brodie and Stonebraker, 1995) feature-driven (Menychtas, Konstanteli, et al., 2014) incremental process model.

4.3.1 Principles

The following principles form the basis for the AWSM Methodology and Platform and, in addition to the requirements, drive the design decisions.

Principle P1 Open Web Standards

The use of standards is widely accepted in software engineering and Web Engineering. This principle advocates using and extending open Web standards, in particular W3C specifications and OMG standards, to allow interoperability with existing methods and tools and provide contributions that can easily be used and enhanced by future research (cf. OpenStand Principles[63]). This especially applies to underlying data models, provided interfaces and technologies used.

Principle P2 Methodology over Process

As shown in section 3.2, there is a wide range of existing Web Migration approaches with corresponding process models. However, these approaches are lacking in the aspects important for Web Migration initiation for ISVs with limited resources and lack of Web Engineering

[63] https://open-stand.org/about-us/principles/

expertise. Thus AWSM aims at providing a methodology that addresses these shortcomings with formalisms, methods, and tools. This principle advocates an openness for existing Web Migration reengineering and transformation process models to prefer re-use over re-definition of a new Web Migration process model. Therefore, AWSM methods need to define mappings for extension of the most relevant Web Migration approaches such as REMICS, ARTIST or UWA/UWAT+.

Principle P3 Model-Driven Agnosticism

The current Web Migration landscape is divided on the use of Model-Driven Engineering (MDE). In section 3.2 half of the approaches employ MDE in some form, half of the approaches does not. Similarly, MDE practices are employed in varying degrees in forward Web Engineering (Moreno et al., 2008). Thus this principle advocates an openness to different degrees of Model-Driven and Non-Model-Driven approaches for the core methods of the methodology. In particular reverse engineering, which forms the basis of reengineering and transformation, should be laid out agnostic but adaptable to concrete Model-Driven or Non-Model-Driven methods to allow interoperability with a wide range of existing Web Migration approaches.

Principle P4 Rapid Prototyping Paradigm

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) has become particularly popular in the context of Agile Development (Abrahamsson et al., 2002) and Human-Centered Design (IDEO, 2015). Thus, this principle advocates applying the Rapid Prototyping paradigm to the Web Migration domain by favoring the creation of concrete, tangible means of communication to demonstrate feasibility and desirability of a web-based version of the migrated Legacy System over theoretical and general argumentation and narrowly focused technical feasibility studies.

4.3.2 Formalisms

The following formalisms form the theoretical basis of the AWSM methods. They define basic conceptual models independent of their concrete implementation in the AWSM platform. The conceptual architecture of the AWSM platform is discussed in section 4.4. Following principle P1, the formalisms are based on the OMG's Knowledge Discovery Meta-Model specification, which is designed in a partitioned

way to allow adopters to effectively use only parts "necessary for their activities" (Object Management Group, 2016a). 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).

Legacy System

The Legacy System is the basis for any Web Migration and the central object of investigation of this thesis, serving as the starting point for the three AWSM methods. KDM (Object Management Group, 2016a) provides a comprehensive metamodel for describing Legacy Systems. This section describes the conceptual model of a Legacy System based on a subset of KDM concepts which provides the common foundation for the understanding of Legacy Systems in the context of the AWSM methodology. In particular, the *inventory model* of the KDM *source package*, the *platform package* and the *UI package* are relevant for AWSM. A Legacy System $\mathfrak{L}$ is a tuple

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

B is the legacy codebase consisting of a set of source code files (SourceFile in KDM), configuration files (KDM: *ConfigFile*) and other textual documents (KDM: *Document*). Dep are the dependencies of the Legacy System (KDM: *LinkableFile*), in particular third-party libraries (KDM: *LibraryFile*). The dependency matrix $\underline{D} : \{1, \ldots, |B|\} \times \{1, \ldots, |Dep|\} \mapsto [0, 1]$ is a binary matrix with $\underline{D}_{i,j} = 1 \iff f_i \in B$ depends on $dep_j \in Dep$, else $\underline{D}_{i,j} = 0$. E are the executables (KDM: *ExecutableFile*) of the Legacy System. In some scenarios 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. D represents the persistent data of the Legacy System such as databases (KDM: *DataManager*), storage files (KDM: *FileResource*), XML or JSON files (KDM: *DataFile*). U is the set of 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 4.3.2.

This conceptual model describes 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. In contrast, the following source code knowledge model formalism describes the relevant *software assets* (Object Management Group, 2016a)

that are represented through the artifacts but not necessarily existing in explicit forms.

Source Code Knowledge Model

The source Code Knowledge Model (SCKM) is at the core of the AWSM reverse engineering method. As argued in the introduction, Legacy Systems contain valuable *knowledge from the problem and solution domain* (Marcus et al., 2004) that must be preserved throughout migration. Loss of knowledge is a risk (cf. C1 Risk and HMW1) that needs to be addressed to assist Web Migration initiation (cf. RO2). Knowledge in legacy code bases is implicit: 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 (Hansen et al., 1999) 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 (Chikofsky and Cross, 1990) and stored in a *knowledge base* (ISO/IEEE, 2017b) to feed into subsequent reengineering or transformation. OMG's ADM provides a basic model for legacy source code knowledge above procedure level in the KDM specification as described in section 3.1.1. Thus it is used as the basis for the SCKM. However, KDM is focused on redocumentation knowledge, i.e. equivalent representations of knowledge within the same abstraction level, the implementation level (Chikofsky and Cross, 1990), allowing to describe the structure of the legacy codebase in terms of modules, classes, methods, files. In particular, KDM often 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 (Rodríguez-Echeverría, Conejero, Linaje, et al., 2010), within the source code. The SCKM extends KDM towards the representation of design recovery knowledge, i.e. knowledge on higher levels of abstraction while maintaining the connection to its occurrence in structural parts of the source code.

Migration aims at retaining the functionality of Legacy Systems in new environments (Bisbal et al., 1999), with functionality referring to domain knowledge and business logic (Wagner, 2014). Thus, there are two basic types of knowledge in legacy source code: Features and domain knowledge. *Features* describe functionality of the Legacy System (What) (IEEE Computer Society, 2014; Rubin and M. Chechik, 2013; V. Rajlich and Wilde, 2002) and can be represented as user stories, scenarios, use cases (KDM: *ScenarioUnit*) etc.. *Domain knowledge* is the knowledge supporting the implementation of features, describing parts of the problem and solution domain of the Legacy System (How) (Marcus et al., 2004; V. Rajlich and Wilde, 2002). Typically, three categories corresponding to a classical three-tier-architecture are considered in program decomposition: *presentation* (KDM *UI domain*), *application logic* (KDM

ConceptualFlow) and *persistence* (KDM *data domain*) (Canfora et al., 2000). Presentation comprises information about the user interface layout and the user interaction handling. Persistence defines knowledge about data models, data flow, caching, etc. The SCKM extends the three basic categories allowing a more detailed distinction of domain knowledge in the source code. In particular, *business processes* and *business rules* (KDM *RuleUnit*) (Aversano et al., 2001; Harry M. Sneed et al., 2010a; Wagner, 2014; W. M. Ulrich, 2011) are crucial problem domain knowledge. Solution domain knowledge comprises *algorithms* (KDM: *BehaviorUnit*), *configuration* (KDM: *ConfigFile*) and *deployment* (KDM: *Deployment*). Note that both business processes and algorithms describe processes in the source code, but business processes are processes that exist in the problem domain whereas algorithms are solution domain processes. Furthermore, Legacy Systems can contain *explanatory* natural-language knowledge embedded in the code as comments (KDM: *CommentUnit*). A similar distinction can be found in the taxonomy of legacy artifacts in ARTIST (Brunelière, Cánovas, et al., 2013).

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 of $\mathfrak{L}$. These knowledge types can be represented by a variety of representations like UML class diagrams, BPMN diagrams, flow charts, SBVR[64] rules, other forms of models (KDM: *Model*), informal or semi-formal natural language texts etc., depending on the intended use (cf. principles P2 and P3) and can even have no representation ($r = 0$) since knowing the type of a particular piece of $\mathfrak{L}$ is already knowledge.

Thus, an instance of knowledge k in the Legacy System $\mathfrak{L}$ is a tuple

$$k = (t, r) \tag{4.2}$$

of type $t \in T$ and a representation r.

$$T = \{\text{feature}, \text{presentation}, \text{persistence}, \text{business rule}, \text{business process}, \text{algorithm}, \text{configuration}, \text{deployment}, \text{explanatory}\} \tag{4.3}$$

As this knowledge is implicitly represented and distributed (e.g. partial classes) in the source code, it is crucial to codify the connection between the knowledge representation and its occurrences in the source (V. Rajlich and Wilde, 2002), in particular considering the importance of decomposability for migration (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Canfora et al., 2000). Based on the conceptual model of KDM, the location l can be specified as a reference (KDM: *SourceRef*) to a specific segment of code s (KDM: *SourceRegion*) within a physical

[64] https://www.omg.org/spec/SBVR/

source code artifact $f \in B$ (KDM: *SourceFile*) of the legacy codebase. The source file content is interpreted as linear stream of characters, allowing to define a segment in terms of inclusive start α and end ω position in that stream:

$$l = (s, f) \tag{4.4}$$

$$s = (\alpha, \omega),\ \alpha \in \mathbb{N}_0, \omega \in \mathbb{N}_0 \tag{4.5}$$

The result of associating a piece of knowledge (*intension* K. Chen and Václav Rajlich, 2010) with its location in the legacy code base (*extension* K. Chen and Václav Rajlich, 2010) is an *annotation* (KDM: *AnnotationElement*) $a \in K^* \times L^*$ which enables *traceability*:

$$a = (k, l) \tag{4.6}$$

Thus, the reverse engineering method of RO2 can be considered the application of a function $re : B^* \mapsto A^*$ that maps a legacy codebase B onto a set of annotations $A = \{a_1, a_2, \ldots, a_n\}$: $re(B) = A$, identifying a set of knowledge instances $K \in K^*$ and their occurrences $L \in L^*$ in the source code. The information forms a *legacy code knowledge base* $\mathbb{K}_B$:

$$\mathbb{K}_B = (K, A) \tag{4.7}$$

Chapter 5 describes the interoperable and queryable implementation of $\mathbb{K}$ through ontological modeling of the SCKM using semantic Web technologies.

Legacy User Interface

The user interface is an essential concern of Web Migration since migration from a desktop GUI to a web-based GUI rendered in the browser visibly changes the "*look and feel*" (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Distante, Perrone, et al., 2002) and impacts existing users. This formalism defines a conceptual model of the legacy user interface that serves as the basis for the AWSM customer impact control method. User interfaces are described in terms of Task, Behavior, Thesaurus, Layout and Material (Bakaev, Khvorostov, et al., 2017a). This conceptual model focuses on the layout aspect, because the layout 1) necessarily undergoes the most visible change through Web Migration, whereas changes in e.g. Task or Thesaurus can be avoided, 2) is the basic design artifact of the new Web UI, influencing other aspects like Behavior and 3) has the most degrees of freedom during re-design, in contrast to

e.g. Material which is governed by the platform. Based on U, the set of resources (KDM: *UIResource*) that form the user interface of the Legacy System, a concrete legacy user interface $u \in U$, i.e. top-level UI container elements without parents (KDM: *UIDisplay*) like screens, windows, dialogues, is a tuple

$$u = (C_u, w, h), C_u \subseteq C_{\mathfrak{L}}, w \in \mathbb{N}_0, h \in \mathbb{N}_0 \tag{4.8}$$

C_u is the subset of UI controls of that particular user interface, w and h are the width and height of the user interface. Since GUIs are rendered on raster scanned screens, the smallest unit of display is a pixel. Thus, lengths like w and h and positions are represented as non-negative integer numbers in the unit of pixel. No statement is made with regard to the physical size of a pixel, which may vary depending on the hardware. $C_{\mathfrak{L}} \subset T_c \times \mathbb{N}_0^4 \times C_{\mathfrak{L}}$ is the set of all UI controls of the Legacy System $\mathfrak{L}$. A UI control $c \in C_{\mathfrak{L}}$ (a concretisation of *UIElement* using KDM's generic extension mechanism) is a tuple

$$c = (t_c, b, c_p), t_c \in T_c, b \in \mathbb{N}_0^4, c_p \in (C_{\mathfrak{L}} \cup \{0\}) \setminus \{c\} \tag{4.9}$$

with t_c being the type of the UI control, b the rectangular bonding box occupied by c within the surrounding container and c_p the parent UI container. UI controls can be nested inside other UI controls (e.g. a button inside a form inside a window). If a UI control contains at least one other UI control, it is referred to as UI container. The nesting is represented via the c_p references to the parent. For root-level UI controls c_p is $c_p = 0$. A UI control c cannot be contained within itself. Thus it is excluded from the domain of possible parents. The resulting hierarchy of UI controls forms a tree-based representation of the UI which is commonly used for further UI analysis (Grechanik, C. W. Mao, et al., 2018; Roy Choudhary, Prasad, et al., 2014b; Sanoja and Gancarski, 2014; Cai et al., 2003). There can be a wide variety of UI control types T_c:

$$T_c = \{\text{label}, \text{button}, \text{input}, \text{checkbox}, \ldots\} \tag{4.10}$$

The bounding box $b \in \mathbb{N}_0^4$ 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.

$$b = (x, y, w, h) \tag{4.11}$$

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 $c = (t_c, b, c_p)$ has a parent UI container

$c_p \neq 0$ then the coordinates of its bounding box $b = (x, y, w, h)$ are relative to c_p, 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 c_p.

Chapter 7 describes an empirical method to measure the similarity between legacy and Web user interfaces and a method for automatic transformation, based on this conceptual model.

Migration Approach Compliance

To assess a set of migration approaches with regards to suitability for a specific migration scenario, it is necessary to define a conceptual model of compliance between an approach and a given scenario. This formalism defines a *vector space model* (Salton et al., 1975) as commonly used in information retrieval for calculating the compliance of the approach. Assessment is calculated according to k criteria. The set of n candidate approaches A_C consists of vector representations $\vec{a_c} \in \mathbb{N}_0^k$ of each approach according to the k criteria. Likewise, scenario $\vec{sc} \in \mathbb{N}^k$ is a vector defining the intended criteria. Let $x_i = \langle a_{ci}, s_{c_i}^{-1} \rangle$ for $1 \leq i \leq k$ form an auxiliary vector $\vec{x}$ that represents the compliance of $\vec{a_c}$ per criteria through component-wise division of $\vec{a_c}$ and $\vec{sc}$. Compliance factor $R \in \mathbb{R}_0^+$ can be defined as the 1-norm of $\vec{x}$:

$$R = ||\vec{x}||_1 = \sum_{i=1}^{n} |x_i| \tag{4.12}$$

To allow a more fine-grained balance of criteria, they can be weighted with regard to their importance using a weighting vector $\vec{g} \in (\mathbb{R}^+)^k$. The weighted compliance factor R^* can then be defined as:

$$R^* = \frac{\langle \vec{g}, \vec{x} \rangle}{||\vec{g}||_1} = \frac{\sum_{i=1}^{n} |g_i \cdot x_i|}{\sum_{i=1}^{n} |g_i|} \tag{4.13}$$

Section 4.4.1 describes the application of the compliance factor for supporting Web Migration strategy selection through a decision support system.

4.3.3 Methods

To address research objectives RO2, RO3, and RO4 respectively, AWSM provides the three methods outlined in the following. We follow the ReMiP taxonomy of Sneed et al. (Harry M. Sneed et al., 2010a) as shown in fig. 4.4, which divides *migration processes* into distinct sequential or parallel *migration phases*, throughout this thesis. These phases comprise at least one *migration activity* which employs one or more *migration methods*. The three AWSM methods described in the following are migration methods as they provide a conceptual model for conducting a migration activity. For each AWSM method, at least one *migration technique* is specified,

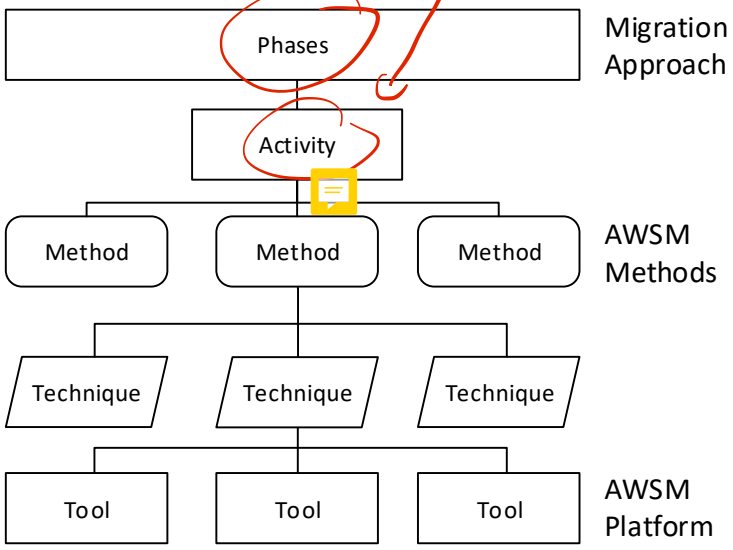


Figure 4.4.: ReMiP Taxonomy (adapted from Harry M. Sneed et al., 2010a) mapped to AWSM

detailing how to conduct a part of a method. Each technique can be supported by a software implementation as *migration tool*, which are parts of the AWSM Platform.

4.3.3.1. AWSM Reverse Engineering method

Addressing RO2, the AWSM Reverse Engineering method (AWSM:RE) specifies an approach to identify and manage problem and solution domain knowledge in legacy source code according to the conceptual models of Legacy System and SCKM introduced before. Unlike existing redocumentation and design recovery methods, AWSM:RE focuses 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). Following principle P1, AWSM:RE represents reverse-engineered knowledge in a queryable and interoperable way using semantic Web technologies thus providing a basis for subsequent reengineering and transformation approaches. The concretization of the conceptual SCKM as OWL-Ontology allows adaption of AWSM:RE to different Web Migration methods at varying degrees of model-driven adoption, according to principle P3. AWSM:RE is described in chapter 5.

4.3.3.2. AWSM Risk Management method

The AWSM Risk Management method (AWSM:RM) addresses RO3 by specifying a solution to demonstrate desirability and feasibility of a potential web-based version of the Legacy System based on the existing software artifacts as defined in the conceptual model of the Legacy System which can be achieved with only limited

resources and Web Engineering expertise. Unlike existing risk management methods in the early stages of Web Migration, it focuses on creating 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) according to principle P4 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 leveraging the new WebAssembly W3C standard (W3C, 2018b) in compliance to principle P1. AWSM:RM is described in chapter 6.

4.3.3.3. AWSM Customer Impact Control method

The AWSM Customer Impact Control method (AWSM:CI) addresses RO4 by providing a mechanism to control the impact of Web Migration on the user bases of existing customers with limited resources and lack of Web Engineering expertise. Based on the conceptual model of legacy user interfaces introduced above, 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. Unlike existing automatic analysis methods for Web user interfaces that rely on DOM[65] 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 measures. Following principle P1, analysis results are made available using JSON. AWSM:CI is described in chapter 7.

4.3.3.4. Mapping to the Reference Migration Process

Table 4.4 provides a mapping of the AWSM methods onto the migration phases and disciplines of the Reference Migration Process (Harry M Sneed et al., 2010b) in order to facilitate integration with existing Web Migration approaches as per principle P2. While the techniques of AWSM:RE and AWSM:RM belong to the earliest ReMiP 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 as per RO1.

[65] Document Object Model, cf. (W3C, 2015)

Table 4.4.: Mapping of AWSM Methods to ReMiP

AWSM 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
AWSM:CI	Migration & Transition	Target Design, Implementation	Migration Environment

4.4 AWSM Platform

The AWSM Platform comprises all support tools for the AWSM methodology. It is a web-based reverse engineering and Web Migration management platform to support management and migration engineer stakeholders (cf. table 4.2), providing 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, knowledge discovery tools, a process guidance and generation tool for Rapid Web Migration Prototyping and a customer impact control measurement tool. To achieve good integration into ongoing development activities (cf. C4 Agile), the AWSM Platform integrates with software project management platforms for management stakeholders and with integrated development environments (IDEs) for migration engineer stakeholders (similar to the ARTIST Methodology Process Tool (MPT) Konstanteli et al., 2015; Menychtas, Konstanteli, et al., 2014). Principle P1 drives the design of the AWSM Platform. The tools which constitute the AWSM Platform addressing research objectives RO2, RO3 and RO4 are described in the following chapters together with the AWSM methods which they support. The AWSM Strategy Selection Decision Support System, which addresses RO1, is described in the following section.

4.4.1 AWSM Strategy Selection Decision Support System

The AWSM Strategy Selection Decision Support System (S2DCS) is a part of the AWSM Platform that supports ISVs to select a global Web Migration strategy (cf. ReMiP *strategy selection* discipline Harry M Sneed et al., 2010b; Gipp and Winter, 2007). According to principle P2, AWSM is a methodology that provides solutions addressing the identified gaps in current Web Migration approaches, designed to be open for integration with existing Web Migration reengineering and transformation process models. As shown in section 3.2, there is a wide range of approaches. In

our systematic mapping study (Heil and Gaedke, 2017), we identified and evaluated 122 primary studies, comprising not only academic publications but also existing software tools. Selecting a suitable approach for the specific situation among this plethora of possibilities is difficult for ISVs.

Thus, based on the results of the survey (Heil and Gaedke, 2017), a decision support system was created. Addressing RO1, the S2DCS supports ISVs to commence a Web Migration by enabling them to select a suitable strategy based on existing approaches and tools. Using S2DCS ISVs can either decide to employ the approach that suits best to their specific Web Migration problem or build their own composite strategy based on the information provided from a set of appropriate approaches. As S2DCS represents a *faceted search* interface (Tunkelang, 2009) to a knowledge base of existing approaches and tools, 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 4.1.2, the faceted search is realized as a guided dialogue interaction. The S2DCS user answers a set of questions 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 (Nguyen et al., 2009) 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 the compliance factor formalism introduced in section 4.3.2 and ordered by their rank. The criteria for selecting approaches are defined by target and source system characteristics (e.g. architectures, technologies), supported migration phases (e.g. legacy analysis, implementation, etc. cf. ReMiP) and dynamic criteria depending on the selection of the other criteria (e.g. SOAP or REST when SOA target was selected).

The user can review the details of the resulting approaches 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. In the context of AWSM, S2DCS supports ISVs to determine a suitable Web Migration strategy (migration approach in fig. 4.4, cf. principles P2 and P3) into which the AWSM methodology gets embedded to address the identified shortcomings of existing approaches.

TODO:screenshot in-appendix, URL ref

4.5 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 several principles, formalisms, methods, and tools to facilitate Web Migration initiation. Its 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 include KDM-based conceptual models of Legacy Systems and knowledge in Legacy Systems and of legacy user interfaces. The methods specify several techniques for knowledge rediscovery, risk management and customer impact control. The tools of the AWSM Platform provide implementations of the proposed ideas and techniques, facilitating application of the AWSM methods and focusing on integration with migration approaches, ongoing development activities and environments. The following three chapters provide a detailed view on the methods and tools of the AWSM Methodology and Platform.

Das Kapitel ist DEIN Hauptbeitrag. Er sollte klar darstellen, daß Dein Hauptresearch Goal und sein Unterziel gelöst wird. Es sollte darstellen wie das geht – das gelingt Dir nicht. Insgesamt überzeugt die Struktur dieses Kapitels nicht. Dir gelingt es nicht die Aufgabe einfach darzustellen. Das liegt vermutlich a zu viele unnötige Text-Defs. und zu wenig Bilder und was um Darstellung es und was eigentlich zu was (ROS) beiträgt.

5 AWSM Reverse Engineering Method

This chapter addresses research objective RO2:

> To provide a reverse engineering method, models and tools that allow to identify and manage existing knowledge in legacy source code with limited resources and lack of Web Engineering expertise.

First, the current situation is analyzed to identify requirements and review related work, a conceptual model and implementation of mechanisms to address RO2 are described, and the evaluation of the method including detailed experimentation results is reported.

5.1 Research Questions

To design and evolve a solution for research objective RO2, this chapter addresses three research questions:

AWSM:RE Research Question 1: How to identify problem and solution domain knowledge in Legacy Systems with limited resources?

AWSM:RE Research Question 2: How to transfer the Crowdsourcing paradigm into a reverse engineering context?

AWSM:RE Research Question 3: 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?

5.2 Analysis

Legacy Systems represent valuable problem and solution domain knowledge about business processes, rules, etc. (Aversano et al., 2001; Harry M. Sneed et al., 2010a; Wagner, 2014; Bodhuin, Guardabascio, et al., 2002; W. M. 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). The importance of knowledge and knowledge management in Web Migration has recently

been acknowledged (Razavian, Nguyen, et al., 2010; Razavian, 2013). Relevant and non-relevant parts of business logic must be distinguished (W. M. Ulrich, 2011). Knowing where the knowledge is located in the source code is essential because decomposability is a vital pre-requisite for migration (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Canfora 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., 2010a; Ian Warren, 2012; Batlajery et al., 2014; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008), inhibiting a systematic way to deal with changes (Sosa-Sanchez 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, similar to tacit knowledge in organizations that is not expressed explicitly but guides human behavior (Nonaka, 2008), the knowledge Legacy Systems is not explicitly documented but governs how they operate. 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 the knowledge is essential for successful migration and requires *reverse engineering* techniques and tools (Harry M. Sneed et al., 2010a) when the legacy source is the only available representation of knowledge (Software Engineering Standards Committee of the and IEEE Computer Society, 1998). While migration in general aims at retaining functionality of the Legacy System in a new environment (Bisbal et al., 1999), SOA migration approaches show that explicit modeling of business processes is also an opportunity to renew and extend them with new functionality, representing a target state with extended scope (Sosa-Sanchez et al., 2014).

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 in the light of organizational resistance (Khadka, Batlajery, et al., 2014) hindering knowledge sharing. Existing re-documentation approaches focus on solution domain knowledge, e.g. discovery tools in (Amazon Web Services Inc., 2018) on systems level (databases, servers, OS, etc.), OMG Standards KDM and ASTM focus mainly on the syntactic level. "Knowledge discovery is often limited to reverse engineering of legacy code. The business process and rules recovery is poorly addressed" (Mohagheghi and

Sæther, 2011). Other knowledge discovery approaches focus on extracting instances of specific metamodels, e.g. UWA in RE-UWA (Bernardi et al., 2009).

Manual knowledge discovery is time-consuming and difficult as surveys with software engineering professionals show (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 (CanforaHarman and Di Penta, 2007). This is in line with results from our own experiments employing machine learning technologies for knowledge discovery as indicated in section 5.3.1.4. 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.

5.2.1 Requirements

The following requirements have been derived based on RO2, 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.

5.2.2 Related Work

Reverse Engineering is "a software engineering approach that derives a system's design or requirements from its code" (ISO/IEEE, 2017b) which falls under *perfective maintenance* (ISO/IEEE, 2006) activities, improving maintainability. Reverse Engineering and program understanding have seen significant research interest (CanforaHarman and Di Penta, 2007). (S.R. Tilley et al., 1996) provides a reference framework for understanding Reverse Engineering and program understanding. Reverse engineering is "a process of examination" aiming at the identification of artifacts, their relationships and the generation of abstractions (IEEE Computer Society, 2014) with various cognitive factors like limited human memory and the

size and complexity of Legacy Systems making it very difficult (S.R. Tilley et al., 1996). These artifacts are at different levels: *data*, the factual basis, *knowledge*, the sum of data and derived information such as relationships, rules, and *information*, "contextually and selectively communicated knowledge" (S.R. Tilley et al., 1996). Two subareas of reverse engineering are distinguished: *redocumentation* extracts representations at the same level of abstraction, *design recovery* creates higher level abstractions (Chikofsky and Cross, 1990).

The canonical approach of reverse engineering can be considered a three-step process: *model*, to construct domain models using conceptual modeling, *extract*, to gather the raw data from the Legacy System, and *abstract*, to create abstract representations that support exploration and subsequent engineering activities (S.R. Tilley et al., 1996). Data extraction can be achieved through automatic static or dynamic (Stroulia and Systä, 2002) analysis and informal data extraction (e.g. interviews). *Knowledge management* is crucial for migration (Razavian, 2013). From a knowledge management perspective, reverse engineering realizes the objective of codification (Hansen et al., 1999). Thus representation and management of extracted data is important (Razavian, 2013). Extracted data is represented according to a suitable *data model* supporting storage and retrieval for subsequent analysis, forming a potentially vast knowledge base. Navigation through the hyperspace of information related to the Legacy System is crucial and can be supported through "powerful query and analysis languages" (S.R. Tilley et al., 1996). The idea of making code navigable through dedicated query languages (Paul and Prakash, 1996; Mendelzon and Sametinger, 1995) has long been a part of reverse engineering research. However, these early approaches focus on low-level code representations. Current *code query technologies* follow a "extract-abstract-present paradigm" (Khadka, Reijnders, et al., 2011). Query languages are particularly relevant for integration with model-driven methods which typically consist of "1. formal definition of valid models (metamodels), 2. querying of models (query languages) and 3. transformation of models (transformation languages)" (Fuhr et al., 2013).

(CanforaHarman and Di Penta, 2007) provides an extensive overview of reverse engineering approaches. The survey outlines the evolution of reverse engineering, from basic program understanding for procedural languages to design recovery in object-oriented languages and analysis of Web Applications. Program understanding comprises areas like identification of duplicates, code smells, program slicing. Design recovery has addressed design pattern identification, feature location, and integration of dynamic and static analysis. Visualization of code focuses on abstracting UML representations or depicting metrics, e.g. using polymetric views. Increased integration of dynamic and static analysis, binary reverse engineering, application of machine learning and better integration with forward engineering are trends in reverse engineering research. The survey points out that while redocumentation

approaches have reached a mature level of automation, automated design recovery lacks precision or recall and thus require human expert intervention.

Among the approaches assessed in section 3.2, REMICS (Barbier, Henry, et al., 2013; Barbier, Deltombe, et al., 2013; Krasteva et al., 2013) provides comprehensive consideration of reverse engineering in its *Recover Activity Area*. It aims at complete recovery of knowledge of the selected candidate legacy components for reengineering. Based on the code, system knowledge is recovered and refined, and then further abstracted as system and requirements model. REMICS uses KDM/ASTM for knowledge representation, UML for system model and ReDSeeDS RSL (Kaindl et al., 2009) for requirements. The goal is to automatically recover application logic through static and dynamic analysis and abstract models for subsequent model-driven transformation. For static analysis, parser-generated KDM/ASTM models are transformed into a UML-based PIM. For dynamic analysis, user interaction traces are recorded, transformed into RSL and manually split/merged to form use cases. REMICS recovery puts a strong focus on the technical perspective and structure of the Legacy System and extracts knowledge only as required for the following model-driven transformation process.

MARBLE (Modernization Approach for Recovering Business processes from Legacy Systems) (Pérez-Castillo, Guzmán, et al., 2011; Pérez-Castillo, Fernández-Ropero, et al., 2011) proposes a comprehensive business process recovery method for Legacy Systems based on ADM, highlighting the similarity between reverse engineering and archeology. It follows ADM's horseshoe reengineering model with a focus on model-driven engineering. Static analysis is the primary technique of MARBLE and is combined with human expert supervision. Recovered knowledge is represented using KDM. It distinguishes four levels of abstraction, defining the transitions between them based on reverse engineering - from code to models - and QVT transformations - between PSM, KDM-based PIM and BPMN business process models. The MARBLE method advocates static over dynamic analysis due to the simple implementation of syntactic parsers and the possibility to quickly repeat analysis when the legacy code base is changed. This is very likely due to ISV's ongoing development activities inhibiting a complete *freeze* (ISO/IEEE, 2017b) of the code. MARBLE PSMs are representations of Java language constructs like variables, classes, methods automatically created by a custom syntactic parser. A PIM KDM representation is derived using a model-to-model QVT-Relations transformation. The final BPMN model of the business processes is generated using QVT-Relations based on well-known patterns. Manual intervention is required to refine the final models. The resulting generated BPMN models are rather close to code-level and require domain experts to achieve problem domain level abstraction. While MARBLE defines a generic framework which can potentially include dynamic analysis and other sources, it is still limited to business processes; other knowledge types are not regarded.

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). The resulting connection between "human-oriented concepts and their implementation-oriented counterparts" allows migration engineers to locate code fragments of interest (S.R. Tilley et al., 1996), which supports maintenance and migration activities (Marcus et al., 2004). The level of abstraction of concepts can be higher than code-level, it is considered "pattern-matching at the end-user application semantic level" (S.R. Tilley et al., 1996), i.e. including *problem domain knowledge* (Marcus et al., 2004). Following the definition in (V. Rajlich and Wilde, 2002):

Definition 5 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.

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. An example of manual Concept Assignment in the Web Migration approaches assessed in section 3.2 can be seen in MIGRARIA which applies it for labeling services (Sosa-Sanchez et al., 2014; Sosa et al., 2013). Automated Concept Assignment was observed in M&S SW (Bodhuin, Guardabascio, et al., 2002). The static analysis approach based on objects, however, is based on a simplified understanding of concepts, ignoring that concepts do not necessarily have a class of their own or can appear distributed across several classes (V. Rajlich and Wilde, 2002).

Automation of Concept Assignment has been proposed in the context of *Concept Location*, which is the process of finding code implementing a specific problem or solution domain concept (Marcus et al., 2004). For a subset of concepts, *features* (V. Rajlich and Wilde, 2002), this is known as *Feature Location* and has been successfully achieved using dynamic analysis (Rubin and M. Chechik, 2013). Tracing features in code is particularly relevant for large-scale re-use-oriented development methodologies like software product line engineering (Rubin and M. Chechik, 2013). Concepts are formalized as a triple of name, intension (meaning), extension (related parts of the Legacy System) (K. Chen and Václav Rajlich, 2010). Concept Location can be considered the inverse process of Concept Assignment as it takes an intension as input and produces extensions as output. Vice versa, Concept Assignment is a

recognition process that takes an extension (e.g. a segment of code) as input and produces a set of intensions as output. (K. Chen and Václav Rajlich, 2010) While migration requires the recognition process, concept location can be performed on the results of Concept Assignment and thus be beneficial also for other maintenance activities. In this thesis, we follow 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 has been combined with program slicing in an approach called concept slicing (Gold, Harman, et al., 2005), which can be used to create wrappers (Canfora et al., 2000) or extract service implementations (Khadka, Reijnders, et al., 2011). Being based on *Hypothesis-Based Concept Assignment* (HB-CA), a-priori creation of the knowledge base representing the domain model is required based on which a "best guess" (Gold, Harman, et al., 2005) is made. While this *plausible reasoning* approach (Biggerstaff et al., 1993) is sufficient to support software maintenance activities, it does not reliably address the risk of knowledge loss for Web Migration (C1 Risk) and a-priori creation of the knowledge base represents significant effort for large Legacy Systems, ignoring the limited resources as of RO2.

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)).

Crowdsourcing is a work organization paradigm that is employed in AWSM:RE to address this situation with a focus on the aspect of limited resources in RO2. Jeff Howe (Howe, 2006) initially coined the term:

Definition 6 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 research soon gained importance, defining it as "online, distributed problem-solving and production model" and outlining a research agenda (Brabham, 2008). Based on a synthesis of research, Brabham describes the four core ingredients as: "an organisation that has a task it needs performed, a community (crowd) that is willing to perform the task voluntarily, an online environment that allows the work to take place and the community to interact with the organization, and mutual benefit for the organization and the community." (Brabham, 2013) This perspective draws a boundary between Crowdsourcing and related phenomena such as open source or commons-based peer production. A similar distinction is made in the taxonomy proposed in (K. Mao et al., 2017) which differentiates *bespoke* (active) Crowdsourcing from *repurposed* (passive) Crowdsourcing.

Crowdsourcing in Software Engineering has been an active field of research. (Rick Kazman and H.-M. Chen, 2009) envisions the Metropolis model, a "new system-development model" that is "more appropriate for the service-dominant, crowd-sourced world", providing a mental framework explaining characteristics of crowdsourced-systems, realms of roles of stakeholders and principles. Theoretical groundwork in that field was presented in (Latoza and Hoek, 2016) which identifies eight *dimensions of Crowdsourcing in software engineering* that allow defining specific Crowdsourcing models such as *peer production*, *competition* and *microtasking*. Crowdsourcing for software creation based on a platform for crowd-supported creation of composite Web Applications combining passive and active Crowdsourcing has been proposed in (Nebeling, Leone, et al., 2012). In the field of HCI (Human-computer interaction), Crowdsourcing has been applied for solving small UI design problems at a large leveraging diversity of microtasking results in *CrowdDesign* (Weidema et al., 2016) and to adapt Web layouts to a variety of different screen sizes in *CrowdAdapt* (Nebeling, Speicher, et al., 2013). (Stol and Fitzgerald, 2014) presents the perspective of enterprise customers and proposes the application of Crowdsourcing to complement automated testing, arguing that quality is one of the main concerns and thus Crowdsourcing is better suited for self-contained areas without interdependencies. Applying collaborative Crowdsourcing, (Satzger et al., 2014) envisions a distributed software engineering model where the software architect sub-divides problems into Crowdsourcing tasks which are further divided and solved by the crowd through collaboration mechanisms. A comprehensive overview of Crowdsourcing in software engineering is presented in (K. Mao et al., 2017), showing an increasing interest from the (forward) software engineering community.

Crowdsourcing in Reverse Engineering in contrast, has not seen much interest. (Saxe et al., 2014) presents *CrowdSource*, solution for malware classification in software binaries. The approach employs a statistical natural language processing (NLP) system which assesses the printable strings in those binaries. The classification is done based on a corpus that was aggregated from posts in the StackExchange Q&A website. Therefore, it employs a passive approach of repurposed Crowdsourcing, as it makes use of publicly available content (the questions, answers, and comments) created by a crowd (the users of StackExchange). Among the approaches discussed in section 3.2, none considers the application of Crowdsourcing, neither bespoke nor repurposed. Although reverse engineering does not benefit from the potential diversity of alternative solutions, it can still 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 derived from RO2 in section 5.2.1, section 5.3.1.4 argues in favor of the application of Crowdsourcing in reverse engineering by reformulating the problem of Concept

Assignment as classification problem and analyzing similarity to the established microtasking model in eight dimensions.

5.3 Method and Tool

This section presents the AWSM:RE method and tools from the AWSM platform which support identification and management of existing knowledge in legacy source code with limited resources and lack of Web Engineering expertise.

5.3.1 Conceptual Model

This section describes the conceptual model of the AWSM:RE method addressing the four aspects of knowledge extraction process, knowledge representation, integration, and crowdsourced reverse engineering.

5.3.1.1 Knowledge Extraction Process

The AWSM:RE method is based on Concept Assignment as introduced above. This allows to identify and secure knowledge implicitly represented by the Legacy System. The extraction mechanism is based on the Legacy System and source code knowledge model formalisms introduced in section 4.3.2. Concept Assignment takes the codebase B of Legacy System $\mathfrak{L}$ 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 extraction process presented in fig. 5.2 is the conceptual model of this function, which can be carried out by different actors.

Artifacts consumed and produced by the extraction process are:

- Artifacts in the Codebase ($f \in B$ in the SCKM)
- Code segments ($s \in f$ in the SCKM)
- Concepts (cf. Definition 5)
- Assignments (expressed as annotations a in eq. (4.6))
- Internal and external representations f' of artifacts $f \in B$

Roles involved in this process are:

- Annotator
- Annotation Platform
- Knowledge Base
- Reviewer
- Representation
- Management
- Operator

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 4.2), by a system actor, i.e. a static or dynamic analysis approach as described in section 5.2.2 or by the crowd. The Annotation Platform is a system role that enables the Annotator to create annotations by representing the codebase B. The Knowledgebase is a system role equivalent to

$\mathbb{K}_B$ which is responsible for storing the annotations created by the Annotator. The Reviewer is an optional human role required to verify the results of system or crowd Annotators. Reviewer is a sub-class of migration engineer. The Representation is the system role responsible for providing access to the Knowledgebase for human and system stakeholders in subsequent reengineering or transformation activities and migration management. Management is a human role that oversees the extraction process (cf. table 4.2). 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.

Process

The conceptual knowledge extraction process model consists of three processes: *Setup*, *Concept Assignment* and *Management and Usage*. 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.

Setup Process

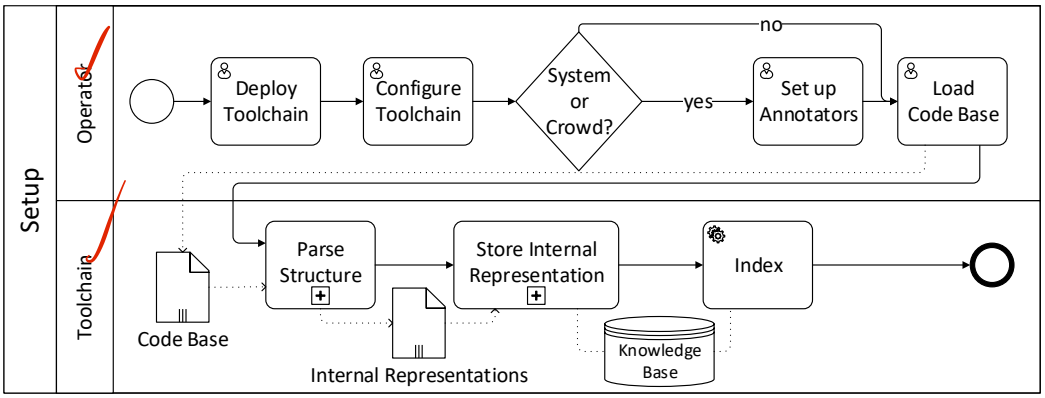


Figure 5.1.: Setup Process

As shown in fig. 5.1, the operator sets up the toolchain, i.e. Annotation Platform, Knowledge Base, Representation and Annotators if realized as system actors, and loads the codebase B in the Annotation platform. By parsing the structure of B, the Annotation Platform identifies the Legacy System's textual artifacts from B and D according to the SCKM, creates internal representations and performs indexing on the contents. Details on the parsing and internal representation are described in section 5.3.2.1

Concept Assignment Process

As shown in fig. 5.2, the Annotator requests an artifact 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 hyperspace (S.R. Tilley et al., 1996) of textual artifacts in B and D according to the parsed structure. It

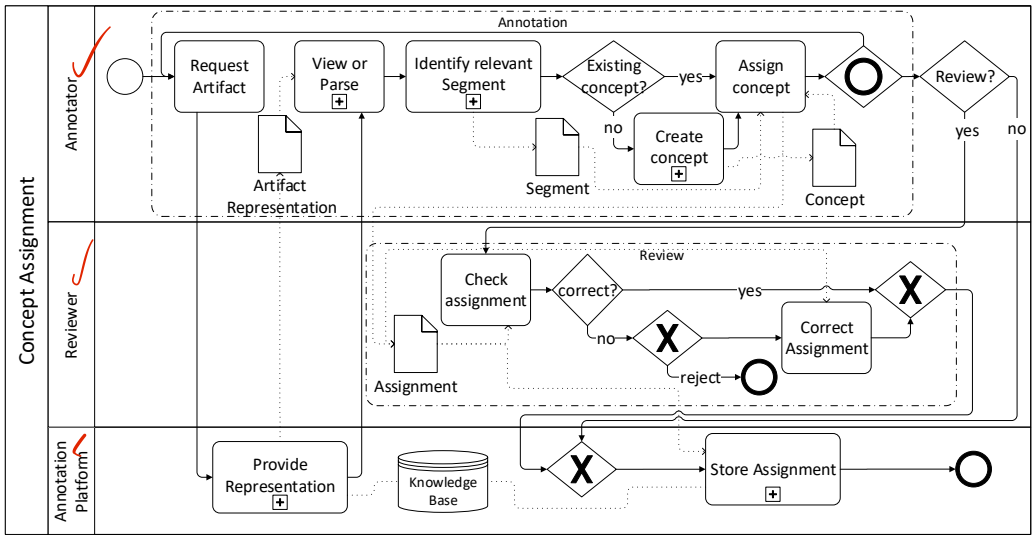


Figure 5.2.: Concept Assignment Process

supports filtering based on the internal organization of the code (according to KDM *Module* subtypes like *Package*, *CodeAssembly*). 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 (cf. cognitive factors (S.R. Tilley et al., 1996)). The Annotator views the artifact and performs the reverse engineering activities of *extraction* and *abstraction* (S.R. Tilley et al., 1996) realized as Concept Assignment: he identifies a relevant segment of the artifact and assigns an existing or newly created concept to it (Biggerstaff et al., 1993). Integration aspects of the Concept Assignment subprocess are described in section 5.3.1.3. 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 5.3.1.4. 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.

Management and Usage Process

As shown in fig. 5.3, the Representation layer supports the Management 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 (Razavian, Nguyen, et al., 2010). It also supports subsequent transformation or reengineering methods at different degrees of human involvement and model-driven adoption

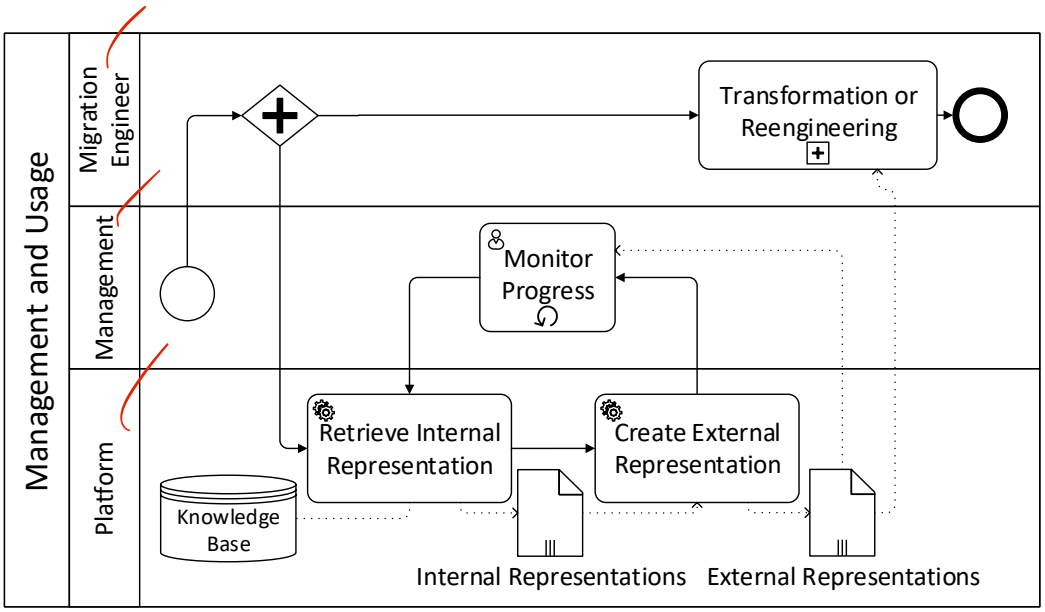


Figure 5.3.: Management and Usage Process

(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 5.3.1.2.

In this way, the knowledge extraction process defines a Concept Assignment-based reverse engineering technique to identify valuable knowledge (Gold, Harman, et al., 2005) in the Legacy System $\mathfrak{L}$ supporting the required identification of relevant components (G. Lewis, Morris, Smith, et al., 2008), elicitation of architectural knowledge (Razavian, Nguyen, et al., 2010) and improving decomposability (Canfora et al., 2000).

5.3.1.2. Knowledge Representation

This section presents the conceptual model of the external representations in fig. 5.3. Research on representations in reverse engineering is focused on visual models, in particular on UML and graphical metrics representations (CanforaHarman and Di Penta, 2007). As defined in the Management and Usage process, different external representations for different stakeholders are required. Management stakeholders require visual aggregations to monitor progress. Migration engineers require representations suitable for 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). Even within one approach, a huge variety can be found. For instance, SAPIENSA (Razavian, Nguyen, et al., 2010) elicits architectural knowledge, problem-related knowledge (business process and rules),

non-functional requirements (quality attributes such as performance) and solution-related knowledge (structural design, design decisions, discarded alternatives). To enable "an effective knowledge capturing and sharing, the knowledge extracted in this phase should be explicitly represented" (Razavian, Nguyen, et al., 2010).

This requires a generic and extensible conceptual model allowing to associate arbitrary knowledge models with legacy codebases. The SCKM is the basis for this model. A key factor is the ability the query the knowledge base using a "powerful query language" (S.R. Tilley et al., 1996), similar to (Paul and Prakash, 1996; Mendelzon and Sametinger, 1995), but at a higher level of abstraction to support the "extract-abstract-present paradigm" (Khadka, Reijnders, et al., 2011). For model-driven approaches, there is a wide variety of metamodels, e.g. MOF, KM3, and transformation languages, e.g. QVT, ATL, that include aspects of querying (Fuhr et al., 2013). Following principle P1, the representation and the query language should be based on open Web standards.

The SCKM is based on KDM. (Pérez-Castillo, De Guzmán, et al., 2011) argues that KDM can be considered an *ontology of Legacy Systems*. Thus, AWSM:RE provides an *ontology of knowledge in Legacy Systems* using the *Web Ontology Language (OWL)*[66]. This allows for a generic knowledge representation that can be extended and integrated with knowledge from external linked open data (LOD) sources, semantic , and reasoning, using SPARQL[67] 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*. It enables exploration through navigation in the legacy knowledge hyperspace implemented as hypertext-based navigation (S.R. Tilley et al., 1996). Realization of the SCKM as OWL Ontology extends KDM for the wider scope of AWSM:RE as data model that supports abstraction (S.R. Tilley et al., 1996), similar to EKDM in REMICS that, in spite of KDM's XMI schema consisting of several XML schemas, was required for use in industrial contexts due to "poor maturity" (Barbier, Deltombe, et al., 2013).

Implementation of external representations for Management stakeholders is discussed in section 5.3.2.1, external representations of the SCKM as OWL ontology and the SPARQL endpoint are discussed in section 5.3.2.3.

5.3.1.3. Integration

This subsection addresses integration aspects of the AWSM:RE method on a conceptual level.

[66] https://www.w3.org/TR/owl-overview/
[67] https://www.w3.org/TR/sparql11-overview/

Integration with ongoing Development

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 (cf. RO1). Thus, reverse engineering as part of Web Migration needs to be integrated, not only the process level in terms of activities, but also for artifacts that drive the development. AWSM:RE adapts the idea of *longterm migration* (Borchers and Hildebrand, 1998) to only migrate those parts of a Legacy System that require maintenance anyway when they are changed for maintenance to reverse engineering. This integration of reverse engineering activities with forward engineering is called *Continuous Reverse Engineering* (CanforaHarman and Di Penta, 2007). 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 (S.R. Tilley et al., 1996) for creating a *mental representation* of the source code (Pennington, 1987) in forward engineering is re-used. 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 (CanforaHarman and Di Penta, 2007). The resulting continuity of reverse engineering improves consistency of code and the knowledge base. On artifacts level, integration can be achieved by associating reverse engineering artifacts, i.e. a set of annotations $A = \{a_1, a_2, \ldots, a_n\}$ with forward development process artifacts, i.e. a set of stories $S = \{s_1, s_2, \ldots, s_n\}$ in a backlog. In this way, they form a *migration package* (Heil and Gaedke, 2016; Gipp and Winter, 2007; Zillmann et al., 2011) that feeds a *migration backlog* (Heil and Gaedke, 2016) from which stories can be integrated into planning of iterations, e.g. into sprint backlogs, achieving integration with agile planning at artifacts level. To support migration engineer stakeholders, integration into editors/IDEs is required. To support management stakeholders, integration into software project management platforms is required. Section 5.3.2.2 describes implementation aspects of the conceptual integration aspects discussed in this section.

Integration with existing web migration methods

This section briefly outlines integration of AWSM:RE with existing Web Migration methods as required by principle P2. AWSM:RE can be easily integrated with the tool-supported model-driven methodology of *REMICS*. It belongs to the recover activity (Barbier, Henry, et al., 2013; Barbier, Deltombe, et al., 2013), where the parser-based static analysis tools and user-interaction-trace-based dynamic analysis tools take the role of Annotator in AWSM:RE. The analysis results in EKDM and ReDSeeDS RSL are represented as concepts and integrated particularly well with the KDM-based SCKM of AWSM. The tool-centric focus of REMICS based on the Eclipse environment makes Eclipse the predestined target of IDE integration. With REMICS'

agile extension (Krasteva et al., 2013) being the only approach addressing agile development based on Scrum, AWSM:RE agile integration on process and artifacts level into REMICS *modernization sprints* can be easily achieved. Integration with *ARTIST* (Menychtas, Konstanteli, et al., 2014) can be achieved in the Migration phase of the ARTIST process by feeding the results of MoDisco, taking the role of Annotator, into the knowledge base. ARTISTs Taxonomy of legacy artefacts (Brunelière, Cánovas, et al., 2013) has significant overlap with the types defined in the SCKM, so the results of ARTIST MDRE can be represented in AWSM:RE without significant changes. The AWSM Knowledgebase would then implement the role of the ARTIST repository enabling reuse and management of discovered models and providing a web-based interface. The distinction of knowledge types in SCKM has significant overlap with the taxonomy of legacy artefacts of ARTIST (Brunelière, Cánovas, et al., 2013), providing good compatibility of the underlying semantic layers. Similar to REMICS, integration with ARTIST's Eclipse-based MPT (Methodology Process Tool) (Konstanteli et al., 2015) would implement AWSM:RE's IDE integration. The migration packages described above enable a *feature-driven incremental migration* through marking of features to be migrated as proposed by ARTIST (Menychtas, Konstanteli, et al., 2014). AWSM:RE can be integrated with the methodological *UWA/UWAT+* approach which is based on conceptual modelling (Distante, Scott Tilley, and Canfora, 2006; Distante, Canfora, et al., 2006). Owing to the generic conceptual model of the SCKM, the Concept Assignment process as described in fig. 5.2 can be considered as concrete implementation of the UWA/UWAT+ reverse engineering phase, abstracting and formalizing knowledge from the Legacy System compliant to UWA meta-models. The AWSM platform would thus improve UWA/UWAT+ by addressing its missing tool support for the reverse engineering phase. Furthermore, AWSM:RE Concept Assignment can be used to distinguish business-process-specific from general helper classes following the *SOAMIG* method described in (Fuhr et al., 2013) and to provide human validation of dynamic analysis results, to identify relevant and non-relevant legacy components (G. Lewis, Morris, Smith, et al., 2008) in *SMART*, for distinguishing Entity, Task and Utility services in Marchetto2008 (Marchetto and Ricca, 2008), and for architectural knowledge elicitation in *SAPIENSA* (Razavian, Nguyen, et al., 2010). AWSM:RE can also be used to improve *decomposability* by annotating separate aspects of UI, business logic and the data layer, which is a vital pre-requisite for migration (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Brodie and Stonebraker, 1995; Canfora et al., 2000). To identify concrete integration points, a Web Migration approach can be mapped onto the *ReMiP* reference model (Harry M Sneed et al., 2010b; Gipp and Winter, 2007) where AWSM:RE belongs to the *Conceptualization and Design* phase and the *Legacy Analysis* core discipline.

5.3.1.4. Crowdsourced Reverse Engineering

This section specifies the Concept Assignment process in fig. 5.2 for Crowd Annotators. As outlined in section 5.3.1.1, the Annotator role can be impersonated by ISV staff, an automated system, or the crowd. To address the limited resources constraint in RO2, 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 in general (CanforaHarman and Di Penta, 2007) and Concept Assignment in particular (Biggerstaff et al., 1993) often leads to low precision or recall (CanforaHarman and Di Penta, 2007) for complex information systems and value-added knowledge discovery of concepts requires human experts (Pérez-Castillo, De Guzmán, et al., 2011).

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) (Deerwester et al., 1990) with 200 features combined with Multi-class Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel (Scholkopf et al., 1997) (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 (C. Lee and G. G. Lee, 2006) 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 (Biggerstaff et al., 1993) becomes identification of relevant code segments s and selecting a class $c \in \hat{C}$ from a given set of classes. After bootstrapping, equivalent to the model step in the canonical model-extract-abstract reverse engineering process (S.R. Tilley et al., 1996), through human or system Annotators, the reformulated classification problem can be solved using Crowdsourcing.

Based on the eight foundational and orthogonal dimensions of Crowdsourcing for software engineering introduced by Latoza et al. (Latoza and Hoek, 2016) – crowd size, task length, expertise demands, locus of control, incentives, task interdependence, task context, and replication – we characterized the AWSM:RE Concept Assignment process for Crowd Annotators as shown in fig. 5.4. The resulting localization in the Crowdsourcing space placed AWSM:RE very close to the established

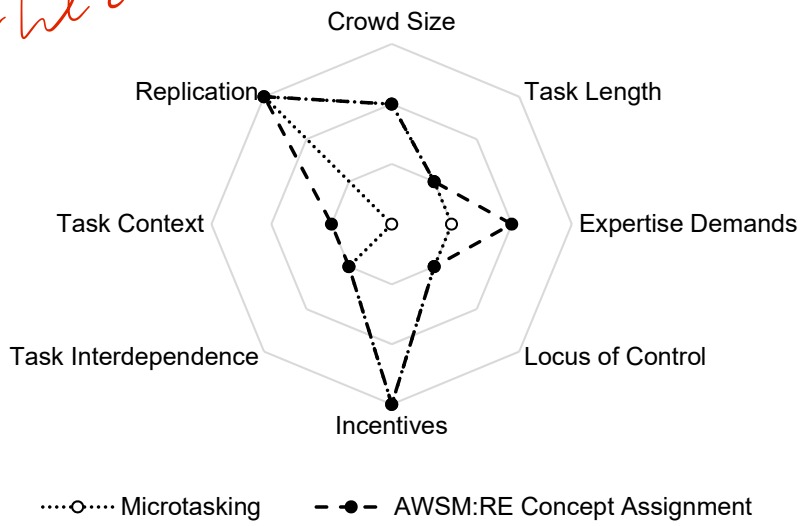


Figure 5.4.: Crowd-based Concept Assignment compared to Microtasking (Heil, Förster, et al., 2018)

and successful *microtasking* 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 software testing (Latoza and Hoek, 2016), 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 characteristics.

TODO:Fragment vs segment

For the CSRE extension (Heil, Siegert, et al., 2019) of AWSM:RE Concept Assignment, *CS Platform* is introduced as system role representing a suitable Crowdsourcing marketplace to post an open classification call to a crowd. The adapted process for crowd annotators in fig. 5.5 addresses three main *challenges of the application of Crowdsourcing in reverse engineering* (Heil, Siegert, et al., 2019):

1. Automatic Extraction and Creation of Crowdsourcing Tasks from B
2. Balancing Controlled Disclosure of Proprietary Source Code with Readability
3. Aggregation of Results and Quality Control

Section 5.3.2.4 outlines solutions to address these challenges by detailing the collapsed subprocesses in fig. 5.5. The general process is as follows: the migration engineer (1) sets the scope of code to be classified using the filtering capabilities of the Annotation platform described above. The annotation platform (2) automatically extracts code fragments for classification, these are (3) pre-processed to

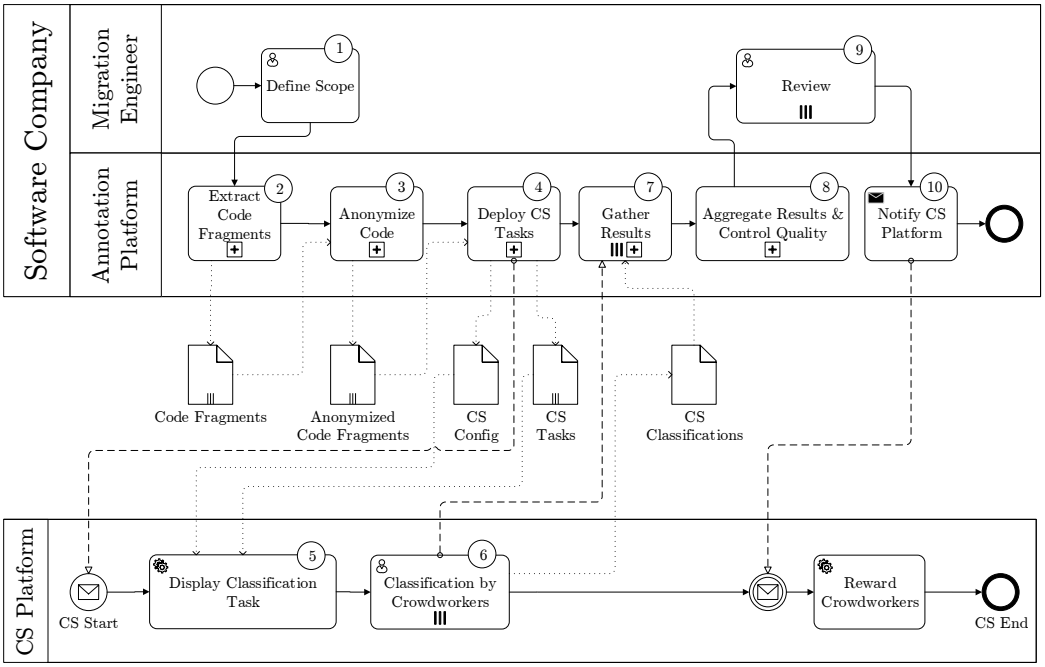


Figure 5.5.: Crowdsourcing-based classification process (adapted from Heil, Siegert, et al., 2019)

achieve the intended anonymization properties, and the annotation platform (4) deploys classification tasks in the CS Platform. 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 (5) presented a textual description of the available categories for classification, according to the ontology in section 5.3.1.2. The interactive external representation of the code fragment to be classified allows the crowdworker to enter his classification (6). (7) Results are gathered per crowdworker and (8) the classification results are aggregated across the different crowdworkers, and quality control measures are applied. (9) The pre-filtered results are forwarded to the review subprocess in fig. 5.2 and (10) the annotation platform notifies the CS platform to reward the participating crowd workers according to the reward policy. Section 5.3.2.4 describes implementation of this process.

5.3.2 Implementation

According to IEEE, “tools are key for reverse engineering and related tasks such as redocumentation and design recovery” (IEEE Computer Society, 2014). Thus, this section describes the implementation of the AWSM:RE method presenting the parts of the AWSM Platform that provide supporting infrastructure for the extraction, knowledge representation and integration as well as the realization of CSRE-based Concept Assignment.

5.3.2.1. Annotation Platform

"Reverse engineering tools assist the process by working backwards from an existing product to create artifacts such as specification and design descriptions, which can then be transformed to generate a new product from an old one." (IEEE Computer Society, 2014) The AWSM Annotation Platform (AWSMAP) is the reverse engineering tool assisting the AWSM:RE knowledge extraction process by supporting the Concept Assignment on B. It supports Concept Assignment for the three types of Annotators introduced above and manages the extracted knowledge, thus implementing Knowledge Base $\mathbb{K}_B$. Figure 5.6 shows its architecture. It provides two interfaces over the Knowledge Base: an API for supporting system users in annotation and subsequent transformation or reengineering, and a user interface for supporting human users in the different roles defined above. Integration with the existing management and development environment of the ISV is described in section 5.3.2.2, the queryable external representation layer is described in section 5.3.2.3. The AWSMAP was prototypically implemented using C# .NET 4.5 MVC 5.2.

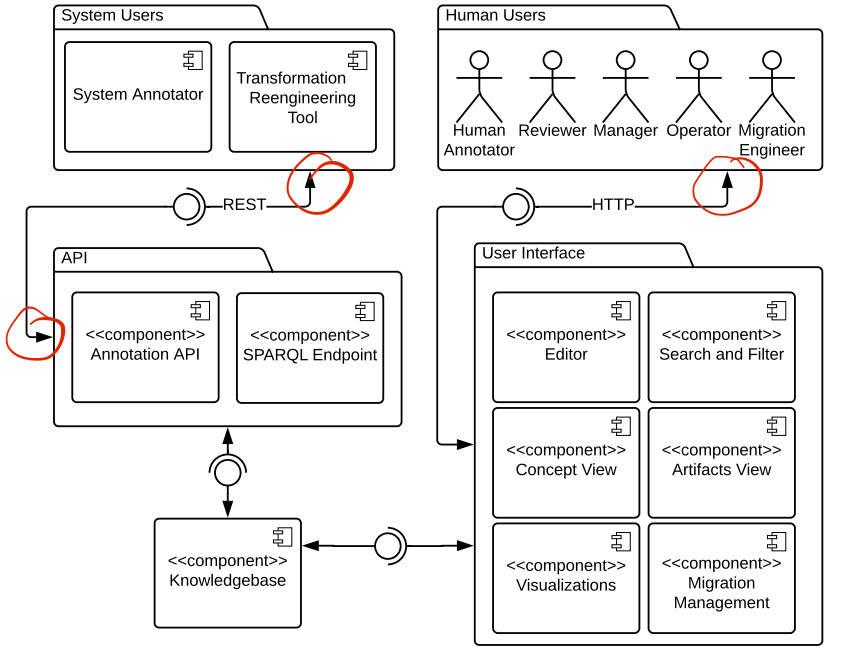


Figure 5.6.: AWSMAP Architecture Component Diagram

The web-based user interface supports migration engineers to perform Concept Assignment. 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 $f \in B$. Artifacts are represented with syntax highlighting ace.js[68]. As shown in fig. 5.7, existing annotations are displayed as overlays. New annotations can be created by selecting a segment of code s 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. As seen in fig. 5.8 contains a name, type and description text. Additionally, attachments can be uploaded, e.g. UML or BPMN diagrams or other artifacts specific for subsequent processing (cf. P2 and P3). Locations $l = (s, f)$ of the concept in B are listed on the right side with the links leading to the corresponding segment s in artifact f.

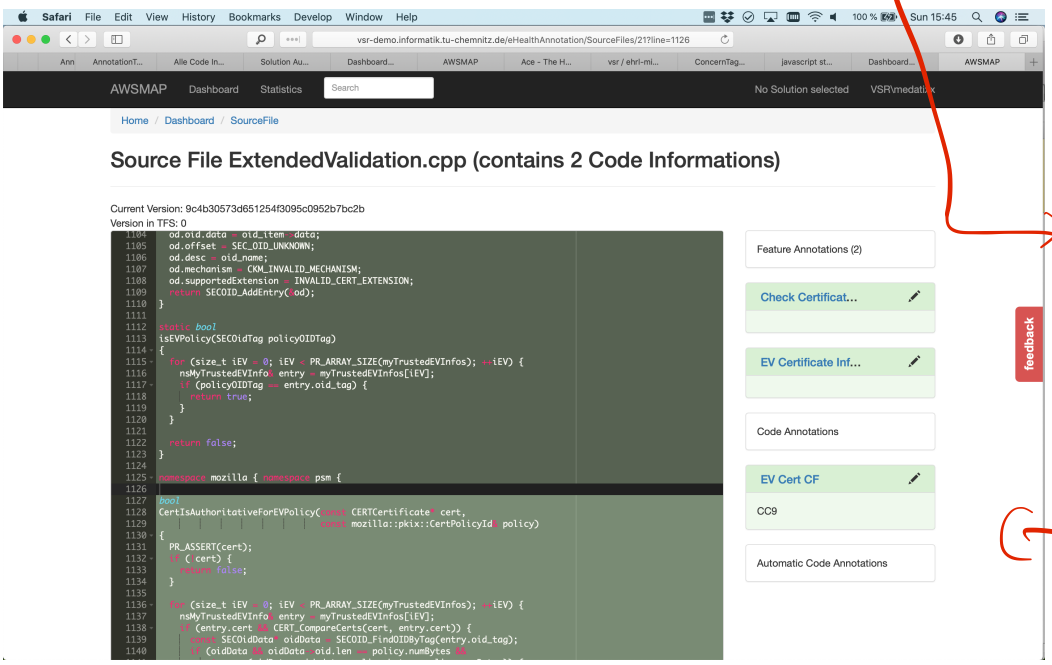


Figure 5.7.: Annotation Platform Editor Screenshot

ConcernTagger (Eaddy et al., 2008) is a conceptually similar tool, but it does not provide a web-based user interface and is a separate tool in contrast to the integrated AWSMAP platform.

One of the core concepts of AWSMAP derived from principle P1 is referencability of all artifacts and knowledge through URLs. This implements hypertext-based navigation of the legacy information hyperspace (S.R. Tilley et al., 1996) and fosters integration by making knowledge linkable in internal email or chat communications, intranet wikis, blogs, etc.

For Management stakeholders, AWSMAP provides a user interface with different statistics to monitor progress and visualizations of the concept structure. Figure 5.9

[68] https://ace.c9.io/

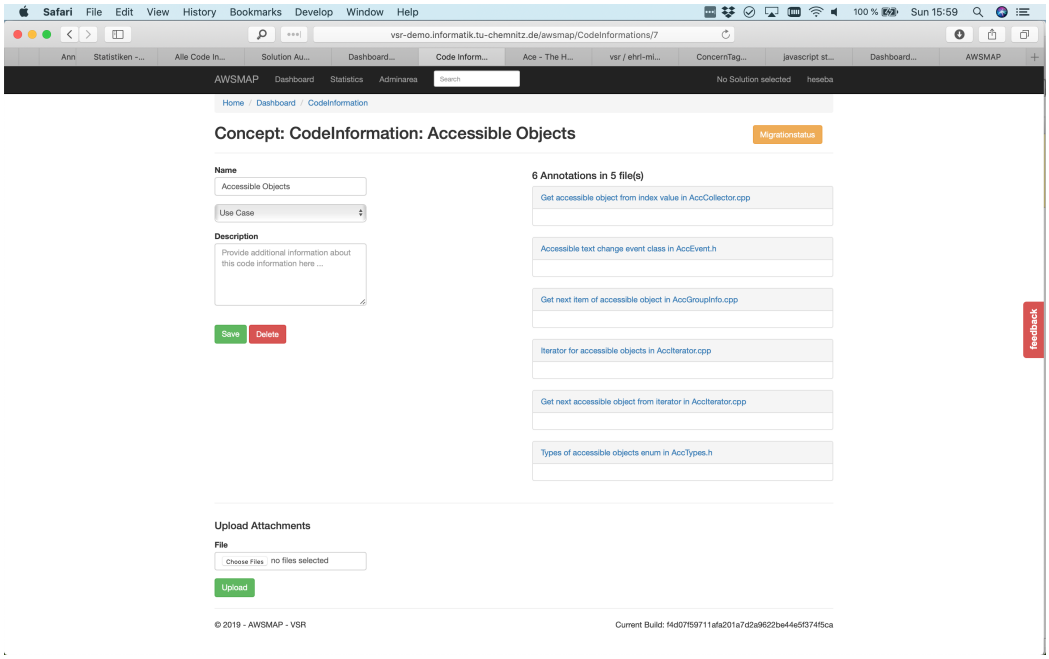


Figure 5.8.: Annotation Platform Concept View

shows Annotator statistics. Figure 5.10 shows a Sankey Diagram of artifact (left side) feature (right side) interlacing, other available visualizations are Force Graphs and Chord diagrams.

5.3.2.2. Integration

This subsection addresses integration aspects of AWSMAP as shown in fig. 5.11. Software engineers of ISVs create and manage artifacts of a software system using an environment of development tools. To support creation, typically *Integrated Development Environments (IDEs)* are used. Reverse engineering tools should be integrated into development environments to facilitate adoption by migration engineers instead of being built as separate tools (Müller et al., 2000). Thus, AWSMAP supports integration into IDEs via its RESTful API. The implementation in the context of section 2.1 is realized as Visual Studio Extension. 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 5.12 shows the creation of a new annotation in the Visual Studio Extension.

For typical ISVs, the artifacts of the Legacy System $\mathfrak{L}$ are under version control using a *Version Control System (VCS)*. AWSMAP supports VCS-integration by implementing the "load codebase" step in [@#fig:awsm.re.setup] as import from an existing VCS repository. In the context of section 2.1, VCS integration was implemented using the Team Foundation Server (TFS) REST API[69]. The operator specifies the TFS instance configuration and repository URL in the AWSMAP web.config in the "configure

[69] https://docs.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-5.0

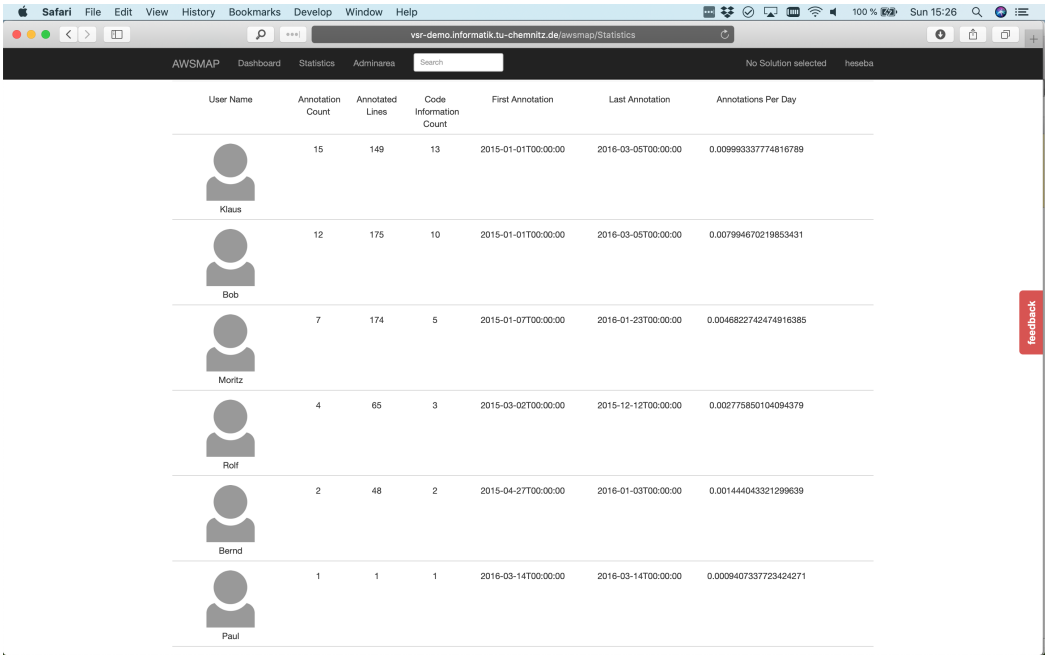


Figure 5.9.: Annotation Platform Statistics

toolchain" step of [@#fig:awsm.re.setup] and can then trigger import and parsing of the codebase from TFS VCS via the Admin Dashboard.

For tool 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 5.3.1.3 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 5.13 shows a screenshot of the definition of a migration package in AWSMAP.

The last integration aspect is *authentication and authorization* of users. ISVs typically operate an Identity Provider to restrict access to their company-internal resources. In the context of section 2.1, *Active Directory (AD)*[70] is used, thus AWSMAP integrates with AD for federated authentication.

5.3.2.3. Queryable External Representation

As introduced above, the external representation of the SCKM is achieved through an *ontology of knowledge in Legacy Systems*. Following principle P1, the representation is based on open Web standards: for describing the ontology OWL and SWRL[71] is used, querying is supported via a SPARQL endpoint, the OWL classes referring

[70] https://docs.microsoft.com/de-de/windows-server/identity/active-directory-federation-services
[71] https://www.w3.org/Submission/SWRL/

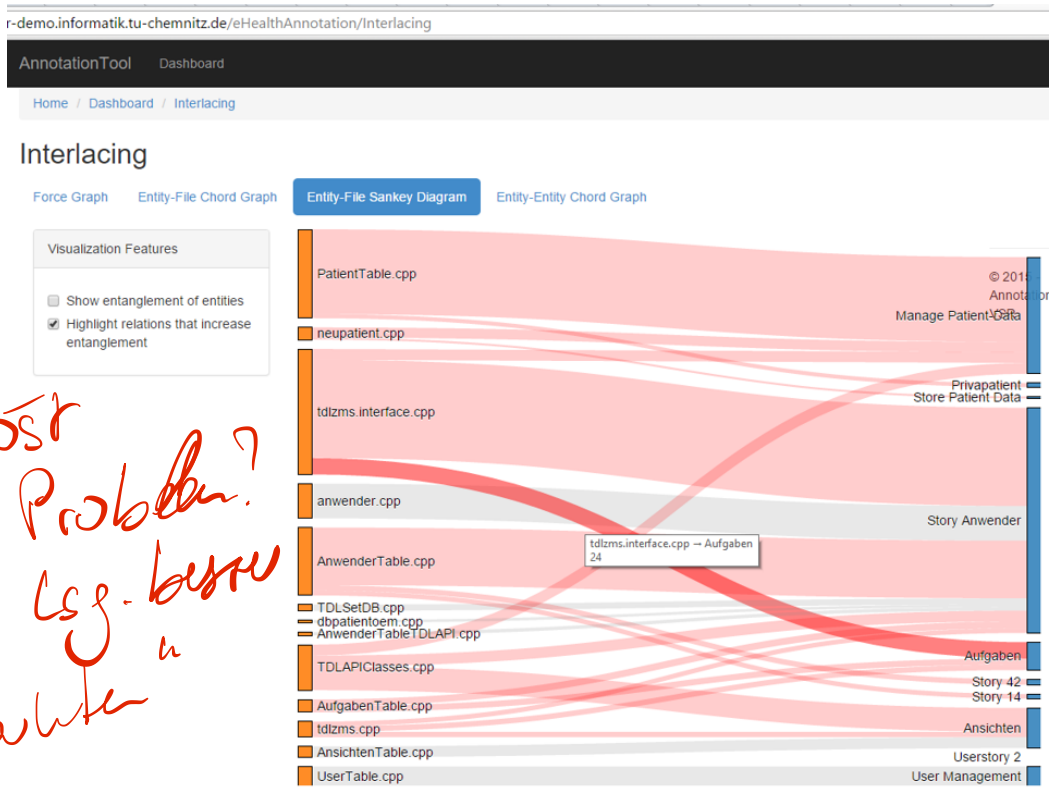


Figure 5.10.: Annotation Platform Visualizations

to Legacy System artifacts are a mapping from KDM to OWL and for modeling annotations the W3C Web Annotation Data Model[72] (OA) is used, to allow for portability and sharing across tools as in (Díaz et al., 2019). The ontology was designed using Protégé[73]. Figure 5.14 shows a UML visualization of the main OWL classes. The two main classes, Feature, and DomainKnowledge and the knowledge types are mappings of the KDM-based SCKM and have been defined in section 4.3.2. The SCKM annotation $a = (k, l)$ is modelled as 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 `Knowledge`. 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 external representation of the Knowledge Base $\mathbb{K}_B$ for system actors is thus a set of RDF[74] 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.

[72] https://www.w3.org/TR/annotation-model/
[73] https://protege.stanford.edu/
[74] https://www.w3.org/RDF/

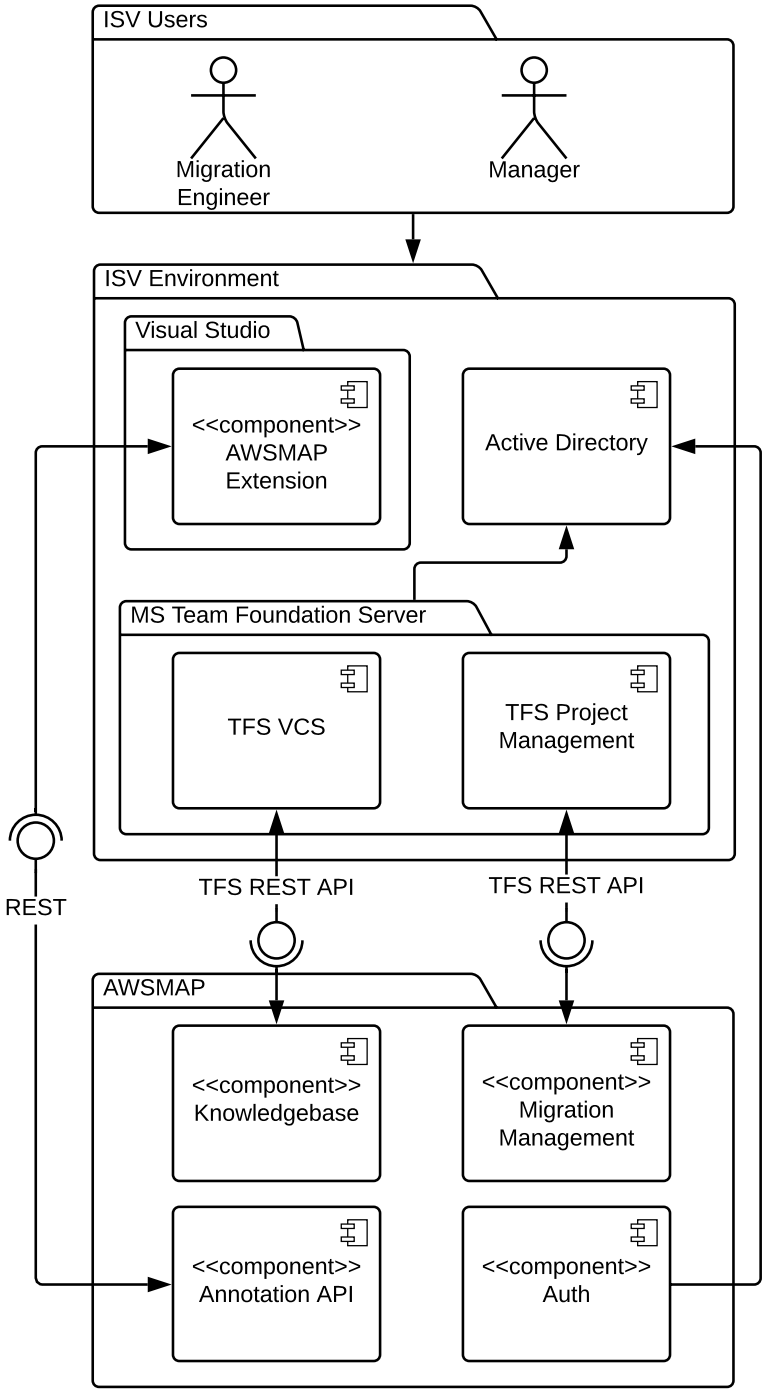


Figure 5.11.: AWSMAP Integration Architecture

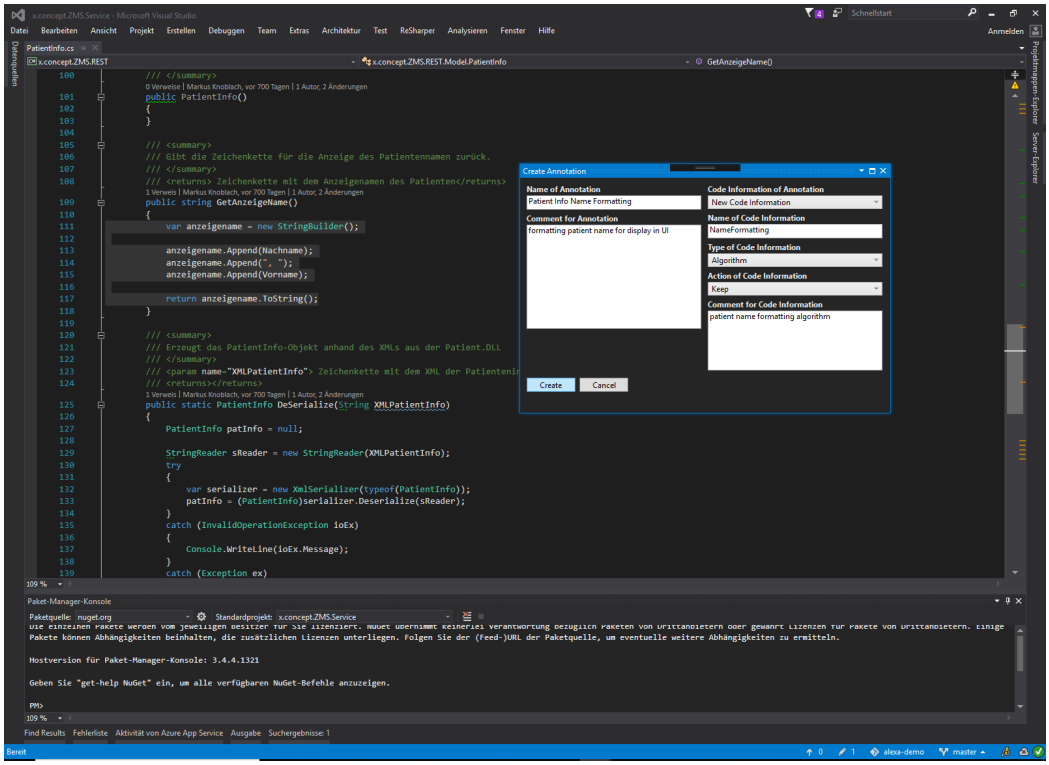


Figure 5.12.: AWSMAP IDE Integration

In particular, reasoning capabilities can support complex use cases. This can be demonstrated with the following example. When querying the Knowledge Base via the AWSMAP SPARQL endpoint, 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 like:

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

Listing 5.1: Partial SPARQL Containment Query

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 GeoSPARQL[75], a query should simply state

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

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

[75] http://www.geosparql.org/

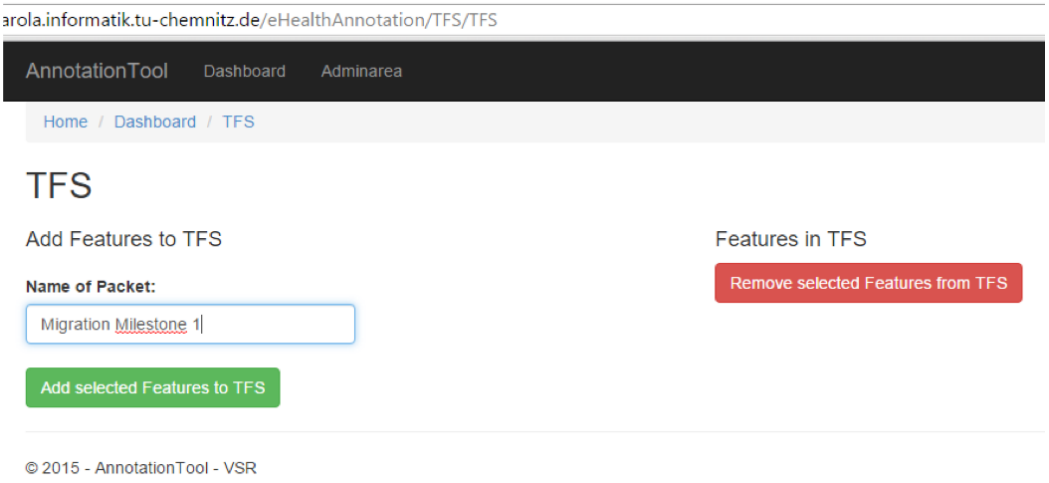


Figure 5.13.: TFS Integration: Migration Packages

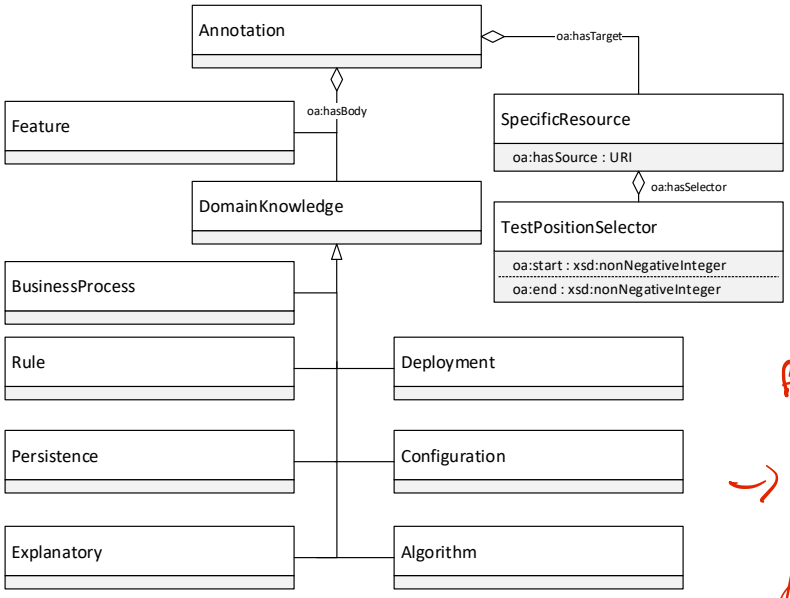


Figure 5.14.: OWL SCKM Ontology Classes

Likewise, complex relationships like knowledge-influences-feature should be interfaced as simple triple patterns. To realize this, the ontology defines additional rules using the *Semantic Web Rule Language (SWRL)*. It allows defining inference rules based on which additional triples are asserted into the knowledge graph. The following shows the SWRL rule for the influences relationship:

```
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:greaterT- hanOrEqual(?ef, ?er) => sckm:influences(?R, ?F
↪ )
```

Listing 5.3: Partial SPARQL Containment Query using sckm:within Property

5.3.2.4. Crowdsourcing

This subsection outlines the implementation of CSRE Concept Assignment by addressing the three challenges introduced in [@#sec:csre]. For experimentation with CSRE Concept Assignment on the bespoke Crowdsourcing platform *microWorkers*[76], AWSMAP was extended with the following features. CSRE Management facilities were added for defining the 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. 5.15 and fig. 5.19. Finally, the features described in the following were implemented to address the three main challenges of CSRE. A more detailed description can be found in (Heil, Förster, et al., 2018; Heil, Siegert, et al., 2019).

Automatic Extraction and Creation of Crowdsourcing Tasks from B

In [@#sec:csre] the 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 used[77] 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 code, and they work entirely automatic. The implementation of step 2 in fig. 5.5 runs Doxygen[78] on B

[76] https://microworkers.com/
[77] cf. https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages
[78] https://www.stack.nl/~dimitri/doxygen/

and parses the generated documentation artifacts to identify relevant code fragments and referenced code.

Balancing Controlled Disclosure of Proprietary Source Code with Readability

The open classification call to a potentially large and unknown group of workers (Latoza and Hoek, 2016) 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* (Ceccato et al., 2014) 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 **??** 5.1 implements step 3 in fig. 5.5 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, 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).

Algorithm 5.1: CSRE Anonymization Algorithm

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

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

Aggregation of Results and Quality Control

Aggregating potentially contradicting results from unknown crowdworkers and ensuring quality is a challenge (Stol and Fitzgerald, 2014; Weidema et al., 2016; Nebeling, Leone, et al., 2012) 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. Thus, quality-control design-time (Worker selection, Effective task preparation) and quality control run-time approaches (*Ground truth, Majority consensus*) are required (Allahbakhsh et al., 2013), constituting the *Quality control and results aggregation properties*. 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* (Allahbakhsh et al., 2013). 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 *compensation policy* is a combination of monetary and non-monetary rewards: quality contributions receive 0.30 USD and a positive rating. The *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.

$$S(w_i) = \frac{|C^+_{w_i}|}{|C^+_{w_i}| + |C^-_{w_i}|} \tag{5.1}$$

Majority Consensus is used 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).

$$V(c_k) = \frac{\sum\limits_{(w,c) \in C | c \in c_k} S(w)}{\sum\limits_{(w,c) \in C} S(w)} \tag{5.2}$$

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

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

For cases where no clear majority can be found, an overview of result distributions (fig. 5.15) and a display of crowdworker rationales (explanations) (fig. 5.19) was implemented.

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 5.15.: CSRE Statistics

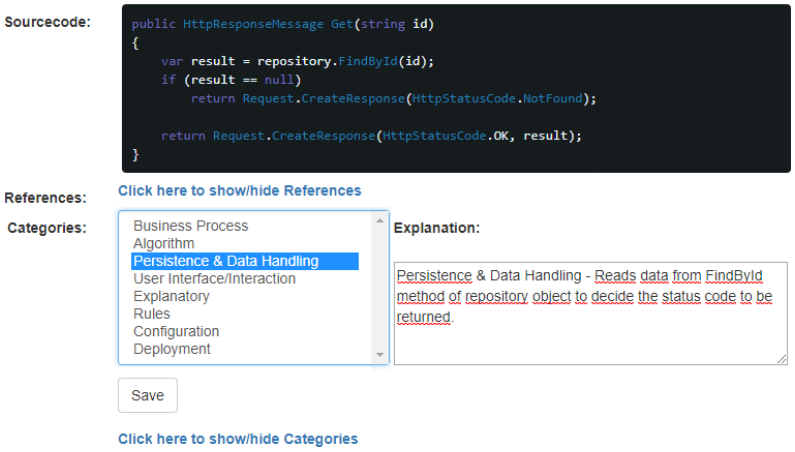


Figure 5.16.: Crowd worker view

5.4 Evaluation

This section evaluates AWSM:RE regarding the requirements in section 5.2.1. Effectiveness and Efficiency are assessed in detail in section 5.4.1.

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 5.4.1.

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 5.4.1.

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, AWSM: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 AWSM:RE. Section 5.3.1.3 presented the conceptual model of integration of knowledge discovery into ongoing development and maintenance activities following the continuous reverse engineering paradigm and section 5.3.2.2 presented implementation of the integration aspects in the AWSM platform through connection of AWSMAP with core components of ISVs' development environment Thus, AWSM:RE meets the integration requirement.

Knowledge Management The knowledge discovered through AWSM:RE is represented based on open Web standards like OWL, RDF and OA as described in section 5.3.1.2 and section 5.3.2.3 and is available for integration with different model-driven or non-model-driven methods as in section 5.3.1.3 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 as described in section 5.3.2.3. Thus, AWSM:RE meets the knowledge management requirement.

5.4.1 Experimental Evaluation of CSRE

As the concept 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.NET[79], 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 $\overline{C}$ was formed from the 8 domain knowledge types of the SCKM ontology fig. 5.14 as concepts to be assigned to the fragments. All 10 fragments were manually classified so that the *test dataset* consisted of 10 classified

[79] http://www.dotnetblogengine.net/

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.

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

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

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))

$$f_e = \frac{|C^-|}{|C|} \tag{5.4}$$

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))

$$E = -\sum_{I=1}^{k} f_i \lg f_i \tag{5.5}$$

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

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

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.

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.

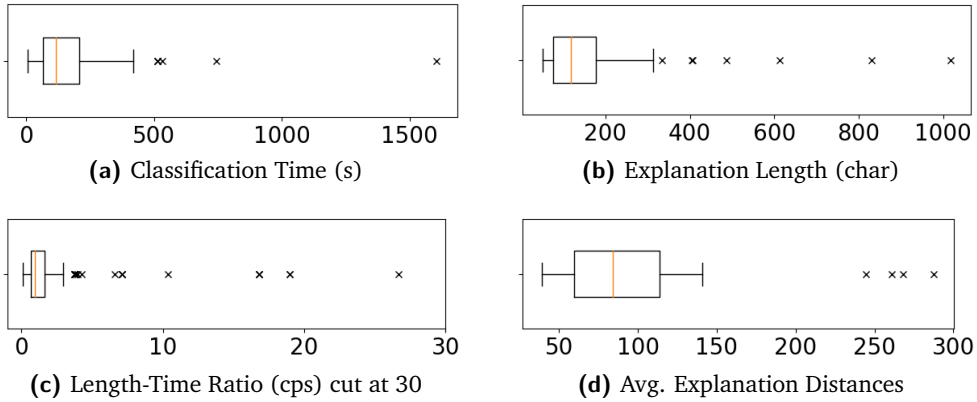
Average of entropy and Herfindahl dispersion measure are $\overline{E} = 0.639$ and $\overline{H^*} = 0.757$, their minima co-occur with minimal error rate, their maxima with the second-highest f_e. There is a significant ($\alpha = 0.05$) positive correlation (Pearson's $\rho = 0.724$, $p = 0.018$) between f_e and E and between f_e and H^* ($\rho = 0.757$, $p = 0.011$), 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.

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 outliers[80] 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. 5.18c belong to *Persistance & Data Handling*, the second most frequently voted, and indicating that the formulation of this concept is not precise enough.

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. 5.17. Time median was $\tilde{t} = 117.5$s, but the observed times varied widely. The interquartile range was at IQR = 143.25s 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. 5.17a 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 use of a record-and-replay script.

Explanation lengths in fig. 5.17b range from 51 to 1017 characters (median $\tilde{d}$=119) and are relatively close (IQR=100.5) to the min. threshold of 50 chars of explanation. Most of the crowdworkers wrote rather short explanations in short time, as visible from the lower quartile of 76.25 chars and outliers for time and length only appearing above Q3. Figure 5.18 shows explanation length and time in relation, with correct classifications in green and wrong in red.

[80] we use $Q3 + 1.5$ IQR as outlier threshold



(a) Classification Time (s)

(b) Explanation Length (char)

(c) Length-Time Ratio (cps) cut at 30

(d) Avg. Explanation Distances

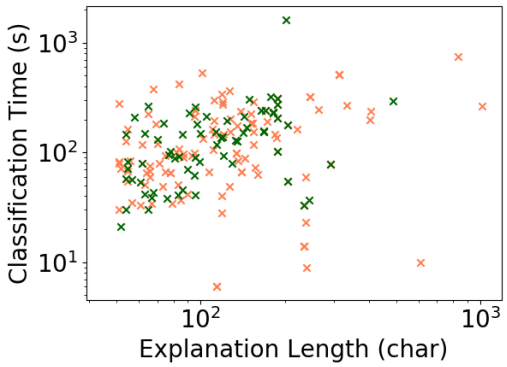
Figure 5.17.: Classification Time and Explanation Length 1-dimensional Distributions (Heil, Siegert, et al., 2019)

Analysis of fake contributions. Cheating is an issue for the quality of Crowdsourcing (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 ratio[81] was calculated. As shown in fig. 5.17c, within a range of 0.13 to 61.2 cps, the majority of values is distributed very closely (IQR 0.96 cps) around 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 competitions[82]. 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.

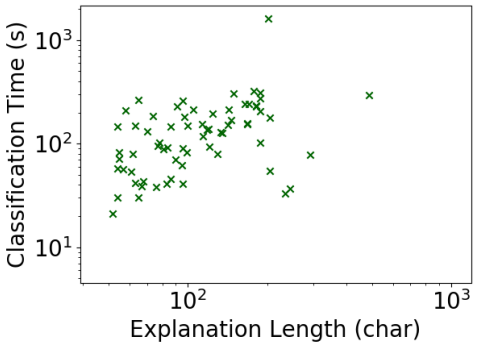
To further identify workers completing the tasks very quickly through copying, average similarity of explanations per user was calculated using pairwise Levenshtein distance. The averages range from 39 to 287 and are concentrated (IQR=54.5) around a median of 83.8 (cf. fig. 5.17d). Identical copies (distance 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

[81] measured in characters per second (cps)
[82] cf. http://www.intersteno.org/

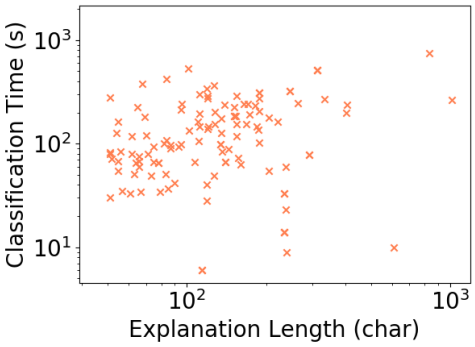
adaptions. Manual inspection of responses showed that 3 of 34 exclusively put code in their explanations, indicating a wrong understanding of the task.



(a) correct and wrong



(b) correct only



(c) wrong only

Figure 5.18.: 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 very interesting, 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 $\overline{C}$ 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 platforms, we would expect an even better quality on a dedicated software development platform such as topcoder[83]. 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

[83] https://www.topcoder.com

Legacy Systems such as Java Swing, C/C++ MFC to be at least similar due to higher popularity[84] 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 – a known and expected characteristic of Crowdsourcing (Allahbakhsh et al., 2013) – 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 5.2.1 are met.

5.5 Summary

This chapter presented the AWSM Reverse Engineering method, which facilitates problem and solution domain knowledge rediscovery for ISVs with limited resources, specifying 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 conceptual model and implementation of AWSM:RE have been described. Experimental evaluation has shown 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.

[84] cf. TIOBE Index https://www.tiobe.com/tiobe-index/

Crowdworker	Erklärung	Benötigte Sekunden
4f6ebfeb	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.	186
	→ Rules	
57a169c1	It is used for presenting data into category after sorting according to ID.	67
	→ Explanatory	
e858273d	This category describes source code, in which general rules or rules of a specific domain are used to check or decide anything.	49
	→ Rules	
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	198
	→ Rules	
0fa041b2	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.	84
	→ Deployment	
81a80188	This code contains deployment commands and the server is hosted for the same purpose.	37
	→ Deployment	
a8ae5daa	because it defines rules of listing the entire method.	126
	→ Rules	
1e386615	the code contains rules to check if statements are true or false	65
	→ Rules	
e0aefe92	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	323
	→ Persistence & Data Handling	
	→ Rules	
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	183
	→ Persistence & Data Handling	
	→ Rules	
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.	14
	→ Algorithm	
	→ Explanatory	
	→ Rules	
71535d20	method, that fills a field of categories with content, so the user can interact afterwards	42
	→ User Interface/Interaction	
dba271d8	The provied 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.	9
	→ Persistence & Data Handling	
Durchschnittszeit		**106**

Figure 5.19.: CSRE Results

Die Tabelle wirkt etwas verloren hier, könnte man besser im Text integrieren

6 AWSM Risk Management Method

This chapter addresses research objective RO3:

> To provide a risk management method, models and tools that demonstrate desirability and feasibility of a potential web-based version of the Legacy System with limited resources and lack of Web Engineering expertise.

Following an analysis of the current situation, identification of requirements and a review of related work, a conceptual model and implementation of the AWSM:RM method consisting of three mechanisms to address RO3 are described. 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 Research Questions

To design and evolve a solution for research objective RO3, this chapter addresses three research questions:

> **AWSM:RM Research Question 1**: 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 2**: How to transfer the Rapid Prototyping paradigm into a Web Migration context?
>
> **AWSM:RM Research Question 2**: How to provide guidance and automation for Rapid Web Migration Prototyping to enable execution with limited Web Engineering and Web Migration expertise?

6.2 Analysis

The detailed problem analysis conducted in section 4.1.2 on the basis of the field research results has revealed *doubts about desirability* and *doubts about feasibility* as two main factors rendering ISVs hesitant to commence a Web Migration. *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., 2010a), 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., 2010a). Examples of this can be seen in the "first migrations" of AWS Migration (Amazon Web Services Inc., 2018), SMART-MP (CMU Software Engineering Institute, 2013) in SMART (G. Lewis, Morris, O'Brien, et al., 2005; G. Lewis, Morris, Smith, et al., 2008) and the experimentation-based feasibility studies in IC4 (Fowley, Elango, et al., 2018).

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* which allow assessing the *plausibility of a web-based version* of the Legacy System and the desirability with regard to *advantages of Web Systems* (cf. also to RIA features in (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012)). 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 and feasibility in early stages, allowing to create a running software prototype as concrete and tangible result (Alavi, 1984). In Web Engineering, especially through widespread use of agile development (Abrahamsson et al., 2002) and transfer of ideas from HCD/Design Thinking (IDEO, 2015) putting more emphasis on customers' perspective and feedback, 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., 2010b) 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.

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.js[85], Rails[86], React[87], Angular[88] 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, WPF[89], Swing[90].

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 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.

6.2.1 Requirements

The following requirements have been derived based on RO3, 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.

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.

6.2.2 Related Work

Prototyping in Software Engineering is an established practice for requirements clarification and validation of understanding of elicited requirements (IEEE Computer Society, 2014; Wood and Kang, 1992). Prototypes can range from paper

[85] https://nodejs.org/
[86] https://rubyonrails.org/
[87] https://reactjs.org/
[88] https://angular.io/
[89] https://msdn.microsoft.com/en-us/library/aa663364.aspx
[90] https://docs.oracle.com/javase/8/docs/technotes/guides/swing

mockups (ISO/IEEE, 2017b) to beta-test versions of software products and exist in different styles, varying from throw-away to evolutionary, depending on the particular prototyping life cycle process (IEEE Computer Society, 2014). Prototyping is a method for risk reduction (ISO/IEEE, 2017b). IEEE defines *Software Prototyping* as follows:

Definition 7 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.

From the software maintenance perspective, Prototyping is an activity of feasibility analysis (Software Engineering Standards Committee of the and IEEE Computer Society, 1998). Software prototypes represent a *functional, demonstrative product*, proving "the relevance of a solution" (ISO/IEEE, 2017b). Prototyping is a method for improving software quality (Wallmüller, 2001) and has been acknowledged to increase the efficiency and effectiveness of software design (Tripp and Bichelmeyer, 1990). *Explorative Prototyping* seeks to integrate user feedback in the design process, *experimental Prototyping* focuses on technical feasibility and *evolutionary Prototyping* incrementally enhances a prototype (Wallmüller, 2001). *Horizontal prototypes* are the result of explorative Prototyping and focus on user interaction, *vertical prototypes* result from experimental Prototyping and implement a small functionality on all application layers (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).

Prototyping is particularly appropriate for unfamiliar situations with limited experience to draw from (Tripp and Bichelmeyer, 1990). In Web Migration, this situation is a consequence of the new target environment, which significantly differs from the legacy desktop application environment and bears risks regarding feasibility and desirability. Prototyping as a methodology has often seen transfer into new domains. As a concept originating in the hardware development domain (automotive, aviation, etc.), it has been transferred to Software Engineering (Wallmüller, 2001). Prototyping has been used for the development of *information systems* for a long time (Wasserman and Shewmake, 1982; Alavi, 1984). An information systems prototype is defined as "an early version of a system that exhibits the essential features of the later operational system", that might "evolve into the actual production systems" or be used "only for experimentation" (Alavi, 1984). Based on interviews with stakeholders from 12 information systems development projects, a

study (Alavi, 1984) highlights the *benefits of Prototyping*: prototypes are "a tangible means of comprehending and evaluating the proposed system", to elicit and clarify requirements, receive feedback and provide a "common reference point for both users and designers by which to identify potential problems and opportunities early in the development process" (Alavi, 1984). Moreover, Prototyping can "cultivate and achieve user participation and commitment to a project" (Alavi, 1984).

Transfer of these potential benefits into the Web Migration domain is the primary motivation of AWSM:RM. The main disadvantage of Software Prototyping is the "danger of users' attention being distracted from the core underlying functionality by cosmetic issues" (IEEE Computer Society, 2014). This problem is not relevant in Web Migration contexts as the underlying functionality is well-known, mature and fixed, resulting from long requirements engineering in the Legacy System as described in section 2.1, whereas the focus on the changes in user interface and user interaction through the Web System target architecture is the main reason for web migration prototype creation. With its intrinsic communication value, Rapid Prototyping has been used in HCI-related fields such as UI and Instructional Design. Rapid Prototyping of *UI Designs* was proposed by the *CrowdDesign* approach (Nebeling, Leone, et al., 2012) already discussed in section 5.2.2. Tripp et al. transfer Rapid Prototyping to *Instructional Design*, i.e. the design of instructional systems (Tripp and Bichelmeyer, 1990), to foster the participation of users in the design process.

Rapid Prototyping in Agile Software Development is widely used, 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 Manifesto[91]. IEEE defines *Rapid Prototyping* as follows:

Definition 8 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

The *spiral model* (Boehm, 1988), which is a precursor influencing many agile development approaches, advocates the use of Rapid Prototyping beginning in the early stages of software development and incremental evolution of prototypes towards an *operational prototype* (cf. operational product ISO/IEEE, 2017b) as central risk management activity. Based on Rapid Application Development (RAD) (Martin, 1991), *Dynamic Systems Development Method (DSDM)* (Stapleton, 1997; DSDM Consortium, 2014) is an agile software development method to systematically apply Rapid Prototyping to address unstable or unknown areas (Highsmith and Cockburn,

[91] https://agilemanifesto.org/

2001). In DSDM, incremental and iterative Prototyping is one of the core techniques to ensure strong involvement of users in ISO 9001 compliant agile software development. The focus on communication aspects of Prototyping in DSDM can also be seen in the fine-grained specification of user roles such as Ambassador and Advisor User. Prototyping in different forms is state-of-the-art practice for large software industry companies like Atlassian (Simpson, 2016), where Prototyping techniques contribute to demo trusts[92] for communication with senior management stakeholders as codified in the team playbook. Rapid Prototyping can also 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).

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) propose the use of prototypes in cloud migration. While the target beneficiary of IC4 is similar to the SME-sized ISV described in section 2.1.1, IC4's use of Prototyping belongs to the experimental Prototyping (Wallmüller, 2001) category. IC4 prototypes 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. virtualised relational database vs. PaaS SQL services vs. NoSQL, etc. To a lesser extent, rapid relocation approaches like 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 3.4 apply. Comprehensive Web Migration methodologies like ARTIST focus on technical feasibility and fail to address benefits and early customer feedback in pre-migration phases (Orue-Echeverria et al., 2014).

The availability of tool support for Software Prototyping plays a crucial role in its success (Tripp and Bichelmeyer, 1990). In our work, we therefore seek to support the rapid web migration process by an open Web standards-based infrastructure.

6.3 Method and Tool

This section presents the AWSM:RM method and tools from the AWSM platform which support rapid creation of a web-based functional demonstrative prototype of the Legacy System in early stages of Web Migration with limited resources and lack of Web Engineering expertise.

[92] https://www.atlassian.com/team-playbook/plays/demo-trust

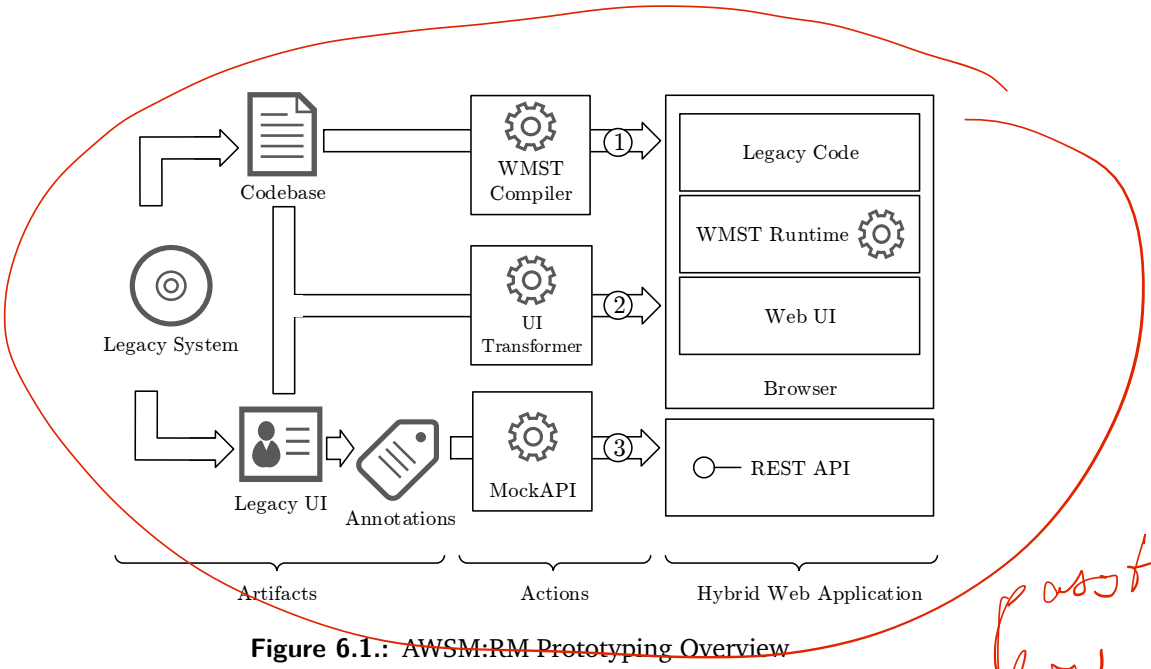


Figure 6.1.: AWSM:RM Prototyping Overview

6.3.1 Conceptual Model

This section describes the conceptual model of the AWSM:RM method addressing the four aspects of Rapid Web Migration Prototyping, business logic re-use, user interface transformation, and integration.

6.3.1.1. Rapid Web Migration Prototyping

AWSM:RM is a Rapid Prototyping method for Web Migration: a *Rapid Web Migration Prototyping* (Heil, Siegert, et al., 2018) method. AWSM:RM prototyping is explorative Prototyping (Wallmüller, 2001) in the Web Migration domain. 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 $\mathfrak{L}$. Thus, re-use on the upper layers is essential in order to achieve the rapidness of prototyping. Figure 6.1 shows an overview of AWSM:RM prototyping.

AWSM:RM is based on three **actions**:

1. Re-use oriented migration of legacy business logic leveraging a suitable *web migration support technology (WMST)* for execution in the Web browser, described in section 6.3.1.2
2. Semi-automatic transformation of the legacy user interface into a Web user interface, described in section 6.3.1.3
3. Rapid Prototyping of a RESTful API through integration with the MockAPI approach, as described in section 6.3.1.4

AWSM:RM consumes and produces the following **artifacts**:

- Legacy System $\mathfrak{L}$ 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 are created as intermediary artifacts as part of action 3

Roles involved in AWSM:RM are:

- Prototyping Engineer
- Annotator
- WMST Compiler
- WMST Runtime
- UI Transformer
- MockAPI

The Prototyping Engineer is the human actor of the AWSM:RM actions. The role is assumed by a migration engineer of the ISV (cf. table 4.2). 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 re-use 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 re-used 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* (IEEE Computer Society, 2014) 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 (cf. evolutionary Prototyping IEEE Computer Society, 2014; Wallmüller, 2001).

6.3.1.2. Re-use of business logic

Re-use is one of the main requirements of this thesis as introduced in section 2.2. In the context of AWSM:RM it is a specific requirement introduced in section 6.2.1 to achieve the rapidness of the Prototyping and to make use of the available expertise of ISV staff. Due to the horizontal nature (Wallmüller, 2001) of AWSM:RM prototypes, the focus is on the re-use of behavioral business logic (BBL) on the higher layers of UI and application logic, similar to the controller layer in an MVC architecture.

Re-use 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 3.4 - or through transformation approaches for the server side. Client-side re-use of business logic, in

contrast, has not received much attention. However, recently a group of technologies which allow for execution of legacy code on the client side, in a Web browser, became available. We refer to these technologies as Web Migration Support Technologies (WMSTs), because they can be used as a basis for client-side migration of legacy code to the Web.

Assessment of Web Migration Support Technologies

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 which 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.: WMST 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, 2018b)

Following the assessment process in fig. 6.2, candidate technologies which had not reached end of support were applied for rapid migration prototyping of the scenario application. The observations provided input for the assessment of the following five parameters:

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

According to our experiment, WebAssembly is the most promising WMST. *WebAssembly (WASM)* (W3C, 2018b) 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

[93] http://www.leaningtech.com/cheerp/
[94] https://developer.chrome.com/native-client
[95] https://developer.mozilla.org/en-US/docs/Plugins/Guide
[96] https://developer.chrome.com/native-client/pepper_stable
[97] https://developer.mozilla.org/en-US/docs/Mozilla/js-ctypes

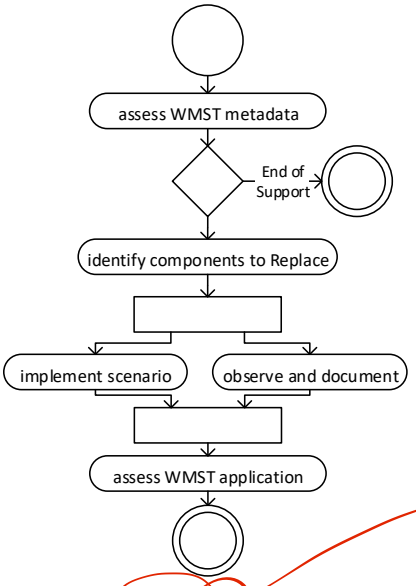


Figure 6.2.: WMST assessment process

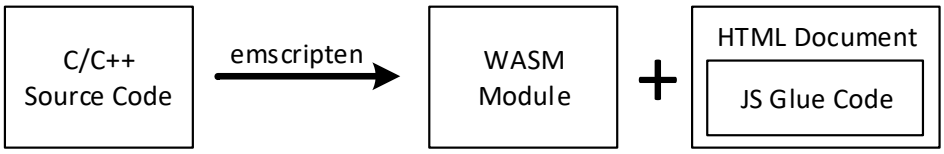


Figure 6.3.: WebAssembly Overview[99]

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. Conversion 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.js[98]. WASM supports dynamic linking for loading DLL dependencies at runtime. As of March 2017, WASM has reached the Cross-browser consensus milestone with preview implementations in all major browsers. 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. Figure 6.3 shows an overview of WASM. Thus, AWSM:RM addresses client-side re-use of business logic in action 1 based on WASM as described in the following.

[98] `http://asmjs.org/spec/latest`
[99] adapted from `https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts`

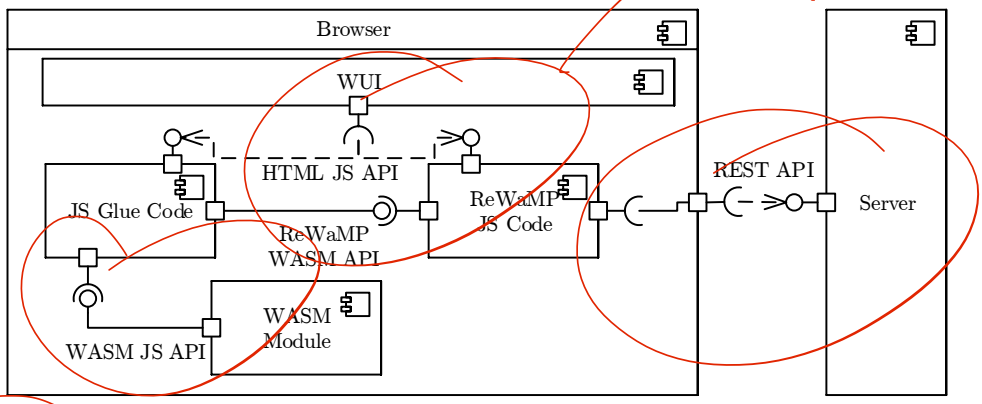


Figure 6.4.: Architecture of ReWaMP Prototypes (adapted from Heil, Siegert, et al., 2018)

ReWaMP: 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 re-use of business logic leveraging WASM. This section provides an overview of the architecture of ReWaMP prototypes, shown in fig. 6.4, and the conceptual process model of ReWaMP, shown in fig. 6.5. The implementation of this process, the supporting toolchain, and the guided Prototyping assistance system are described in section 6.3.2.1.

One ReWaMP prototype is created per user interface $u \in U$. ReWaMP prototypes (Heil, Siegert, et al., 2018) consists of the following components: The *WUI* (Web user interface) 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. For each $u \in U$, 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 for 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.

The ReWaMP process (Heil, Siegert, et al., 2018) in fig. 6.5 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 (cf. section 4.3.2) 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

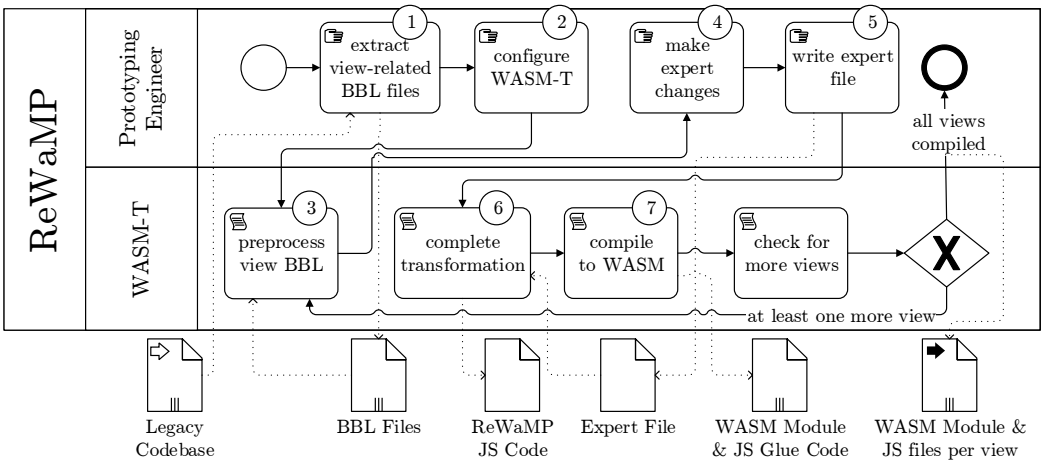


Figure 6.5.: ReWaMP Process (adapted from Heil, Siegert, et al., 2018)

C++ codebases using MFC as in section 2.1.2. The responsibilities of the Prototyping Engineer and WASM-T are represented as separate lanes in fig. 6.5.

The Prototyping Engineer (1) extracts the source files $f \in B_u, B_u \subset B$ containing view-related BBL from the legacy codebase B and (2) provides WASM-T 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 which directly interact with the view u, defining event handlers for UI elements. WASM-T transforms the extracted legacy code to make it compilable with WASM. It uses a semi-automatic process since automated semantic analysis of source code is complex and the BBL in $f_i \in B_u$ can reference dependencies $dep_j \in Dep$, i.e. $\underline{D}_{i,j} = 1$, which are not available for compilation because they are binaries (KDM: *BinaryFile*). In (3), WASM-T pre-processes the BBL 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*[100] 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).

[100] cf. *mock object* (ISO/IEEE, 2017b)

6.3.1.3. User interface transformation

This section presents the conceptual model of the UI Transformer in action 2 of fig. 6.1. As described in section 4.3.2, the layout l_{legacy} of a legacy user interface u is a pixel-based mapping of controls c into two-dimensional cartesian space, representing the euclidean plane. Modern WUIs 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 realised through fluid grids (Marcotte, 2010; Nebeling and Norrie, 2013), defining the WUI as *grid layout* l_{grid} consisting of *rows* and *columns*. Implementations of grid layouts are based on popular frameworks like bootstrap[101] or on the CSS Grid Layout standard (W3C, 2017). Compared to pixel-based layouts, the rows and columns of grid layouts divide the continuous Euclidean plane into discrete *cells*, thus imposing 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:bounding-box] of a control c in l_{grid} reflects these grid constraints as follows

$$\begin{aligned} x &\in \left\{ i \cdot \frac{w_{viewport}}{n_{grid}} \;\middle|\; 0 \leq i < n_{grid} \right\} & y &\in \{ j \cdot h_{row} \mid j \in \mathbb{N}_0 \} \\ w &\in \left\{ k \cdot \frac{w_{viewport}}{n_{grid}} \;\middle|\; 1 \leq i < n_{grid} - x \right\} & h &\in \{ l \cdot h_{row} \mid l \in \mathbb{N} \} \end{aligned} \tag{6.1}$$

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. The grid constraints also define a mapping from a pixel-based onto a grid layout, as the values of indices i, j, k, l determining the position in the grid allow calculation of the cartesian coordinates and vice versa.

Transformation of a legacy user interface u to a web-based version u' thus requires a transformation from the pixel-based l_{legacy} to a grid layout l_{grid}, mapping the cartesian pixel space to the constrained grid-cell space. In the context of Rapid Web Migration Prototyping, this transformation needs to be automated, in order to address the efficiency, expertise and reuse requirements in section 6.2.1.

This layout transformation is an *optimisation problem* with discrete variables: it tries to minimise an *objective function* under a set of *hard and soft constraints*. The objective function is a similarity measure. This optimisation is implemented as

[101] https://getbootstrap.com/

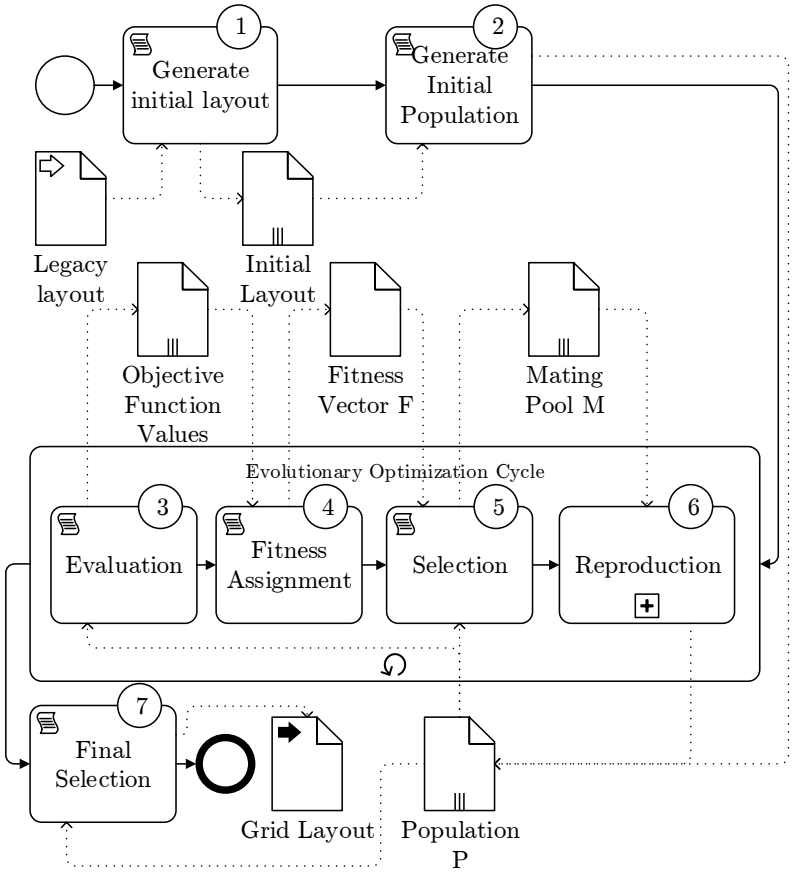


Figure 6.6.: UI Transformation Process

evolutionary algorithm (Weise, 2009). Figure 6.6 shows the conceptual process of the transformation using evolutionary optimisation. 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 P. The process ends by selecting an individual p from the last population. An individual is defined by its *genotype* $p.g$ from *genome* $\mathbb{G}$ which defines the optimisation's *search space*, and its *phenotype* $p.x$ from *phenome* $\mathbb{X}$ which defines the optimisation's *problem space* in which solutions of the optimisation are contained. The phenotype $p.x$ represents the observable characteristics of individual p derived from its genotype through genotype-phenotype mapping $p.x = gpm(p.g)$. Genotype $p.g$ is an instance $l_{grid} \in L_{grid}$. As all information of l_{grid} is observable, genotype and phenotype are identical $p.x = gpm(p.g) = p.g$ and thus populations are $P \subset L_{grid}$.

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*:

$$\begin{aligned} x_g &= \left\lfloor \frac{x_l \cdot n_{grid}}{w_{viewport}} \right\rceil & y_g &= \left\lfloor \frac{y_l}{h_{row}} \right\rceil \\ w_g &= \max\left(1, \left\lfloor \frac{w_l \cdot n_{grid}}{w_{viewport}} \right\rceil\right) & h_g &= \max\left(1, \left\lfloor \frac{h_l}{h_{row}} \right\rceil\right) \end{aligned} \tag{6.2}$$

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 optimisation follows *pareto optimality*: To calculate the **fitness** (4) of individual $p \in P$, let

$$P^* = \{p_i \in P \setminus \{p\} \mid f_k(p_i) \leq f_k(p) \forall f_k \in F\} \tag{6.3}$$

be the set of all individuals $p_i \neq p$ that are dominated by p, then the fitness of p is

$$v(p.x) = |P^*| \tag{6.4}$$

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 in $\mathbb{G}$. 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 (Weise, 2009), which puts more emphasis on searching around possible solutions compared to random selection of individuals.

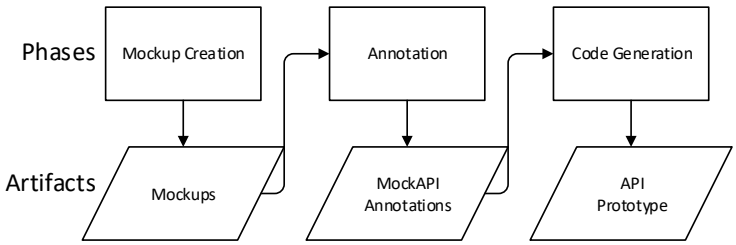


Figure 6.7.: MockAPI Process Flowchart

Final selection (7) is computed after reaching the maximum $n_{generations}$ reputations of the optimisation cycle from the final generation P_{last} based on F:

$$p_{opt} = \arg\max_{p \in P_{last}} F(p.x) \tag{6.5}$$

The conceptual process presented above is a model-to-model transformation between two CIMs. Section 6.3.2.2 discusses the implementation aspects of the UI transformation, to apply the evolutionary algorithm described above to concrete legacy user interfaces and generate grid-based WUIs as model-driven reengineering process.

6.3.1.4. Integration

This subsection addresses integration aspects of the AWSM:RM method.

Integration with existing API Prototyping methods - MockAPI

As described in section 6.3.1.1, web migration prototypes are horizontal prototypes (Wallmüller, 2001), i.e. they 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. However, to achieve the demonstrative (ISO/IEEE, 2017b) nature in a Web Migration context, the client/server communication via a public API needs to be represented. Public APIs of modern Web Applications facilitate serendipity through third-party development of applications and enable integration with third-party services, coining the term API economy (Tan et al., 2016). According to a recent survey, the vast majority (95.8%) of public Web APIs are REST APIs (Neumann et al., 2018). Therefore, 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).

MockAPI as shown in fig. 6.7 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 tool[102]. The DSL allows to specify the data model in terms of data objects and CRUDS[103] 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[104].

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 4.3.2, these visual artifacts can be retrieved from the executables E of Legacy System $\mathfrak{L}$. As MockAPI was designed to support different levels of mockups, hand-drawn sketches, as well as digital mockups, created using tools like balsamiq[105], legacy UI screenshots and mockups can be used interchangeably.

Integration with ongoing Development

To address requirement C4 Agile, the AWSM:RM method needs to be integrated 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 supporting the development process (ISO/IEEE, 2017b). Many iterative and agile development approaches, therefore, place Rapid Prototyping at the early stages of software development and most advocate iterative and incremental Prototyping towards an operational prototype as continuous risk management activity (Boehm, 1988; Martin, 1991; Simpson, 2016; Stapleton, 1997; DSDM Consortium, 2014; 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 explorative or experimental prototyping (Wallmüller, 2001) 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. As will be shown in section 6.4, 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.

[102] http://agilemdd.lifia.info.unlp.edu.ar/mockapi/
[103] Create, Read, Update, Delete, Search
[104] https://nodejs.org/
[105] https://balsamiq.com/

Integration with existing web migration methods

This section briefly outlines the integration of AWSM:RM with existing Web Migration methods as required by principle P2. AWSM:RM can be integrated with the model-driven methodology of *REMICS* (Mohagheghi and Sæther, 2011). While REMICS focuses on specifying the core migration activity areas of recover and migrate, it follows the SMART methodology (G. Lewis, Morris, Smith, et al., 2008) in initial phases (Mohagheghi and Sæther, 2011; Benguria et al., 2013) embedding the *establish context* activity (G. Lewis, Morris, Smith, et al., 2008) and adding sub-activities for describing benefits and describing disadvantages of a migrated solution (Mohagheghi and Sæther, 2011) to provide information for the *feasibility decision point*. In the REMICS context, this belongs to the *requirements and feasibility* activity area and the *requirements* phase of the REMICS methodology lifecycle. In REMICS' agile extension (Krasteva et al., 2013), a *requirement scrum* realizes this phase, with a dedicated requirements ScrumDemo task (Benguria et al., 2013). The requirements phase is the integration context and the ScrumDemo task the concrete integration point for AWSM:RM on process level. On artifact level, the rapid migration prototyping method can enhance REMICS methodology through providing a concrete artifact for assessing benefits and disadvantages of a migrated solution for the front-end at the feasibility decision point. While the back-end services focus of REMICS makes the demonstration of advantages for non-technical stakeholders hard, AWSM:RM would provide a way to advocate modernization by making advantages visible in the front-end.

Integration with *ARTIST* (Orue-Echeverria et al., 2014) can be achieved in the *Pre-migration phase* of the ARTIST process by including AWSM:RM to its tasks. In particular, within the *BUSFEAS* task, activity *BUSFEAS.A3* aims to estimate cost and benefits of the migration as key metrics for decision making (Orue-Echeverria et al., 2014). In the way specified through ARTIST, this requires well-informed staff in order to derive profit estimates without a clear understanding of benefits in terms of functional and non-functional advantages for customers without feedback mechanism. Only at a much later stage, after the migration decision, activity *VALPRO.A1* addresses customer needs through interviews, surveys, etc., but it also lacks a method for providing a concrete, tangible prototype to serve as means of communication. Therefore, integration of AWSM:RM can enhance ARTIST by facilitating a more informed migration decision and providing a means for gathering customer feedback in initial phases prior to migration. This integration would embed AWSM:RM into the BUSFEAS task, either as a separate activity or as an extension of BUSFEAS.A3 on process level. On artifacts level, AWSM:RM prototypes would represent an additional *output artefact* of BUSFEAS.A3 classified as [Software] or extending the ARTIST artefact classification taxonomy as dedicated [Prototype]. The *UWA/UWAT+* approach (Distante, Scott Tilley, and Canfora, 2006; Distante, Canfora, et al., 2006) does not specify migration decision making. Its first phase, *requirements*

elicitation, however, has a strong focus on stakeholders, their goals and corresponding requirements. Thus, AWSM:RM could be integrated either at the beginning of this phase or, similar to REMICS, by adding the *establish context* activity from SMART (G. Lewis, Morris, Smith, et al., 2008) and integrating AWSM:RM in this activity. Integration with other approaches can be achieved in a similar fashion, as activity in phases prior to the migration decision, where AWSM:RM prototypes provide an artifact demonstrating the web-based version and thus facilitating communication, assessment, and feedback. To identify these integration points, a concrete approach can be mapped onto the *ReMiP* reference model (Harry M Sneed et al., 2010b; Gipp and Winter, 2007), where AWSM:RM belongs to the *Preliminary Study* phase, the *Target Design* core discipline and the *Project Management* base discipline. For instance, in the context of AWS Migration (Amazon Web Services Inc., 2018), integration of the AWSM:RM method would provide a concrete and tangible contribution to the *building a business case for migration* activity.

6.3.2 Implementation

This section describes the implementation of the AWSM:RM method presenting the parts of the AWSM Platform that provide supporting infrastructure for Rapid Web Migration Prototyping, in particular the WASM-based ReWaMP toolchain, guided prototyping assistance and the generation of Web user interfaces.

6.3.2.1. ReWaMP Toolchain and Guided Prototyping Assistance

Implementation of ReWaMP based on WASM introduces four technical challenges, which we will briefly outline 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.

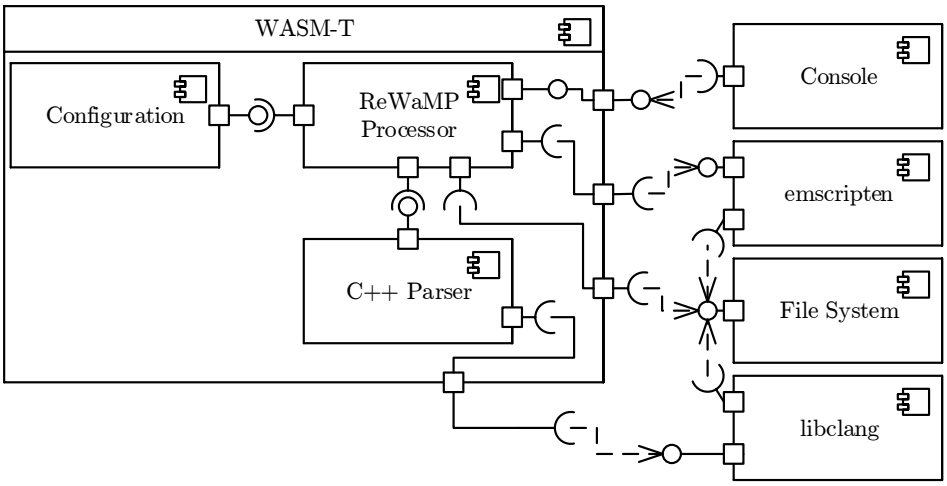


Figure 6.8.: WASM-T Architecture

WASM-T Transformation Toolchain. Figure 6.8 shows the architecture of WASM-T, the toolchain supporting the ReWaMP process. WASM-T is implemented in python and uses the python bindings of libclang[106] for parsing the legacy code.

The main components are ReWaMP processor and the C++ parser. The *configuration* interacts with the Prototyping Engineer as in step 2 in fig. 6.5 and stores the list of all main and related BBL files and the tool paths of libclang and emscripten.

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. C.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. **??** 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. 6.5. *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 *Tranformation completion* the ReWaMP processor fills function body stubs with generated code for 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 C.1.

[106] `https://clang.llvm.org/docs/Tooling.html`

Algorithm 6.1: ReWaMP Process Algorithm

Input: Configuration Cfg, extracted view behavior files B_u per view u
Output: ReWaMP runtime for each view u

function ReWaMPProcess()
  B_u = readAllFilesInDir(Cfg.workingDir)
  foreach f^* *in* Cfg.*mainFiles* **do**
    $cursor_{now}$ = getMainCursor(f^* in Cfg.workingDir, True)
    B'_u = getRelatedFiles($cursor_{now}$, B_u)
    B'_u = editIncludings($cursor_{now}$, B'_u, Cfg.headers)
    B'_u, $maps$ = minorChanges(B'_u)
    dir_{rel} = createRelatedDir(f^*)
    $cursor_{now}$ = saveRelatedFilesAndGetMainCursor(f^* in dir_{rel}, B'_u)
    B'_u = deleteUnusedParams($cursor_{now}$, B'_u)
    B'_u, $bodies$, $funcs$ = editMFCFunctions($cursor_{now}$, B'_u)
    saveRelatedFiles(B'_u)
    printHints($cursor_{now}$, B'_u)
    waitForME()
    B'_u = loadRelatedFiles()
    $cursor_{now}$ = getMainCursor(f^* in dir_{rel})
    $bodies$ = buildExpertFunctions($cursor_{now}$, B'_u, $bodies$)
    B'_u = addJSONMethods($cursor_{now}$, B'_u)
    B'_u = processFlags(B'_u)
    $cursor_{now}$ = saveRelatedFilesAndGetMainCursor(f^* in dir_{rel}, B'_u)
    $embinds$, $funcs$ = collectEmbinds($cursor_{now}$, B'_u, $maps$, $funcs$)
    $cursor_{now}$ = saveRelatedFilesAndGetMainCursor(f^* in dir_{rel}, B'_u)
    B'_u = addFunctions($cursor_{now}$, B'_u, $funcs$)
    $cursor_{now}$ = saveRelatedFilesAndGetMainCursor(f^* in dir_{rel}, B'_u)
    B'_u = writeNewBodies($cursor_{now}$, B'_u, $bodies$)
    B'_u = addSendStatusFunction(B'_u)
    B'_u = addEmbinds($embinds$, B'_u)
    saveRelatedFiles(B'_u)
    copyLibraryFiles(dir_{rel})
    addJSInitFunction($embinds$, $maps$)
    compileFilesToWasm(B'_u)

RWMPA Guided Prototyping Assistance System

Initial experiments with ReWaMP and its toolchain WASM-T (cf. section 6.4.1) identified the need for providing guidance due to the complexity of the ReWaMP process and improved tool support compared to the console-based interaction with WASM-T and external editor-based manual interventions. This has led to the implementation of the *Rapid Web Migration Prototyping Assistance (RWMPA)*, an assistance system which guides Prototyping Engineers through the ReWaMP process. The ReWaMP process guided through RWMPA is referred to as *RWMPA workflow* in the following. It contains adaptions to the conceptual ReWaMP process 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 shown in fig. C.2. Table 6.2 shows the mapping between ReWaMP and RWMPA workflow tasks. 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 Engine[107] on top of which the RWMPA Web Application provides a WUI 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 integrated with specific Web Migration approaches as described in section 6.3.1.4.

Table 6.2.: ReWaMP tasks and realisation 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

ReWaMP	RWMPA workflow
1M extract view-related BBL files	5S find related files, 6U select files
2M configure WASM-T	done automatically, not in workflow
3T preprocess view BBL	7S preprocess
4M expert changes	10U del. code, 11U repl. code, 16U exp. changes
5M expert file	16U expert changes
6T complete transformation	17S save
7T compile to WASM	14S compile to WASM

As shown in fig. C.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 workflow 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

[107] `https://camunda.com/products/bpmn-engine/`

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 Ionic 4[108]. Plugins are compiled to W3C WebComponents (W3C, 2018a) (cf. principle P1) for embedding in a step worker page using on the stencil[109] compiler, migration services are based on node.js.

RWMPA provides a set of plugins including file management based on elFinder[110] and source code editing for steps 4 and 5 in fig. 6.5 with syntax highlighting using Ace. Figure 6.9 shows a merge view of RWMPA supporting the Prototyping Engineer to review and accept or reject changes from WASM-T preprocessing.

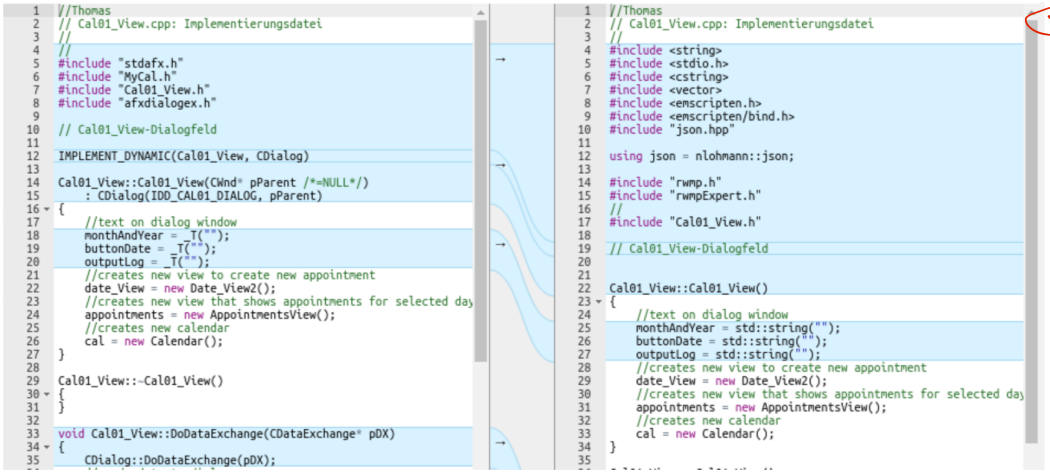


Figure 6.9.: RWMPA Step Worker Page Example: Merge View

6.3.2.2. UI Transformer

The user interface transformation for Rapid Prototyping described in section 6.3.1.3 is implemented in a component of the AWSM platform called *UI Transformer*. It supports conversion of pixel-based legacy layouts based on the MFC framework (cf. section 2.1) to responsive fluid grid layouts based on bootstrap. UI Transformer is implemented in Python 3 with a C implementation of computationally complex transformation and objective functions evaluation. To implement the transformation process based on evolutionary optimisation as described in section 6.3.1.3, UI Transformer requires three stages as shown in fig. 6.10. These three stages represent the three parts of an automated, model-driven reengineering horseshoe known from ADM (Pérez-Castillo, Guzmán, et al., 2011): 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.

Layout Analysis. The first stage of UI Transformer extracts the required information according to the CIM-level formalisms introduced in section 4.3.2 and section 6.3.1.3

[108] https://ionicframework.com/
[109] https://stenciljs.com/
[110] https://github.com/Studio-42/elFinder

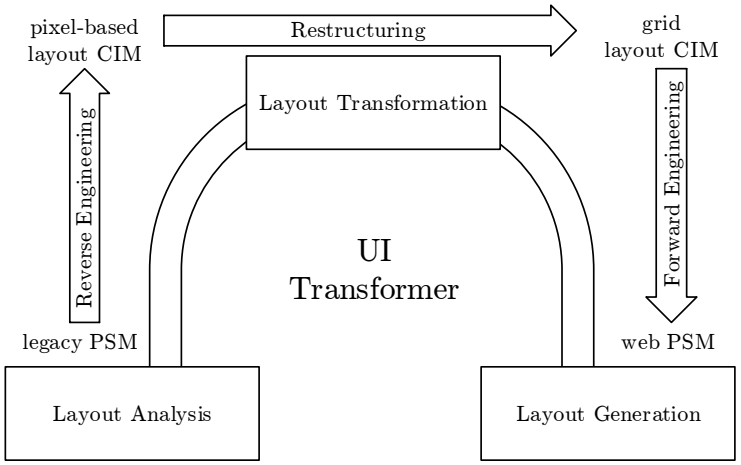


Figure 6.10.: UI Transformer as model-driven reengineering process

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*[111] containing definitions of various application resources. Using a tool like *Resource Hacker*[112] they can also be retrieved from executables in E. Descriptions of user interfaces u are expressed in *dialog* elements. UI Transformer parses the resource definitions to extract viewport, controls, types, positions, and sizes. Parsing is done based on *regular expressions (RegEx)*. To determine the *hierarchy* of controls, which occur 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} according to the formalism introduced in the conceptual model.

Layout Transformation. Layout transformation takes l_{legacy} as input and implements the optimisation algorithm in fig. 6.6. 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 $l_{legacy,i}$ 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 re-use measurement results. An example of a complex measurement is *Levenshtein distance* (Black, 2019), used to identify variations in the order of controls $c \in u$. A C-based implementation[113] was used for Levenshtein distance calculation.

Objective Functions $f_k : L_{legacy} \times L_{grid} \mapsto \mathbb{R}_0^+$ compare a legacy layout instance with a grid layout based on the calculation of measurements on both layouts and

[111] `https://docs.microsoft.com/en-us/cpp/ide/resource-files-cpp`
[112] `http://www.angusj.com/resourcehacker/`
[113] `https://github.com/ztane/python-Levenshtein/`

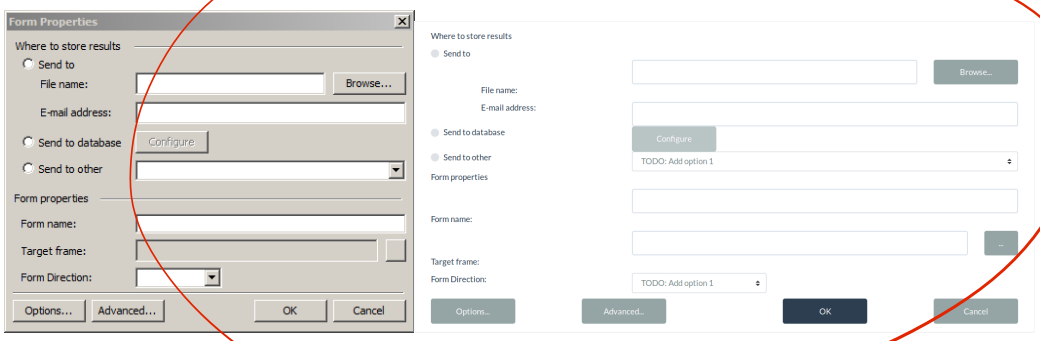
a calculation to aggregate the two measurements into a non-negative real number. The transformation aims at creating grid-based layouts that represent the legacy layout. Similarity of layouts is a phenomenon highly related to human perception, not yet very well understood and thus an active field of 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 (cf. *Feature Engineering* Domingos, 2012) 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 ctypes[114].

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. 6.6 and generating the PSM of the WUI, i.e. an HTML file with corresponding CSS directives, which can be rendered in a Web browser. Despite principle P1, layouts are generated using bootstrap, as older browsers do not support CSS Grid. Field research in section 4.1.1, 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 Mako[115] 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 row and then ordered by column. Iteration over the hierarchy

[114] `https://docs.python.org/3/library/ctypes.html`
[115] `https://www.makotemplates.org/`



(a) Legacy Layout **(b)** Transformed Layout

Figure 6.11.: Layout Transformation Sample

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.

6.4 Evaluation

This section evaluates AWSM:RM regarding the requirements in section 6.2.1. Additional experimental evaluations further address effectiveness, efficiency, and expertise.

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 re-use 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 AWSM: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 AWSM:RM with test subjects with limited Web Engineering and no Web Migration expertise. AWSM: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.

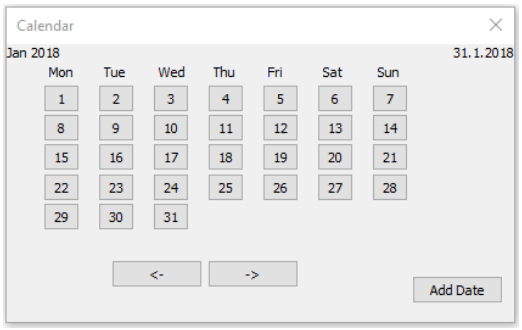
Plausibility. The web migration prototypes resulting from AWSM:RM are Web Applications using the standard Web technology stack of HTML, CSS, and JavaScript in addition to the new WebAssembly standard. Unlike wrapper approaches, they run entirely independent of the Legacy System, and they run in any major 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.

6.4.1 Experimental Evaluation of ReWaMP

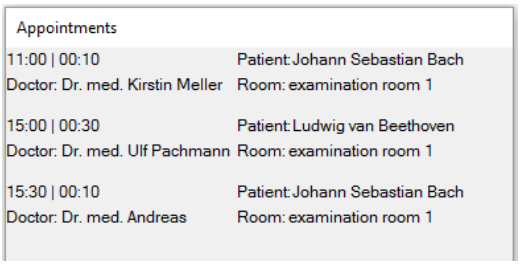
Experimental evaluation of ReWaMP focuses on the aspects of time and effort for ReWaMP, on required expertise of the Prototyping Engineer, on a detailed understanding of the most prolonged/most difficult tasks in ReWaMP and on possible results from the application of ReWaMP.

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. 6.12. 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.

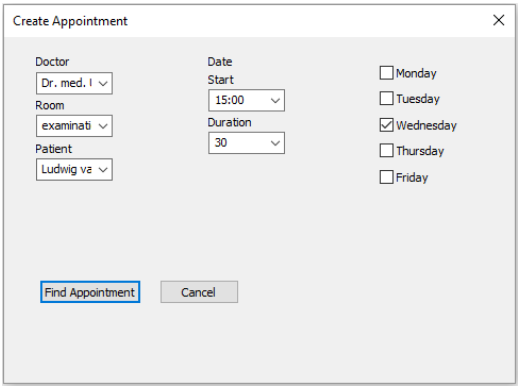
The experiment was conducted through observed test runs of the ReWaMP process with test subjects. The *test subjects* impersonated the role of Prototyping Engineer and were observed during execution by a *researcher*. Table D.1 shows the guidelines



(a) Calendar Dialog u_{cal}



(b) Appointments View $u_{appointments}$



(c) New Appointment Dialog u_{new}

Figure 6.12.: Test Dataset B_T Views

for interaction of the researcher and the test subject. The questionnaire shown in appendix D.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 by a brief introduction to the scenario, its context visualized as fig. 6.1 and the ReWaMP process visualized as fig. 6.5. 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). Clion[116] 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 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.

[116] `https://www.jetbrains.com/clion/`

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. 6.5, 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.

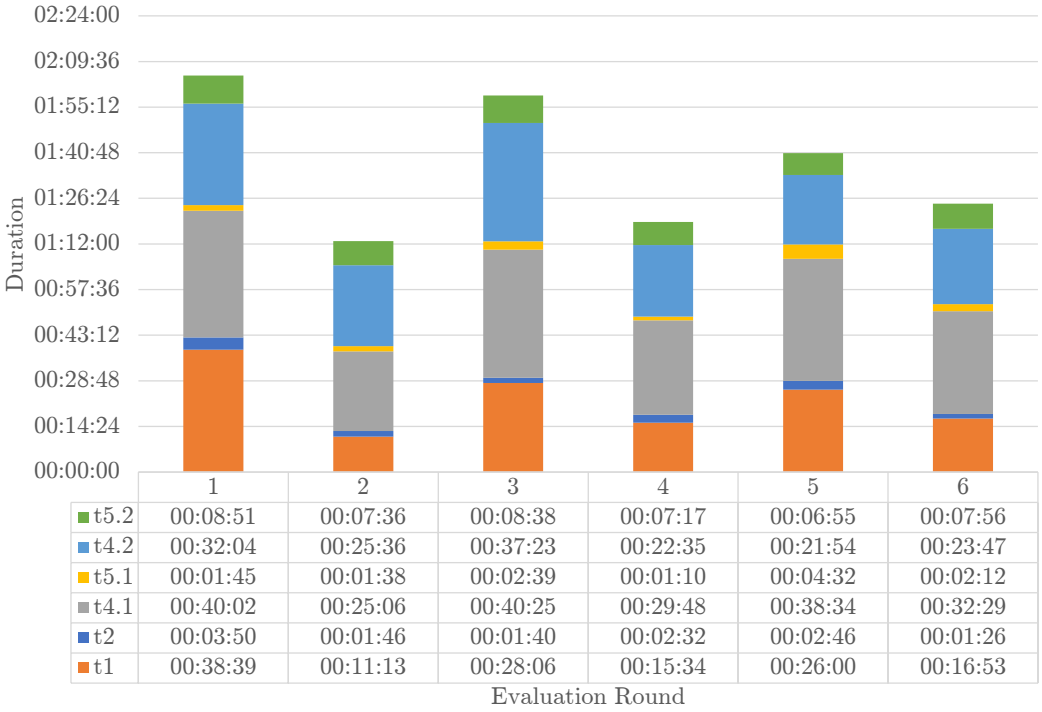
Effort evaluates the amount and extent of changes to the existing legacy code required by ReWaMP, measured in terms of e_T - the lines of code added/changed/deleted by WASM-T and e_M - the SLOC added/changed/deleted manually by the Prototyping Engineer. e_M was measured through observation of the test 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 (σ = 2s) time of $\overline{t_T}$ = 00:00:47h was required by WASM-T. The average time for manual tasks was $\overline{t_M}$ = 00:49:36h (σ = 00:10:06h). Considering time distribution on the tasks shown in fig. 6.13, 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

$\overline{t_M}$	$\overline{t_T}$	$\sigma(t_M)$	$\sigma(t_T)$	$\overline{t1}$	$\overline{t2}$	$\overline{t4}$	$\overline{t5}$
00:49:36h	00:00:47h	00:10:06h	2s	00:11:22h	00:02:20h	00:30:49h	00:05:06h

Table 6.4 provides an overview of the effort results, averaged over the test subjects per on view. Figure 6.14 shows the effort distributions. WASM-T performs almost $3/4$



	1	2	3	4	5	6
t5.2	00:08:51	00:07:36	00:08:38	00:07:17	00:06:55	00:07:56
t4.2	00:32:04	00:25:36	00:37:23	00:22:35	00:21:54	00:23:47
t5.1	00:01:45	00:01:38	00:02:39	00:01:10	00:04:32	00:02:12
t4.1	00:40:02	00:25:06	00:40:25	00:29:48	00:38:34	00:32:29
t2	00:03:50	00:01:46	00:01:40	00:02:32	00:02:46	00:01:26
t1	00:38:39	00:11:13	00:28:06	00:15:34	00:26:00	00:16:53

Figure 6.13.: Time distributions per task and test run

of all required changes (fig. 6.14a). The manual effort e_M (fig. 6.14c) distribution shows that the Prototyping Engineer mainly performs delete operations, whereas $2/3$ of the automatic changes e_T from WASM-T (fig. 6.14d) add code.

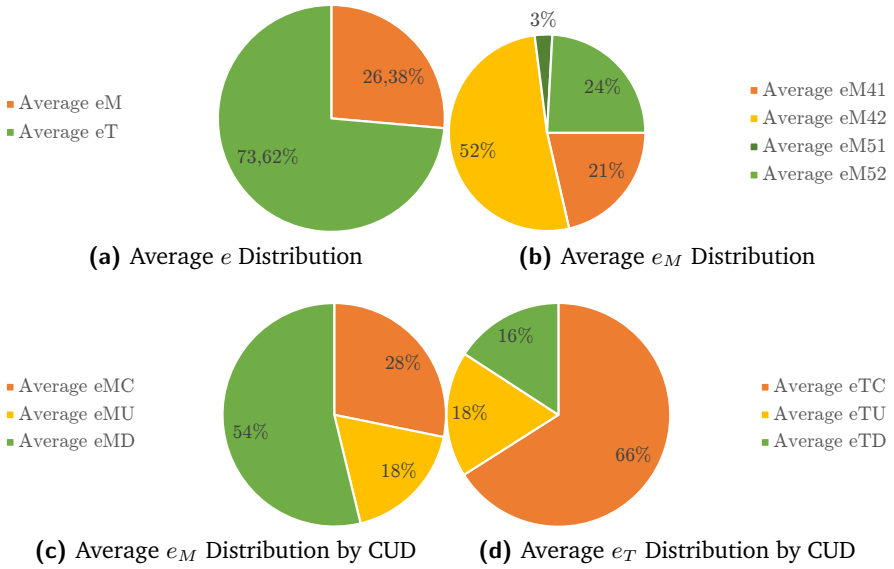
Table 6.4.: Effort Measurement Statistics, in SLOC

$\overline{e_M}$	$\overline{e_T}$	$\sigma(e_M)$	$\sigma(e_T)$	$\overline{e_M^C}$	$\overline{e_M^U}$	$\overline{e_M^D}$	$\overline{e_T^C}$	$\overline{e_T^U}$	$\overline{e_T^D}$
121	338	4.5	4.5	34	22	65	224	62	54

Figure 6.15 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 required 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 online[117].

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. Considering 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 and readability of the legacy code and unfamiliarity with the code base. While the latter problem would not be faced to the same extent by



(a) Average e Distribution **(b)** Average e_M Distribution

(c) Average e_M Distribution by CUD **(d)** Average e_T Distribution by CUD

Figure 6.14.: Effort Distributions

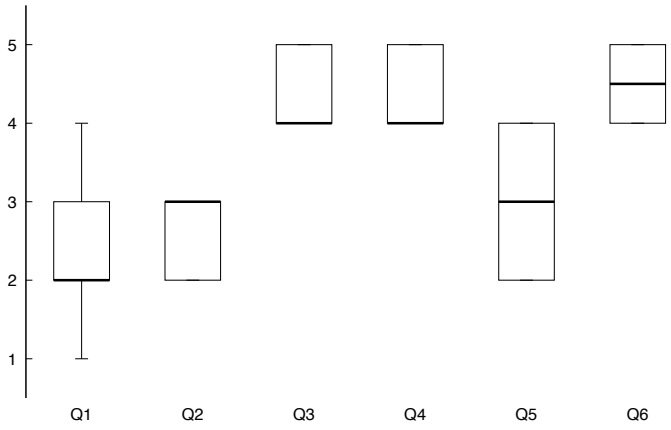


Figure 6.15.: ReWaMP Questionnaire Results, Q1 ReWaMP easy to understand, Q2 manual work for ReWaMP, Q3 WASM-T did much work, Q4 without WASM-T more work, Q5 process was short, Q6 without WASM-T more time

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.

6.4.2 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 E.1 shows the guidelines for interaction of the researcher and the test subject. The questionnaire shown in appendix E.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 time[118] was between 00:59:12h and 01:30:54h. Table E.3 shows the full time evaluation data, aggregations are shown in table 6.5. On average per view, $\bar{t}$ = 00:37:52h, of which $\overline{t_S}$ = 00:00:52h (σ = 7s) was required by the service tasks. The average time for user tasks was $\overline{t_U}$ = 00:37:00h (σ = 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.

Table 6.5.: RWMPA Time Measurement Statistics

$\overline{t_U}$	$\overline{t_S}$	$\sigma(t_U)$	$\sigma(t_S)$	$\overline{t6}$	$\overline{t10}$	$\overline{t11}$	$\overline{t16}$
00:37:00h	00:00:52h	00:05:43h	7s		00:01:11h	00:01:58h	00:16:59h	00:08:30h

Table E.2 shows the full empirical evaluation data. As shown in fig. 6.16, 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).

Analysis. In comparison to ReWaMP, the time for manual interventions of the Prototyping Engineer was reduced from $\overline{t_M}$ = 00:49:36h (σ = 00:10:06h $\approx 20\%\overline{t_M}$) in ReWaMP to $\overline{t_U}$ = 00:37:00h (σ = 00:05:43h $\approx 15\%\overline{t_U}$). The time distribution across tasks in comparison to equivalent tasks of ReWaMP table 6.2 shows some

[118] time measurements reported are based on five test subjects due to invalidity of measurements for u_{new} for two test subjects

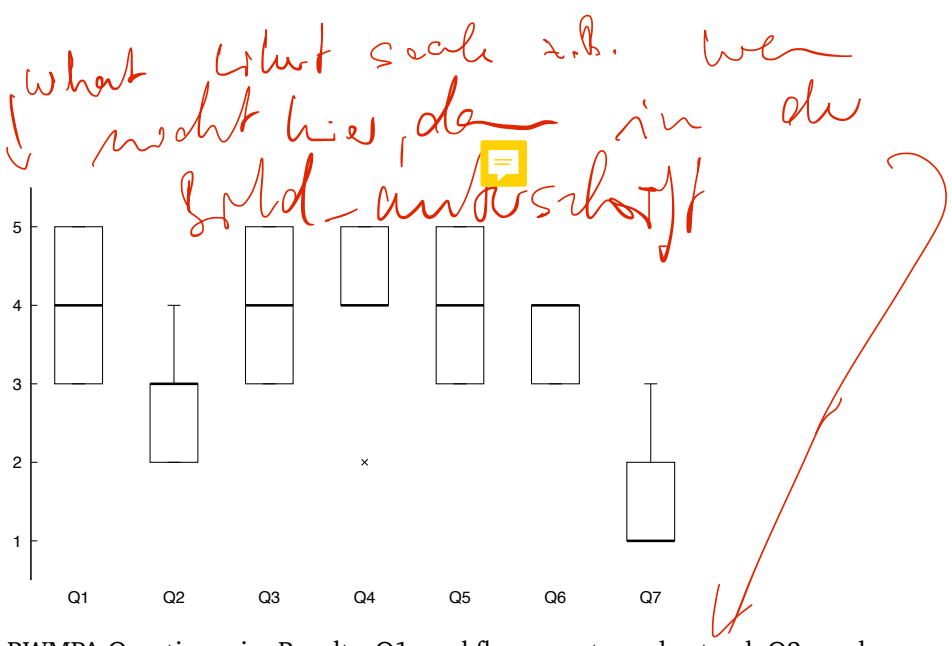


Figure 6.16.: RWMPA Questionnaire Results, Q1 workflow easy to understand, Q2 much manual work required, Q3 workflow reduced work effort, Q4 workflow guidance supported workflow understanding, Q5 task guidance supported task understanding, Q6 aware what is to do, Q7 manual configuration of tools

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 6.4.1 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.

6.4.3 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 $L_T = \{l_1, \ldots, l_{50}\}$ 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, $\overline{d} = 0.32$) 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 F.1 implemented as Google Form 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. 6.10: 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 initialisation 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 6.3.1.3, 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 (median. 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.

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 F.1.

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

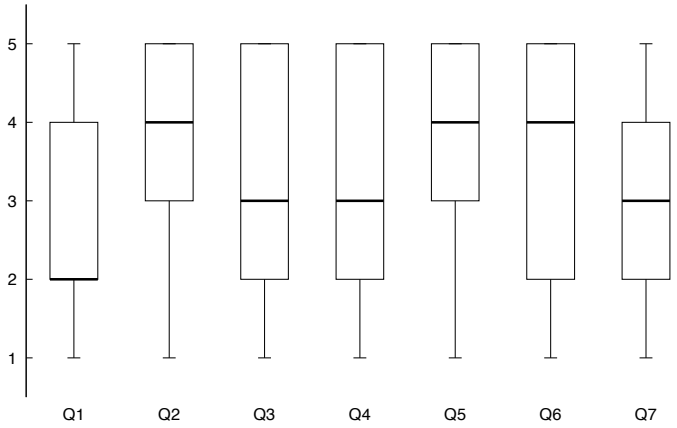


Figure 6.17.: UI Transformer Questionnaire Results, Q1 same size, Q2 elements at least as large, Q3 left alignment consistent, Q4 right alignment consistent, Q5 same labels, Q6 same order, Q7 layouts identical

Each test subject contributed $30 \cdot 7 = 210$ ratings, resulting in 2100 ratings overall, 300 per question in the questionnaire. Figure 6.17 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 F.2.

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

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 from 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 which is not required to be run repeatedly. The comparison between t_6 and $t_1 1$ shows that the potential performance increase through implementation in a low-level hardware-oriented platform like C can significantly accelerate the transformation.

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 ($Q7_6 = 4.1$). 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.

6.5 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, based on the results of first evaluation, has been enhanced with a guidance and automation system. The conceptual model and implementation of AWSM:RM 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.

7 AWSM Customer Impact Control Method

This chapter addresses research objective RO4:

> To provide an HCI method, models and tools to control the impact of Web Migration on customers with limited resources and lack of Web Engineering expertise.

Following an analysis of the current situation, identification of requirements and a review of related work, the conceptual model and implementation of the AWSM:CI method consisting of three mechanisms to address RO4 are described. The method is evaluated against the requirements, and the evaluation is supported by an experiment which includes analysis of both empirical evaluations and objective measurements.

7.1 Research Questions

To design and evolve a solution for research objective RO4, this chapter addresses three research questions:

> **AWSM:CI Research Question 1**: How to control the impact of Web Migration on customers with limited resources and lack of Web Engineering expertise?

> **AWSM:CI Research Question 2**: How to measure the similarity of non-web and Web versions of a user interface?

> **AWSM:CI Research Question 3**: How to analyze visual UI similarity through computation of objective measurements?

7.2 Analysis

ISVs as described in section 2.1 are successful in their market segment, and their legacy software has accumulated large userbases, with long-running customer relationships and contracts for updates and maintenance. These existing users are familiar with the use of the Legacy System and well-adapted to its legacy user interface. This adaption can extend up to the business process level: one of the

observations from the field research described in section 4.1.1 is that the business processes in many doctors' offices are governed by the processes represented in the PMS. This makes it hard to introduce larger changes in user interaction or even replace the PMS because of the potential *customer impact* (cf. fig. 4.2). When conducting Web Migration 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, Tortora, De Lucia, et al., 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). In the forward engineering phase of a traditional horseshoe reengineering approach, it is "important to comply with the cognitive characteristics of the legacy application" (Distante, Scott Tilley, and Canfora, 2006). Learning how to use a new Web user interface requires an effort, which ISV customers are not immediately willing to take. Thus changing the user interface poses a risk for the ISV as it can mean, in the worst case, to lose customers. The awareness of this risk contributes to intra-organisational resistance (Khadka, Batlajery, et al., 2014).

Web Migration visibly impacts the user interface and thus the users. The legacy user interface consists of one or more windows, rendering its controls according to the desktop GUI framework and platform in use. A Web user interface, in contrast, is rendered in a Web browser in one or more browser tabs, rendering its controls according to the Web UI framework, platform and specific CSS styles in use. Changing the legacy interaction paradigm towards a navigational paradigm is one of the most critical challenges of Web Migration (Distante, Perrone, et al., 2002). The industry requirement of maintaining a similar look and feel (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006; Distante, Perrone, et al., 2002) is formally expressed as *UI Similarity*. Analysis of UI Similarity is a research area of HCI, application of which exceeds Web Migration: it is used to support software quality tasks such as testing, cross-platform GUI matching, GUI modernization, and project effort estimation (Grechanik, C. W. Mao, et al., 2018).

In order to "avoid presenting the user with a drastically different application" (Distante, Scott Tilley, and Canfora, 2006), changes need to be controlled, favoring an incremental transition evolving the software over abrupt changes (cf. also fig. 4.2). Design decisions concerning the user interface need to be informed through analysis of UI Similarity. 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). The goal is to "preserve the old (established) mode of operation *as far as possible*" (Distante, Scott Tilley, and Canfora, 2006). This possibility defines the target scope for UI Similarity in the context of Web Migration: it 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.

In the specific context of Web Migration, not all dimensions of UI Similarity are equally important. For instance changes of Thesaurus such as renaming elements or Task, i.e. modifying the business processes, can and should be avoided (Harry M. Sneed et al., 2010a; Harry M Sneed, 1995) during Web Migration. On the other hand, changes in Layout and to some degree in Behavior and Material follow conclusively from the *change of environment* (IEEE Computer Society, 2014) through Web Migration. Layout changes result from the different layout paradigms (pixel-based to grid-based, cf. section 6.3.1.3), material changes (GUI framework UI controls to CSS-styled UI controls) and viewing conditions (fixed-size desktop window to variable sized browser window), which cause changes in the positioning and size of UI controls. Positions and sizes of UI 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: capturing the legacy UI layout is a "preponderant requirement" of modernization to regenerate the legacy look and feel (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012).

Measurability is a recognized necessity when dealing with legacy systems to quantify their state (Masak, 2006). However, existing legacy metrics are based on the Legacy System only and do not consider target and source system together. To control the customer impact of user interface changes through Web Migration, changes affecting UI Similarity must be made measurable. The importance of measurement for engineering in general and software engineering in particular is widely acknowledged (IEEE Computer Society, 2014). Existing Web Migration approaches as presented in section 3.2 barely consider UI Similarity. Those which do (Barbier, Deltombe, et al., 2013; Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Distante, Scott Tilley, and Canfora, 2006; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006; Distante, Perrone, et al., 2002), consider UI Similarity, often referred to as a similar "look and feel" (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006) on abstract level, without providing a concrete way of measuring it, or for TUIs only (Bodhuin, Guardabascio, et al., 2002) which have a significantly reduced design space.

Existing approaches for measuring UI similarity cannot be applied to legacy and Web UIs because they are based on code analysis or target a different degree 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.

7.2.1 Requirements

The following requirements have been derived based on RO4, 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.

7.2.2 Related Work

UI Similarity has been of interest in various fields of research. In the following, an overview of related work that addresses aspects of UI Similarity is presented.

Web Application Feature Matching. FMAP (Roy Choudhary, Prasad, et al., 2014a) is an approach for automatic matching of features across different UI versions of a multi-platform Web Application, in particular between desktop and mobile WUI. FMAP addresses unintended differences in feature sets to prevent negative impact on user experience through inconsistencies. It defines *consistency* of WUI versions based on the correspondence of features, with features referring to a functionality provided through a set of services invoked in a defined order. This can be determined in the *traces* of HTTP communication between frontend and backend, so the *feature mapping problem* is defined based on abstraction and canonicalization of traces. Abstraction is achieved through recognition of *actions* based on clustering using Jaccard distance. Canonicalisation is done by identification and removal of tandem

repeats of action sequences. Finally, the features are mapped using maximum bipartite matching. Web Application feature matching considers UI similarity, but with a focus on user interaction. It abstracts from the behavior to the task dimension. Applicability for Web Migration is not achievable since it requires near-identical versions of a Web Application with the same backend in order to analyze interaction traces based on HTTP communication. Source and target system in Web Migration are significantly more different, the source system is not a Web System, and they do not share the same HTTP-based backend.

GUI Differencing. Grechanik et al. (Grechanik, C. W. Mao, et al., 2018; Grechanik, Xie, et al., 2009b; Grechanik, Xie, et al., 2009a) contributed insights on UI Similarity through their work on GUI differencing for GUI test script adaption. They propose an automated approach for differencing versions of GUIs without requiring the source code, called GUIDE (Grechanik, C. W. Mao, et al., 2018). Automatic *GUI differencing* means computation of a mapping function that maps each GUI control in one version of a GUI to at most one control in another version. This identification of GUI differences is difficult due to the richness and heterogeneity of controls across various frameworks and platforms. GUIDE makes use of accessibility interfaces to reconstruct tree-based GUI models from running instances of user interfaces. Empirical experiments (Grechanik, Xie, et al., 2009a) revealed that manual GUI matching is a time-consuming perceived as difficult by test subjects, even for GUIs with less than 30 controls. In (Grechanik, Xie, et al., 2009b), Grechanik et al. use a pair-wise match score between all combinations of UI controls in subsequent releases of a GUI calculated as a weighted sum of object property differences to match UI controls based on the highest score, but with inconsistent overall precision results (Grechanik, C. W. Mao, et al., 2018). Generally, automated *sound* (precision=1) and *complete* (recall=1) GUI differencing is *undecidable* (Grechanik, C. W. Mao, et al., 2018). The approximation presented in GUIDE uses tree-edit-distance and through experimentation shows that tree-edit algorithms are more effective than the baseline approach and that precision decreases with increasing number of differences.

Cross-Browser Inconsistencies (XBI) detection is the automatic detection of variations in layout and functionality of Web user interfaces when rendered in different browsers or on mobile platforms with different viewports. *DOM-based strategies* (W. M. Watanabe et al., 2019; Roy Choudhary, Prasad, et al., 2014b; Roy Choudhary, Versee, et al., 2010) require the DOM for segmentation of the WUI and/or identification of XBIs. *Computer-vision strategies* (Saar et al., 2016) work based on visual analysis of screenshots of WUIs. *WebDiff* (Roy Choudhary, Versee, et al., 2010) was one of the first XBI detection tools and introduced the blueprint two-step *differential testing* (Mckeeman and Problem, 1998) strategy followed by various other XBI approaches: 1) segmentation of the WUI according to its DOM structure and 2) comparisons of corresponding UI controls across different versions. WebDiff uses comparison of DOM attributes of UI control pairs detecting changes in position

and size in combination with visual analysis using histogram-based Earth Mover's Difference (EMD) (Rubner et al., 1998) with a threshold.

X-PERT (Roy Choudhary, Prasad, et al., 2014b; Choudhary et al., 2013) follows the WebDiff strategy but extends it towards *behavioural XBIs*. A crawler extracts navigation models in different browsers, and *graph isomorphism checking* detects behavioral XBIs in the state graphs. Controls are matched using the *match index metric* based on DOM attributes. Structural XBIs are detected using layout comparison in *alignment graphs*, which represent relative layout information extracted from the DOM. Visual XBIs are detected using the histogram-based χ^2 distance on screenshots of DOM leaf nodes.

(W. M. Watanabe et al., 2019) propose an XBI detection approach based on classification of DOM elements. It follows the WebDiff strategy, using DOM-based segmentation and applying incompatibility classification. *Outer incompatibilities* refer to differences in position and size, *inner incompatibilities* are differences in the inner content like text formatting, background position, visual effects. The classification is based on a rich feature set, representing position, size, viewport, alignment, complexity and screenshot comparisons using Image Diff, χ^2 distance and pHash distance.

Browserbite (Saar et al., 2016; Semenenko et al., 2013) is a computer-vision-based XBI detection available as commercial tool[119]. In contrast to DOM-based strategies, it uses *image segmentation* to identify *regions of interest* (ROIs) in screenshots of WUIs and applies computer vision techniques to extract features representing these ROIs. This requires a four-step process of screenshot capture, image segmentation, image comparison, and XBI classification. Segmentation is achieved through edge and corner detection in the grayscale screenshot forming a binary corner pixel image, vertical and horizontal *dilatation transforms* to combine dense corner pixels into connected regions and blob analysis for grouping contiguous pixels. Rectangular bounding boxes around these groups are the segmented ROIs. Comparison of corresponding ROIs is calculated based on position, size and raw moments based on histogram comparison. Incompatibilities are detected using binary and quaternary supervised machine learning classification. An experimental comparison of decision trees and *artificial neural networks* (ANNs) showed better F-measures for the ANN implementation.

Both GUI Differencing and XBI detection overlap with analysis of UI similarity. However, they target differences between variants of a UI, for GUI Differencing within a fixed environment, i.e. on the same platform and framework, for XBI detection within different HTML rendering engines. The UI variations introduced are more subtle compared to the consequences of migration from a desktop GUI to the Web:

[119] `http://browserbite.com/`

GUI Differencing addresses intended UI changes through perfective maintenance activities, XBI detection addresses even more subtle unintended changes through differences in how a WUI with identical description is rendered in different browsers. While UI similarity analysis for Web Migration requires a similar basis of matching 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 technical focus of detailed detection of fine-grain UI changes for test script adaption or XBI detection. Smaller and fewer differences allow for higher precision (Grechanik, C. W. Mao, et al., 2018).

Page Segmentation. As shown above, for analysis of similarity of UI layouts, it is required to identify the visually and semantically coherent two-dimensional segments (cf. ROIs in Browserbite) that constitute the UI. UI controls form atomic segments of a UI and form a hierarchy through containment and alignment relationships (cf. alignment graphs in X-PERT). Page Segmentation approaches address the problem of identification of these 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 *Vision based Page Segmentation algorithm (VIPS)* (Cai et al., 2003) is a DOM-based page segmentation approach that also addresses visual perception aspects. It aims at extracting the semantic structure of a Web page as a hierarchy of non-overlapping *blocks* (ROIs), each block is assigned with a *Degree of Coherence (DoC)* value representing its visual-perception-based coherence. The VIPS algorithm identifies blocks from the DOM tree, then identifies horizontal and vertical lines that are not intersecting blocks as *separators* and uses the separators to identify the semantic content structure. Block identification is made top-down starting with the page as block, recursively until a threshold DoC is reached. It uses a rule-based approach, with rules based on DOM properties. Separators are identified by incrementally adding the identified blocks and distinguishing areas not occupied by the blocks; their weight is assigned according to *visual cues* like size, color, etc. The semantic page structure is derived through the merging of blocks based on the separator weights, creating a *visual block tree*. VIPS was one of the first approaches to emphasize the importance of visual perception for segmenting pages into their semantic structure, combining DOM with visual cues. Thus, DOM tree and block tree are not structurally similar, as the block structure identified through VIPS groups elements independent of their position in the DOM tree. A revised version of VIPS is used in the Vision-based Data Extractor (ViDE) (Wei Liu et al., 2010) adding layout features specific for identification of *data records* in WUIs.

Bricolage (Kumar, Talton, et al., 2011) is an approach for Web design retargeting, i.e. transferring content of a Web page into the design of another. This requires

a mapping between the two WUIs which Bricolage achieves through supervised learning with a generalized perceptron algorithm. The input of the mapping are segmented Web pages, derived through Bricolage's page segmentation algorithm *Bento*. Bento re-arranges the DOM in order to align the tree structure better with visual perception for consistent segmentation. It re-arranges parent-child relationships to correspond to visual containment by calculating bounding boxes and assigning DOM subtrees to parents with the minimal bounding box containing all child bounding boxes. The tree is pruned from nodes without visual representation and finally adjusted according to separator identification similar to VIPS. In contrast to VIPS, Bento follows a bottom-up strategy for page segmentation, starting at the DOM leaf nodes.

Block-o-Matic (BoM) (Sanoja and Gancarski, 2014) proposes a hybrid page segmentation approach combining document processing methods with visual-based content segmentation. It aims at integrating perceptual *flow order* (reading order). BoM defines a three-phase process of analysis, understanding, and reconstruction. Analysis derives the *content structure* from the DOM, understanding maps content structure to *logical structure*, reconstruction results in the segmented page. BoM reports an average improvement of 7% over VIPS accuracy.

While the above page segmentation approaches highlight the importance of visual perception by considering visual aspects/cues for improving understanding of the semantic structure of a WUI, they all require the page's DOM as input. In Web Migration, the DOM is only available for the target system, but not the source system. Thus, a full computer-vision based approach using image segmentation techniques independent of the DOM is required. The approach proposed in (J. Kong et al., 2012) addresses shortcomings of DOM-based segmentation approaches through the application of image processing for recognition of atomic interface objects. It follows a bottom-up strategy based on spatial graph grammars (SCG). SCGs represent atomic interface objects as vertices and their relationships (e.g. touch, containment) as edges. Recognition of atomic interface objects using image processing requires two phases: segmentation into candidates using edge detection and analysis of geometric shapes, and classification of GUI element type or rejection of the candidate through nearest-neighbor classification based on distance metrics and pairwise comparisons against a database of labeled samples. Interpretation of the user interface is then achieved through pre-defined grammars. While the bottom-up strategy based on image processing and classification is promising for application in Web Migration, (J. Kong et al., 2012) targets WUIs, does not provide implementation or the nearest-neighbor database and does not report precision/recall measures.

UI Generation from Images was recently presented by Microsoft Research's AI Lab in the Sketch2Code[120] tool. It aims at supporting user interface design by

[120] `https://sketch2code.azurewebsites.net/`

transforming hand-drawn UI mockups into HTML wireframes. Sketch2Code is a computer-vision approach implemented using the Azure Custom Vision Service and was trained on millions of images[121] for object detection of hand-drawn UI controls combined with text recognition. The HTML is generated heuristically based on positions of recognized objects and texts. While Sketch2Code is a strong example of the power of computer-vision based approaches and the use of Artificial Intelligence (AI) in UI Similarity, it is specifically targeted for hand-drawn mockups. Screenshots of legacy desktop GUIs are significantly harder to analyze, due to more visual noise - mockups already represent a near-perfect edge representation of a UI whereas this requires heuristic edge detection through image processing, which significantly facilitates object recognition. Our experiments with Sketch2Code could not produce any useable results when using legacy GUI screenshots[122]. Similar projects have been presented by AirBnB[123] and in pix2code[124].

UI Similarity in Web Migration has received less attention. Among the Web Migration approaches presented in section 3.2, only few have considered aspects of UI Similarity for migration. CelLEST (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) aims at creation of similar WUIs from the Legacy System. Based on interaction traces, behavior and task models are extracted, and the WUI is specified for the identified functionalities. A clustering-based similarity of legacy UI screenshots is used to identify compound states in the state-transition interaction model (Stroulia, El-Ramly, and Sorenson, 2002) based on a custom feature set (Stroulia, El-Ramly, L. Kong, et al., 1999). Runtime interpreters translate the XML GUI specification into XHTML. However, CelLEST is for mainframe systems with TUIs which are simpler, and less diverse and the screenshots are taken from the same system in different UI states. The generated WUIs are very simple representations of TUIs in the browser, so UI similarity is easier to achieve compared to GUIs. While not officially part of it, the authors of MELIS present an approach for identifying similar Web pages within legacy Web Application to support comprehension for reengineering (Lucia, Scanniello, et al., 2007). The approach is based on clustering using a Winner-Takes-All approach based on a Self-Organising feature Map (SOM). The input for the clustering is the structural model of the page as represented by the abstract syntax tree comprised of HTML tags and scripting expressions. For clustering, distances are calculated using Levenshtein distance of canonical concatenated string representations of the syntax trees. This approach is more static than even the DOM-based page segmentation,

[121] according to `https://www.ailab.microsoft.com/experiments/sketch2code`

[122] the 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`

[123] `https://airbnb.design/sketching-interfaces/`

[124] `https://github.com/tonybeltramelli/pix2code`

and XBI detection approaches, as is only focuses on the server-side of WUIs. The HTML page structure resulting from a server-side scripting language (JSP was used in the experiments) does not represent any dynamic aspects such as computed properties through the use of client-side scripts like CSS and JavaScript. (Cajas et al., 2019) addresses re-arranging of widgets for migration of Web Applications to mobile applications based on Task and Behavior models using Markov heuristics. UWAT+ acknowledges the importance of compliance between the cognitive characteristics of legacy and Web user interface for reengineering (Distante, Scott Tilley, and Canfora, 2006), but does not provide a concrete approach for measuring and ensuring this similarity. Legacy user interaction is taken into account to some extent by mapping business process and Web transactions to create the navigation model, but the similarity of layout is not regarded.

As shown above, consideration of UI similarity in Web Migration is sparse. The few approaches which do, focus on interactions, i.e. the Behavior and Task dimension of UI similarity, disregarding layout and visual aspects, are targeting legacy systems with TUIs or which have already WUIs and do only use static page structure information which is significantly behind the state of the art in other fields like XBI detection or page segmentation.

7.3 Method and Tool

This section presents the AWSM:CI method and tools from the AWSM platform which support measuring the UI similarity of layouts between legacy non-web and Web versions of graphical user interfaces throughout the Web Migration with limited resources and lack of Web Engineering expertise.

7.3.1 Conceptual Model

This section describes the conceptual model of the AWSM:CI method addressing the four aspects of visual UI similarity analysis, calibration, visual GUI segmentation, and integration.

7.3.1.1. Visual UI Similarity Analysis

For Web Migration, a computer-vision based approach is required as shown in section 7.2.2: the multitude of GUI-frameworks in use in desktop legacy systems, at times even within one system, inhibit the use of existing DOM-based analysis. Custom parsers per framework to support mapping onto DOM as metamodel are undesirable due to the related effort, possible GUI model mismatch and increasing heterogeneity of the DOM. With client-side component frameworks such as Angular and React and the increased use of Custom Elements for Web Components, even the DOM structure is becoming more dynamic and heterogeneous, making DOM-based analysis more difficult and error-prone. The limitations of code-based analysis on both the legacy and Web side, i.e. futility of static parsing due to a high dependence on runtime state

and a plethora of platforms and languages, can be addressed through a computer-vision approach (Grechanik, C. W. Mao, et al., 2018). The advantages are that no modifications on the underlying platforms, code or executables are required, and independence on parsing changing semantics of GUI structure and scripts allows for a more generic and uniform solution (Grechanik, C. W. Mao, et al., 2018), thus addressing the applicability requirement in section 7.2.1. The computer-vision based approach relies on visual input only, which enables it to operate on visual input that is closer to human perception.

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 $\{u_{web,1}, \ldots, u_{web,n}\}$ 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 (Kumar, Satyanarayan, et al., 2013) defined through *feature engineering* for a *design mining* (Jahanian et al., 2017; Kumar, Satyanarayan, et al., 2013) process (Bakaev, Khvorostov, et al., 2017b) and used as features, i.e. 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, following the *local descriptors* approach of identifying *regions of interest* $R_u = \{r_1, \ldots, r_l\}$ and extracting visual features for these regions (G. Chechik et al., 2010). There are 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 UIs 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 UI 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, as 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*

$$sim : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R} \tag{7.1}$$

needs to be calculated, similar to the metric learning approach in OASIS (G. Chechik et al., 2010). However, an image-based similarity using edge and color histogram comparisons (G. Chechik et al., 2010) is not applicable due to the conclusive differences particularly in the *Material* dimension of similarity and the non-hold of its *relevance sparsity assumption*: assuming a relevance score of zero for unknown combinations makes sense for arbitrary image comparisons, but not for comparisons between non-web and Web versions of the same GUI. Instead, the similarity function needs to be an approximation (an *Indicator* ISO/IEEE, 2017a) of *perceived UI Similarity* $\mathfrak{S}$ (a *Measurable Concept* (ISO/IEEE, 2017a)), which is influenced by subjective cognitive aspects (Bakaev, Khvorostov, et al., 2017a) and semantic structure of the GUI:

$$\mathfrak{S} : U \times U \mapsto \in \mathbb{R} \tag{7.2}$$

As shown in section 7.2.2, UI Similarity is an active field of research and perceived UI Similarity is not yet well understood. $\mathfrak{S}$ can be empirically measured, e.g. 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 samplesets (*generalisation assumption*):

$$sim(\vec{x}_u, \vec{x}_{u^+}) > sim(x_u, \vec{x}_{u^-}) \implies \mathfrak{S}(u, u^+) > \mathfrak{S}(u, u^-) \tag{7.3}$$

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. Like (J. Kong et al., 2012), this 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) as described in

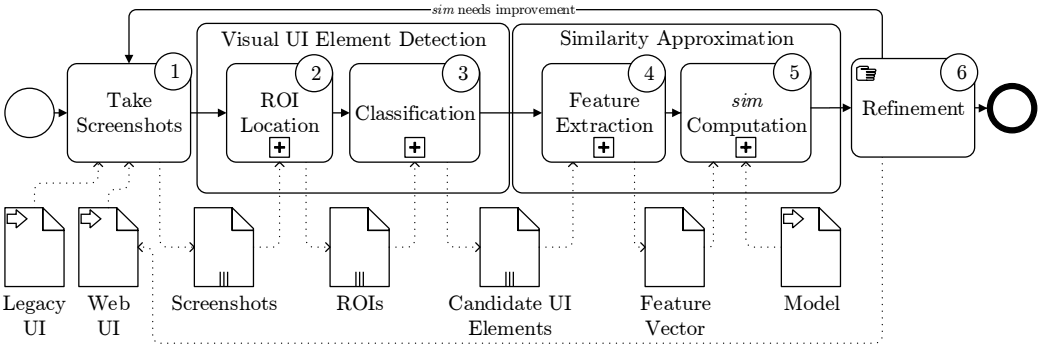


Figure 7.1.: Visual UI Similarity Analysis Process

section 7.3.1.3, followed by the similarity approximation phase based on feature extraction (4) and computation of sim (5) as described in section 7.3.2.1 which can be then used to drive design decisions and refine (6) the WUI and repeat the cycle.

7.3.1.2. Calibration

Visual perception is highly human-dependant and subjective by definition. Experiments show different influence of visual features in perceived UI Similarity $\mathfrak{S}$ 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 7.3.1.1. 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 (Weibo Liu et al., 2017). 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 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 RO4. 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

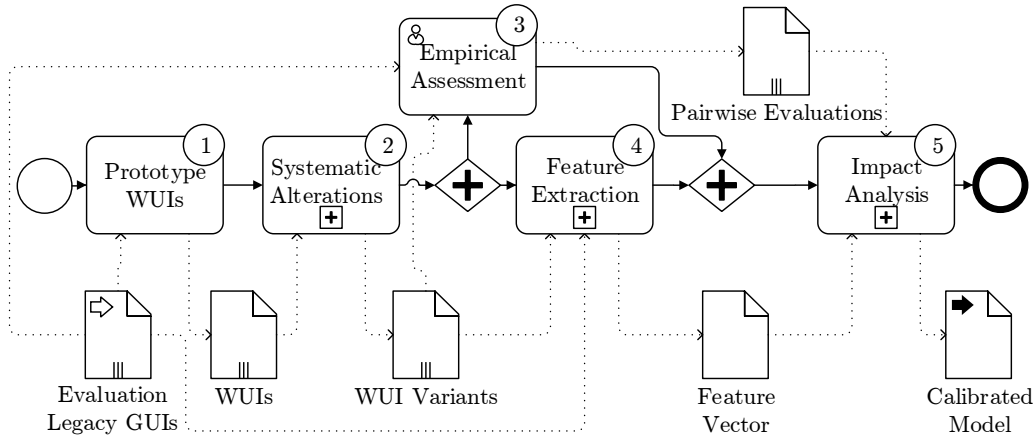


Figure 7.2.: Calibration Process

smaller scope and allows calibration of the model for the perceptual characteristics and viewing conditions, i.e. devices, etc., of the target group.

The calibration process is shown in fig. 7.2. It creates WUI prototypes (1) for a set of evaluation GUIs from the Legacy System, e.g. using AWSM:RM. Then, these WUIs are systematically altered with regard to different visual features, similar to the genome mutation strategy in section 6.3.1.3. Representatives of the target user group provide pairwise similarity evaluations for pairs of one legacy GUI and one WUI variant through empirical assessment (3). This step is equivalent to the evaluation of fitness in section 6.3.1.3. The calibration process uses the same feature extraction (4) like step (4) in fig. 7.1, producing a feature vector. Through combined analysis of the feature vector and the pairwise 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.

Section section 7.3.2.1 describes the implementation of calibration through computation of similarity measures with adjustable weights for different perceptual features.

7.3.1.3. Visual GUI Segmentation

This section describes the conceptual model of steps (2) and (3) of fig. 7.1. The user interface formalism in section 4.3.2 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* (Sanoja and Gancarski, 2014; Cai et al., 2003). 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 (Roy Choudhary, Prasad, et al., 2014b; J. Kong et al., 2012; Kumar, Talton, et al., 2011). 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 (J. Kong et al., 2012). 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 *recognition perspective*, which represents the perspective of a technical solution to approximate the visual perception perspective. Visual GUI Segmentation 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 (Saar et al., 2016) (also called blocks in Sanoja and Gancarski, 2014; Cai et al., 2003) 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). Figure 7.3 provides an overview of the extended conceptual model of user interfaces.

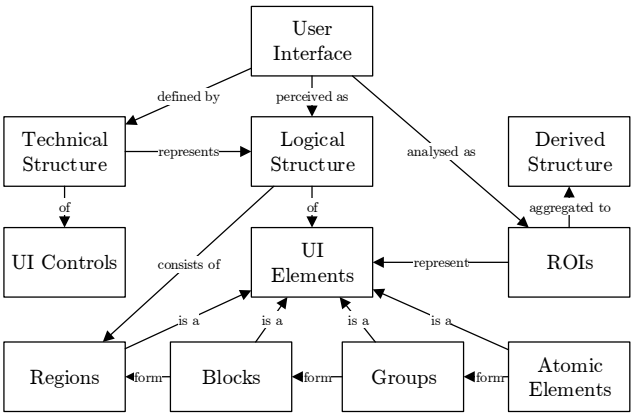


Figure 7.3.: User Interface Concept Map

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 7.2.1. 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. Implementation of the visual segmentation process is described in section 7.3.2.2.

7.3.1.4. Integration

This section describes the integration aspects of the AWSM:CI method.

Integration with ongoing Development

The AWSM:CI method specifies a measurement-based technique 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 (Software Engineering Standards Committee of the and IEEE Computer Society, 1998). Thus, its integration into ongoing forward development activities can be easily achieved. The measurement method 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 (IEEE Computer Society, 2014; Wallmüller, 2001) 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 AWSM: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 (section 7.3.1.1) itself can be started by any migration engineer, the calibration process (section 7.3.1.2) should be executed by staff with some 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 AWSM: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 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.

Integration with existing web migration methods

This section briefly outlines the integration of AWSM:CI with existing Web Migration methods as required by principle P2. AWSM:CI can be integrated with *REMICS* (Mohagheghi and Sæther, 2011). On process level, the AWSM:CI method belongs to the migrate activity area, migration phase (Abhervé et al., 2013). While AWSM:CI does recover layout information from the Legacy System, it cannot be put into the recover activity area, as the nature of similarity measurements requires a comparison between legacy and target UI, with the latter not being available before the migration phase. However, integration of AWSM:CI in REMICS has impact on the pre-migration phase of "establishing the context" (Mohagheghi and Sæther, 2011) which REMICS borrows from SMART (G. Lewis, Morris, Smith, et al., 2008), where availability of AWSM:CI influences the "describe the disadvantages of the migrated solution" activity by providing a means to control disadvantageous customer impact (cf. customer impact risk problem subtree in section 4.1.2). In the agile extension of REMICS (Krasteva et al., 2013), the cyclic nature of fig. 7.1 fits well to the iterative migration scrums. AWSM:CI similarity measurements can provide feedback for iterative transformation and forward engineering implementing incremental refinement (6) in fig. 7.1.

On artifacts level, AWSM:CI integrates well with the the model-driven methodology of REMICS, as it is designed for compatibility (cf. principle P3) with model-driven reengineering approaches: the process in fig. 7.1 represents a model-driven reverse engineering process as it abstracts a CIM, the vector-space representation, from concrete UIs and uses them to compute a similarity measure. With REMICS being based on ADM, the SMM (Object Management Group, 2012) could be used for representing similarity measurement information by defining a custom similarity metric[125]. The analysis is based on the REMICS presentation implementation model, with feedback into the presentation component model *page*. Page is substructured according to MVC, so the relevant part is view, addressing the "graphical organization of the page" (Abhervé et al., 2013). In this way, AWSM:CI is complementary to REMICS, which focuses on the similarity of Task and Behavior, addressing the lack of consideration of Layout similarity.

In ARTIST (Orue-Echeverria et al., 2014), AWSM:CI can be integrated into the Migration phase to fill the gap of lacking consideration of UI similarity. The cyclic refinement integrates well with ARTIST's Modernization-Optimization cycle as specified in Migration Artefacts Reuse & Evolution (Menychtas, Santzaridou, et al., 2013). In particular, the similarity measurement process is relevant for the non-functional re-

[125] metric in SMM terminology, see discussion of measurement vs. metric in section 7.3.2.1

quirements consideration in task NFVALML, activity NFVALML.A2 (Orue-Echeverria et al., 2014), as similarity between the source legacy and target Web user interface is a non-functional requirement. AWSM:CI calibration would require a new task being added to the Target Environment Specification phase, as the resulting calibrated model represents characteristics of the target group. On artifacts level, the calibrated model would be a new ARTIST output artefact of the calibration task, and the similarity measurements would be added as output artefact of NFVALML.A2 and as input artefact for GSCODE.A5, as additional information for the manual code completion.

In *UWA/UWAT+* (Distante, Scott Tilley, and Canfora, 2006; Distante, Canfora, et al., 2006), AWSM:CI belongs to the Forward Design Phase. Following the same argumentation as for REMICS reengineering, the AWSM:CI recovery cannot be achieved in the Reverse Engineering phase prior to the existence of a Web user interface for similarity computation. Within Forward Design, AWSM:CI integrates into the UWA Publishing Design activity. While UWA/UWAT+ focuses on preserving similarity in the Task dimension, integration of AWSM:CI would add consideration of layout similarity. On artifacts level, AWSM:CI similarity measures provide input to the presentation model of the pages specified in Publishing Design (Distante, Canfora, et al., 2006).

In the context of MELIS (Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008) and MIGRARIA (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012), AWSM:CI is a concretization of the objective to achieve a similar "look and feel" (Rodríguez-Echeverría, Conejero, Clemente, et al., 2012; Lucia, Francese, Scanniello, Tortora, De Lucia, et al., 2008; Lucia, Francese, Scanniello, Tortora, and Vitiello, 2006). For the identification of similar Web pages to support comprehension for reengineering (Lucia, Scanniello, et al., 2007), AWSM:CI can be integrated as a parallel alternative to the static parser-based approach, allowing use with more modern WUIs whose visual aspects are determined at runtime at the client side through computations using script languages.

7.3.2 Implementation

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 and the Visual UI Element Detector as part of the AWSM Platform.

7.3.2.1. Computing Visual UI Similarity

Similarity measures play a crucial role in Case-based reasoning (CBR) systems, also known as similarity searching systems (Liao et al., 1998). A similarity measure quantifies the "degree of resemblance between a pair of cases" (Liao et al., 1998), for AWSM:CI these cases are user interfaces. 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) (Liao et al., 1998) in three variations: sim_E based on Euclidean distance:

$$sim_E(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - \text{DIST}_E(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - \sqrt{\sum_{i=1}^{d} w_i^2 \text{dist}^2(x_{u_1,i}, x_{u_2,i})} \tag{7.4}$$

sim_H based on Hamming distance:

$$sim_H(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - \text{DIST}_H(\vec{x}_{u_1}, \vec{x}_{u_2}) = 1 - \sum_{i=1}^{d} w_i \text{dist}(x_{u_1,i}, x_{u_2,i}) \tag{7.5}$$

and sim_R based on Tversky ratio mode:

$$sim_R(\vec{x}_{u_1}, \vec{x}_{u_2}) = \frac{\alpha \times \text{common}}{\alpha \times \text{common} + \beta \times \text{different}} \tag{7.6}$$

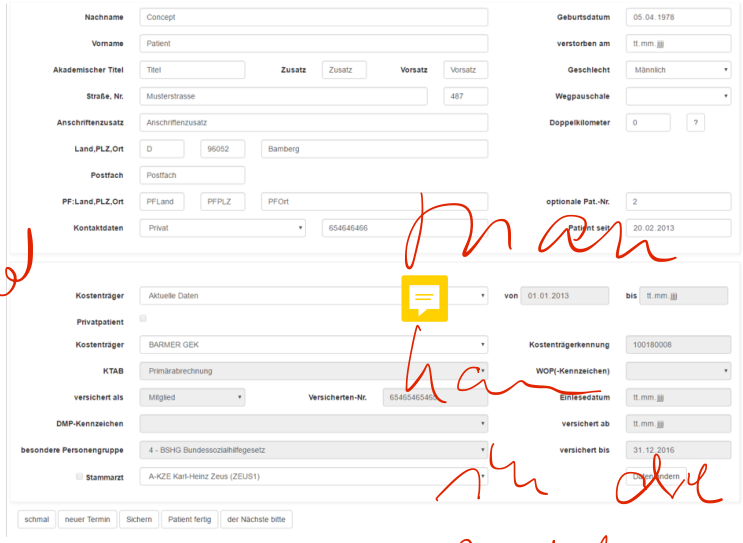
for $\vec{x}_{u_1}, \vec{x}_{u_2} \in \mathbb{R}^d; w_i \in \mathbb{R}^+$ and $\text{common}, \text{different} \in \mathbb{N}_0; \alpha, \beta \in \mathbb{R}^+$.

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 a *pseudo-semimetrics*, i.e. fulfilling positivity, symmetry and the relaxed positive definiteness axiom as one-sided implication $\vec{x} = \vec{y} \implies \text{DIST}_E(\vec{x}, \vec{y}) = \text{DIST}_H(\vec{x}, \vec{y}) = 0$.

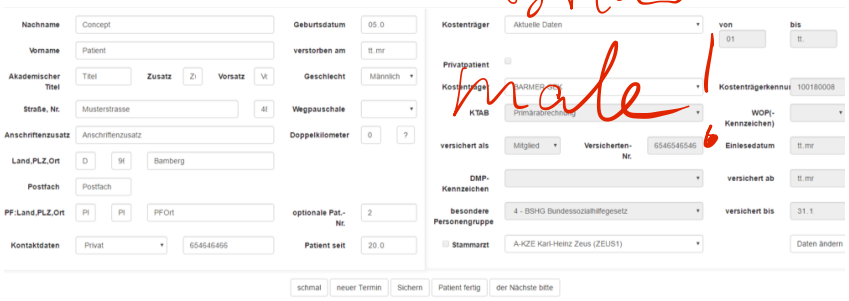
Feature-based normalised distance functions dist define the calculation of DIST_E and DIST_H and the weight factors w_i allow for the calibration process in section 7.3.1.2 to adjust the impact of different features. AWSM:CI uses three distances in the Layout dimension of UI similarity: Orientation, Order and Density (Heil, Bakaev, et al., 2016). All three distances are normalised to percentages using Tversky difference ratios with $\alpha = \beta = 1$:

$$\text{dist}(x_{u_1,i}, x_{u_2,i}) = \frac{\text{different}}{\text{different} + \text{common}} \tag{7.7}$$

These distances represent *base measures* (ISO/IEEE, 2017a) and are computed on the four different levels of the UI hierarchy introduced in section 7.3.1.3, 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 $\{\text{vertical}, \text{horizontal}, \text{other}\}$. Vertical orientation represents aspect ratios of bounding box b (cf. section 4.3.2) with $w/h < \theta_O$ (i.e. portrait), horizontal orientation repre-



(a) Layout



(b) Orientation Variation

Figure 7.4.: Layout Variations

sents 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 7.4 shows an example of a WUI created for section 2.1 and a layout variation with different orientation. *Order* differences can be computed using Levenshtein distances in 8 different orientations on sequences as described in section 6.3.2.2. 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 is based on bounding boxes, calculating the white space ratio (Bakaev, Heil, Khvorostov, et al., 2019; Oulasvirta et al., 2018) W, 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 $W_1/W_2 \in [\theta_D, 1/\theta_D]$. As reasoned in section 7.3.1.1, these 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: $\frac{|E_{u_1} \cap E_{u_2}|}{|E_{u_1} \cup E_{u_2}|} > 1 - \varepsilon$.

7.3.2.2. Visual UI Element Detector

This section describes the implementation of the Visual UI Element Detection subprocess in fig. 7.1. Object detection is among the "most widely known sub-domains in computer vision" (Weibo Liu et al., 2017), solving the problem of precisely locating and classifying target objects in an image (Weibo Liu et al., 2017). Thus, the conceptual process of Visual UI Element Detection 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).

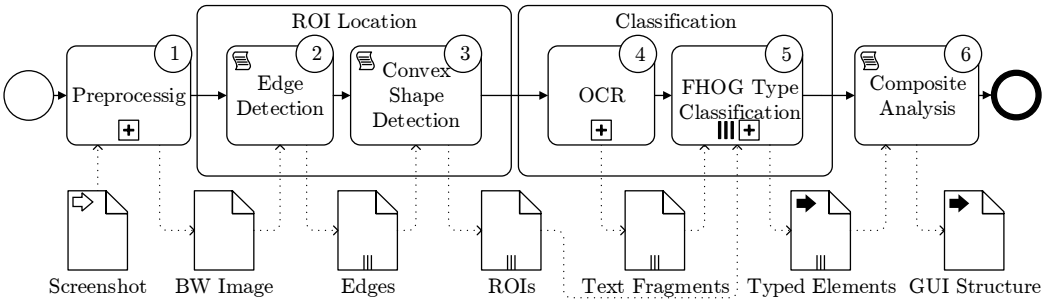


Figure 7.5.: VUE-D Analysis Process

The *Visual UI Element Detector (VUE-D)* is implemented based on the computer vision strategy in (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 platform[126] (Bakaev, Heil, Perminov, et al., 2019). Figure 7.5 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 (cf. Bow, 2002). OpenCV[127] 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

[126] `http://va.wuikb.online/`
[127] `https://opencv.org/`

as boxes b according to eq. (4.11). For *OCR* (4), VUE-D combines OpenCV close edge detection with *Tesseract*[128]. It identifies areas of high-frequency edges, does up-scaling 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 specialised classifiers for different types of UI Elements. Each classifier is trained on one specific type using supervised learning on the Felzenszwalb histogram of oriented gradients (FHOG) latent SVM feature (Felzenszwalb et al., 2010) extractor implemented in dlib[129] which is among the most popular methods for object detection (Weibo Liu et al., 2017) and 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. Figure 7.6 shows application of VUE-D on a GUI screenshot with detection results highlighted.

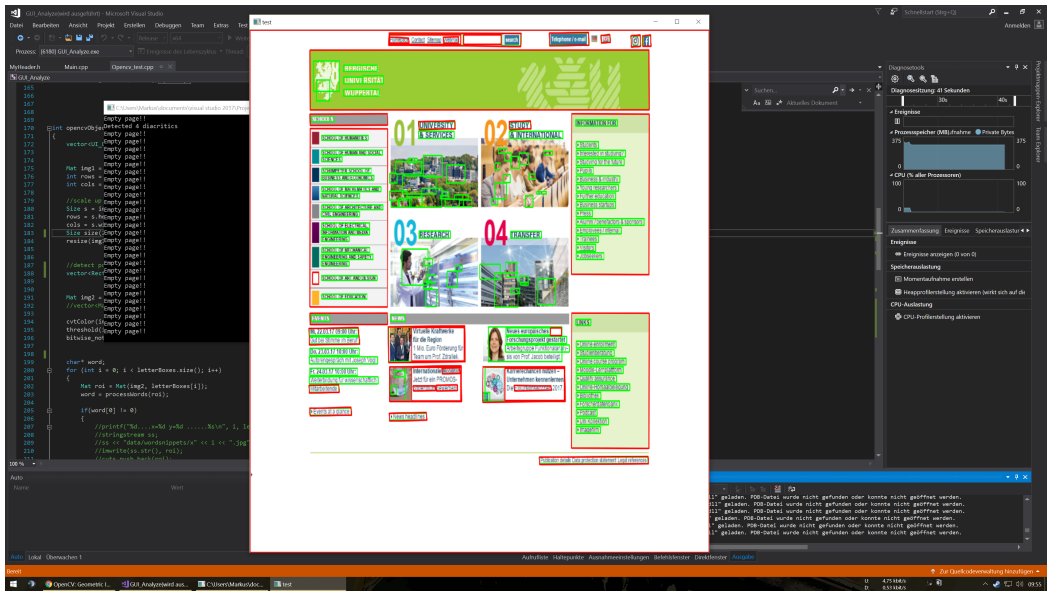


Figure 7.6.: Visually Segmented Screenshot from VUE-D

7.4 Evaluation

This section evaluates AWSM:CI regarding the requirements in section 7.2.1. Additional experimental evaluation further address effectiveness, applicability, and calibratability.

Effectiveness. The effectiveness of AWSM:CI is achieved through specification of a UI similarity between legacy and Web user interfaces that produces concrete

[128] https://github.com/tesseract-ocr/tesseract
[129] http://dlib.net/

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 segmentation 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) presented 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.

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.

Calibratability. The AWSIM:CI similarity measures can be calibrated to the perceptual characteristics of the target group following the calibration process described in section 7.3.1.2. 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.

7.4.1 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 $U_V = \{Vu' | \forall u' \in U_W, \forall V \in \{v_{\text{orientation}}, v_{\text{order}}, v_{\text{density}}\}\} = \{u''_{1,\text{ori}}, u''_{1,\text{ord}}, u''_{1,\text{den}}, u''_{2,\text{ori}}, \ldots, u''_{3,\text{den}}\}$. 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 online[130]. All user interfaces from U_T were functionally identical. No changes were introduced in the Task dimension. For Behaviour, only minor changes through the Web environment exist. Task lists $T_i = \{t_1, \ldots, t_n\}$ of different tasks for each original user interface u_i were specified. The questionnaire shown in appendix G.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

[130] `https://vsr.informatik.tu-chemnitz.de/demos/LayoutSimilarity`

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 7.3.2.1 and table G.1 shows the raw evaluation data. Table 7.2 shows the distances and table 7.3 the similarity measures computed according to section 7.3.2.1. 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 Diff.	dist_{ori}	Order Diff.	dist_{ord}	Density Diff.	dist_{den}
u_1'	1B	0.125	none	0	1G	0.083
$u_{1,\text{ori}}''$	1R, 2B, 2G	0.625	1B	0.167	1R, 1G	0.417

UI	Orientation Diff.	dist_{ori}	Order Diff.	dist_{ord}	Density Diff.	dist_{den}
$u''_{1,\text{ord}}$	none	0	1B	0.167	1B, 1G	0.250
$u''_{1,\text{den}}$	1B	0.125	none	0	2G	0.167
u'_2	none	0	none	0	1B, 1G	0.089
$u''_{2,\text{ori}}$	1R, 1B	0.333	none	0	1R, 2B, 1G	0.603
$u''_{2,\text{ord}}$	none	0	3B	0.333	1B, 1G	0.089
$u''_{2,\text{den}}$	1B	0.083	none	0	1R, 2B, 1G	0.603
u'_3	none	0	none	0	1R	0.333
$u''_{3,\text{ori}}$	1R, 2B	0.417	none	0	1R	0.333
$u''_{3,\text{ord}}$	none	0	4G, 24E	0.357	1R	0.333
$u''_{3,\text{den}}$	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 7.7 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	$\overline{\text{Difficult}}$	$\overline{\text{Like}}$	$\overline{\mathfrak{S}}$
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

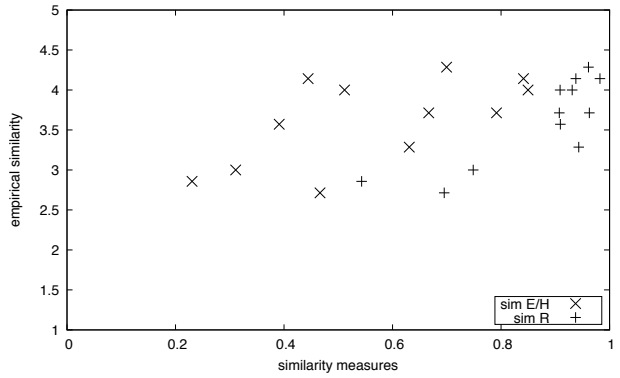


Figure 7.7.: Similarity measures and perceptual similarity scatterplot

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 un-calibrated model with $w_i = 1$. In contrast, the Tversky similarity sim_R does not exhibit a significant relationship. Also, $sim_{E/H}$ shows stronger 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. 7.7.

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. Analysing influence of the three distances in pairwise 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 7.3.1.2, 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.

$$\mathfrak{S} = 4.61604 - 3.24281\text{dist}_{\text{ori}} - 1.27447\text{dist}_{\text{ord}} - 0.888071\text{dist}_{\text{den}} + e[t] \qquad (7.8)$$

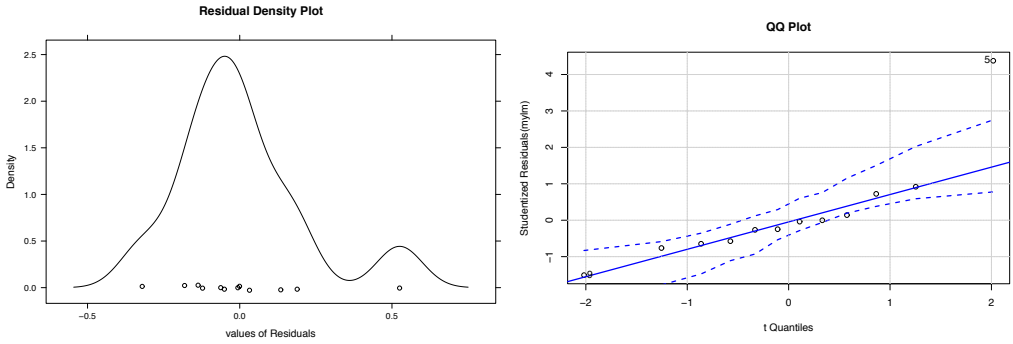
Using the factors of the distances as weights w_i, the similarity measures can be re-calculated as shown in table 7.4. The calibrated similarity measures $sim_{E,\text{cal}}$ and

$sim_{H,\text{cal}}$ improve the correlation with median $\mathfrak{S}$ to $r_s = 0.8383$, $p = 0.0007$, highly significant at $\alpha = 0.001$. Figure 7.8 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. 7.9 visualises the calibrated measures and their corresponding residuals QQ plots.

Table 7.4.: Calibrated sim measures for $w_{\text{ori}} = -3.24281$, $w_{\text{ord}} = -1.27447$, $w_{\text{den}} = -0.888071$, maxima highlighted in bold

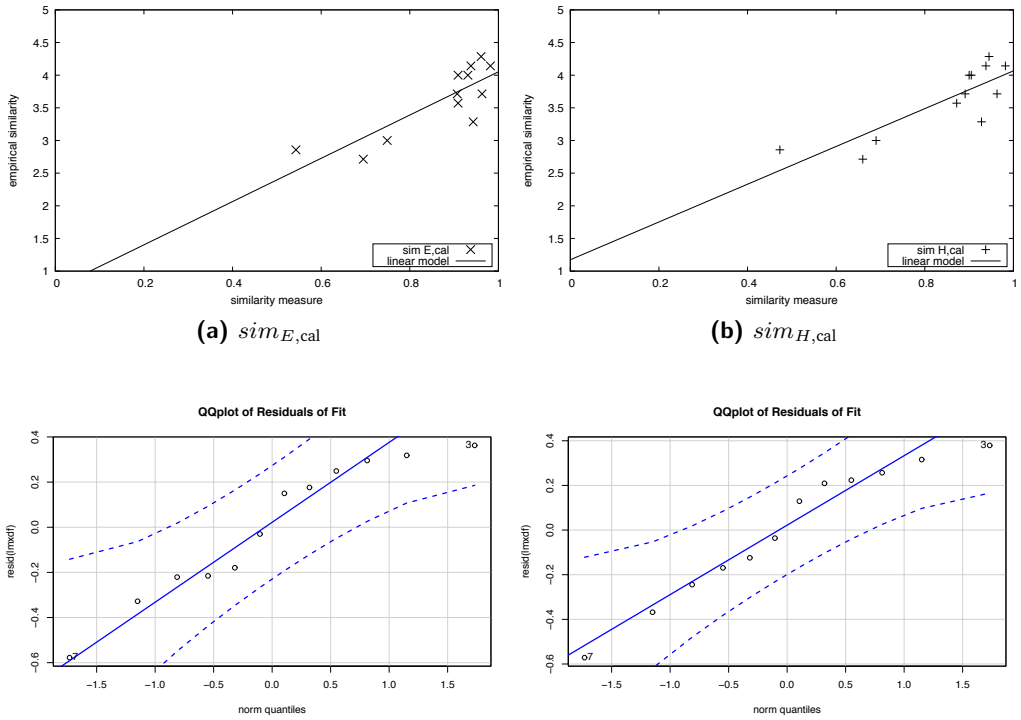
UI	$sim_{E,\text{cal}}$	$sim_{H,\text{cal}}$	$\overline{\mathfrak{S}}$
u'_1	0,9088	0,8999	4
$u''_{1,\text{ori}}$	0,5430	0,4726	2.8571
$u''_{1,\text{ord}}$	**0,9611**	**0,945**	**4.2857**
$u''_{1,\text{den}}$	0,9074	0,8906	3.7143
u'_2	**0,9822**	**0,9822**	**4.1429**
$u''_{2,\text{ori}}$	0,7487	0,6904	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**

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



(a) Residual Density Plot **(b)** QQ Plot of Residuals

Figure 7.8.: Residual Plots, created using (Wessa, 2017)



(a) $sim_{E,\text{cal}}$ **(b)** $sim_{H,\text{cal}}$

(c) $sim_{E,\text{cal}}$ Residuals QQ Plot **(d)** $sim_{H,\text{cal}}$ Residuals QQ Plot

Figure 7.9.: Calibrated Similarity Measures and Residuals QQ plots, created using (Wessa, 2017)

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.

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 ($u'_1 \widehat{=} A3$, $u'_2 \widehat{=} B2$ and $u'_3 \widehat{=} C2$). 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 information 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 perceptual similarity of UI, 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 AWSM: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 AWSM:CI experimentation. The AWSM: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 AWSM: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 perceptual 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 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 and implementation of AWSM:CI have been described. In the evaluation, we have shown 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 perceptual 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 ist wie bei Kap 5 - 6 sehr gut.

Trotzdem bleibe die Synchronisprobleme, die auch in der Language sections fehlen.

Du solltest die Leser besser
mitnehmen. Die Argumentations-
kette ist manchmal
vgl. Pyramide Buch!

Außerdem solltest Du
mehr Bilder einbauen!

⇒ Du kannst doch
so viele tolle Sachen
zeigen, nutze doch
Deine Möglichkeiten!

8 Evaluation

This chapter evaluates the AWSM Methodology and Platform according to the requirements stated in section 2.2. This assessment is followed by a comparison to the state of the art in terms of benefits and weaknesses.

8.1 Requirements Evaluation

In the following, the AWSM Methodology and Platform are assessed with regard to the scope and stakeholder requirements of this thesis.

8.1.1 Scope Requirements

The following scope requirements are evaluated to determine to what extent the AWSM Methodology and Platform address the scope set by the main research question RQ1, 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

The AWSM Methodology and Platform dedicatedly addresses the initial phase of Web Migration prior to actual transformation and beyond knowledge recovery through AWSM:RM. 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 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 fulfilled.

S2 Web Application Target

The AWSM Methodology and Platform has Web Applications according to Definition 2 as target architecture. AWSM 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 AWSM:RM prototypes and the analysis of web user interfaces in AWSM:CI. The target web-based user interfaces 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, AWSM: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 AWSM Platform itself are implemented based on open Web standards (cf. principle P1). According to the assessment scheme, the requirement is fully fulfilled.

8.1.2 Stakeholder Requirements

The following stakeholder requirements are evaluated to assess appropriateness of the AWSM Methodology and Platform for the characteristics of an independent software vendor as detailed in section 2.1.

C1 Risk management

The AWSM Methodology and Platform provides risk management methods beyond basic feasibility-centric approaches through its AWSM:RE and AWSM:RM methods. AWSM:RE specifies a concept-assignment-based knowledge recovery approach which addresses the risk of losing knowledge implicitly represented by the legacy source code during Web Migration. The systematic application of AWSM:RE reverse engineering extracts knowledge from the source code and maintains it for further management and usage in an interoperable way based on the SCKM formalism and a queryable Web standards-based knowledge representation. Following AWSM principle P4, AWSM:RM specifies a rapid-prototyping-based method for creating demonstrative web migration prototypes that allows for identification of migration process and migration result risks. 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. According to the assessment scheme, the requirement is fully fulfilled.

C2 Re-use of legacy assets

The AWSM Methodology and Platform addresses re-use of legacy assets at three different levels, covering the model, view and controller layers of an application: AWSM:RE focuses on re-use on the model and requirement level, AWSM:RM on the business logic and UI level and AWSM:CI on the view and user interaction level. In this way, the three AWSM methods establish continuity of functionality (AWSM:RE and AWSM:RM) and user interaction (AWSM:RM and AWSM:CI) across legacy and target Web System. The re-use of legacy models and requirements is achieved through AWSM:RE by reverse engineering these assets implicitly represented in the codebase into explicit artifacts thus enabling their re-use in both model-driven and non-model-driven subsequent processing. The re-use of business logic and the user interface is achieved through AWSM:RM by semi-automatic (ReWaMP) / automatic (UI Transformer) transformation based on legacy code artifacts. The re-use 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 KDM-based Legacy System, SCKM and Legacy User Interface formalisms provide the conceptual foundation of re-use in AWSM. According to the assessment scheme, the requirement is fully fulfilled.

C3 Expertise & Tool Support

The AWSM Methodology and Platform addresses expertise and tool support requirements through its method design integrating the non-functional expertise requirement for all three AWSM methods and the tools of the AWSM Platform respectively. AWSM:RE addresses the expertise requirement by re-using program comprehension results from ongoing forward engineering and by automatic decomposition of Concept Assignment into microtasks solved using crowd expertise. AWSM:RE tool support comprises the annotation tool, its CSRE extension, and the tools for integration with ongoing development. AWSM:RM addresses the expertise requirement by enabling business logic re-use 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. 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. The AWSM Platform 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 fulfilled, acknowledging that Web Migration activities can hardly be conducted without any additional expertise requirements.

C4 Agile Development Process Integration

The AWSM Methodology and Platform addresses development process integration through its method design specifying the conceptual integration of each AWSM method into ongoing development activities and the integration support tools of the AWSM Platform. AWSM principle P2 avoids specification of a stand-alone process: the three AWSM methods targeting the initial phase can be integrated easier than full-coverage Web Migration approaches. AWSM principle P3 facilitates integration with a variety of different model-driven or non-model-driven development processes. AWSM:RE integrates with ongoing development through continuous reverse engineering, embedding Concept Assignment with forward engineering in the comprehension phase, re-using the mental representation. This is supported by the IDE integration tool. AWSM:RE integrates Web Migration project management with software project management via the migration package/migration backlog formalism and the corresponding project management tool integration. AWSM: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 fulfilled, 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.: Evaluation of AWSM in Comparison to State of the Art, requirements S1 Initial phase, S2 Web Application Target, C1 Risk management, C2 Re-use 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	◐	◐	◐	◐	○	○
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	●	●	●	●	◐	◐

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 4.4). 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, as specified in AWS Migration, ARTIST, REMICS, SMART and SAPIENSA, and integration of the Rapid Web Migration Prototyping method has been defined in section 6.3.1.4. AWSM's S2DCS supports the Strategy Selection (Harry M Sneed et al., 2010b) which is an essential step in the initial 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.

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 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 AWSM, 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 AWSM: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, AWSM: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 AWSM: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 AWSM:RM in a similar way. Like ARTIST, AWSM uses extensive knowledge recovery to address the risk of knowledge loss. While AWSM is agnostic to the specific nature of target knowledge representations, enabled by AWSM's KDM-

based SCKM and the queryable knowledge representation of AWSM:RE, ARTIST's extensive model-driven knowledge recovery can be supported by AWSM:RE as described in section 5.3.1.3.

Re-use of functionality is expectedly high across all Web Migration approaches — all approaches but AWS Migration and PRECISO feature functionality re-use— as this is a definitive property of any software migration approach. The distinctive property is re-use-based continuity of user interaction. AWSM addresses re-use on all three application layers as outlined above in order to maintain functionality and user interaction. A similarly high level of re-use 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 3.4) 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 re-use. 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 complimentary and can be integrated in 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 re-use, as they only consider re-use of functionality, but can be easily extended with AWSM techniques due to their methodological framework structures. The integration of AWSM:CI with UWA/UWAT+, REMICS and ARTIST is described in section 7.3.1.4.

Concerning expertise and tool support, AWSM only receives a half rating, conceding that in spite of the extensive tool support of the AWSM platform, 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 TUIMigrate 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 platform 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 Platform provides several dedicated integration 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 scoring[131], where AWSM ($\Sigma = 5$, $\overline{S} = 0.833$) is rated higher than the top-ranking approaches in section 3.2 SMART ($\Sigma = 3.5$, $\overline{S} = 0.583$), REMICS ($\Sigma = 3$, $\overline{S} = 0.5$), ARTIST ($\Sigma = 3$, $\overline{S} = 0.5$) and UWA/UWAT+ ($\Sigma = 2.5$, $\overline{S} = 0.417$). AWSM, however, does not directly compete with complete migration processes as defined for instance by REMICS, ARTIST or UWA/UWAT+. Instead, similar to SMART, AWSM is a *complimentary methodology* providing a set of principles, formalisms, and methods supported by the tools of the AWSM Platform which address aspects of Web Migration which 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

[131] based on a mapping S of the three possible ratings as follows: ○↦ 0, ◐↦ 0.5, ●↦ 1

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. 4.4). This usage intention is visible in the AWSM principles, the method's design and the consistent specification of integration for the most relevant and comprehensive Web Migration approaches of all three AWSM methods.

8.3 Summary

While previous sections evaluated the AWSM methods and techniques in isolation, this section analyzed the extent to which the AWSM Methodology and Platform as a whole fulfills the requirements for a Web Migration solution addressing the shortcomings of existing approaches as elicited in section 2.2. 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 section 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.

9 Conclusion and Outlook

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. We detailed the situation of medatixx as representative sample of 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 this, requirements for an appropriate solution have been elicited. Analysis of the state of the art in the context of 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 characteristics of SME-sized ISVs, despite academia having shifted focus away from Web Migration. Based on field research and the application of LFA and HCD methods, a research and solution design was derived which defines research objectives to address ISV's doubts about feasibility and desirability.

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 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 a shortcoming of existing approaches that contributes to the 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 Platform 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 facilitates recovery of problem and solution domain knowledge from the legacy codebase for ISVs with limited resources 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 facilitates the demonstration of feasibility and desirability of Web Migration and the plausibility of a web-based version of the Legacy System through a novel Rapid Web Migration Prototyping strategy. The AWSM Platform supports this strategy with a WebAssembly-based toolchain and a guidance and automation system. The AWSM Customer Impact method facilitates the measurement and control 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 Platform has been evaluated in five 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 Platform were consistently found useful by the test subjects. The main identified problems were task complexity, addressed by an additional guidance and automation system, and the non-mature nature of understanding perceptual similarity of user interfaces, which is still a young field with ongoing research. Analysis of the the AWSM Methodology and Platform 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 Platform 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 Crowdsourcing (Heil, Siegert, et al., 2019; Heil, Förster, et al., 2018) and Rapid Prototyping (Heil, Siegert, et al., 2018) 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 6.4.2).
- Perceptual 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 Contributions

This thesis produced models, methods, techniques and architectures that support ISVs to commence Web Migration. The following list summarizes the 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 main 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 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 platform architecture integrated with ISV's software infrastructure.
- Definition of a queryable open Web standards-based representation of knowledge in legacy codebases through ontological modeling 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 perceptual 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 Platform 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 with 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 Platform 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 Platform Improvements

The AWSM Methodology and Platform 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 $l = (s, f)$ (cf. eq. (4.4)) 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 to put more emphasis on the extraction process from extension (K. Chen and Václav Rajlich, 2010) l to intension (K. Chen and Václav Rajlich, 2010) 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 7.3.2.2. Integration with comprehensive Web Migration approaches as described in section 5.3.1.3 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 $\mathfrak{L}$. 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. **??** 5.1 presents a simple anonymization technique but AWSM:RE would benefit form a more sophisticated algorithm that improves crowdworkers' code comprehension while maintaining the three anonymization properties defined in section 5.3.2.4. 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 considered. Agreement measures can be used to 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 votes[132] better, e.g. using Fleiss Kappa, and define an extended confidence-based quality control mechanism.

[132] 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

Visualizations as external representation of Knowledgebase $\mathbb{K}_B$ as described in section 5.3.2.1 facilitate achieving the high level of code comprehension required for Web Migration. AWSMAP provides different progress statistics as shown in fig. 5.9 and visualizations of concept relationships: Sankey diagrams as shown in fig. 5.10, Force Graphs and Chord diagrams which represent the interlacing between artifacts and features or features and features, due to particular interest in this issue from medatixx. 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)[133], WASM is becoming an increasingly relevant technology no longer limited to the client side[134]. For the WASM-based infrastructure of AWSM:RM, support for POSIX-like system functions in the browser[135] can be used to increase the capabilities of ReWaMP prototypes and further reduce the manual code adaption effort, since fewer dependencies Dep of $\mathfrak{L}$ will require expert changes due to higher availability of libraries and system functions. To take advantage of the flourishing WASM environment, a review of the ReWaMP process and the 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 6.4.3. 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 6.4.3.

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 improving detection rate through application of the multi-agent system (MAS) paradigm

[133] `https://wasi.dev/`
[134] cf. `https://github.com/CraneStation/wasmtime/`
[135] currently available as polyfill: `https://wasi.dev/polyfill/`

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 7.3.2.2, 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 perceptual 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 3.4, poses a significant challenge. The AWSM Methodology and Platform 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 5.3.1.4 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.is[136]. An interesting research challenge is to investigate how reverse engineering using the W3C Web Annotation

[136] `https://hypothes.is/`

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 Solid[137] project.

WebAssembly at its current stage[138] was used as a technological enabler for Rapid Prototyping in the Web Migration domain based on the ideal of re-use and the available ISV staff expertise. With the recent upswing of WASM and its increasingly rich environment - including package management[139], development environment[140], server-side runtimes[141] and integration with existing technologies like Microsoft's Razor Engine[142] - 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 re-use 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 5.3.1.3. Thus an important research challenge is to investigate advances in Web Migration through transfer of knowledge from other domains (cf. also to HCD's *Analogous Inspiration* method (IDEO, 2015)), 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, 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

[137] `https://solid.mit.edu/`

[138] our 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

[139] `https://wapm.io/`

[140] `https://webassembly.studio/`

[141] `https://wasmer.io/`

[142] `https://blazor.net/`

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 collaboration[143] 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) project[144] follows a similar objective and agenda. Understanding and analyzing visual perception of user interfaces is in its early stages and provides various research opportunities.

[143] http://www.wuikb.online/en/platform/about
[144] https://interfacemetrics.aalto.fi/

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

AWSM:RE Materials

B

B.1 SCKM Ontology

The following is an extract from the SCKM Ontology[145] in Turtle notation. Since SWRL-Rules are very verbose in Turtle, they are shown separately in the following section.

Listing B.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#> .
@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 ,
```

[145] available from the AWSM project page https://vsr.informatik.tu-chemnitz.de/projects/2016/awsm/

```
                    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> .
```

B.2 SCKM Ontology SWRL Rules

The following is an extract from the SCKM Ontology comprising its SWRL-Rules in SWRL Human Readable Syntax[146].

Listing B.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) ^ 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) ^
```

[146] cf. https://www.w3.org/Submission/SWRL/

```
    ↪ 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

C

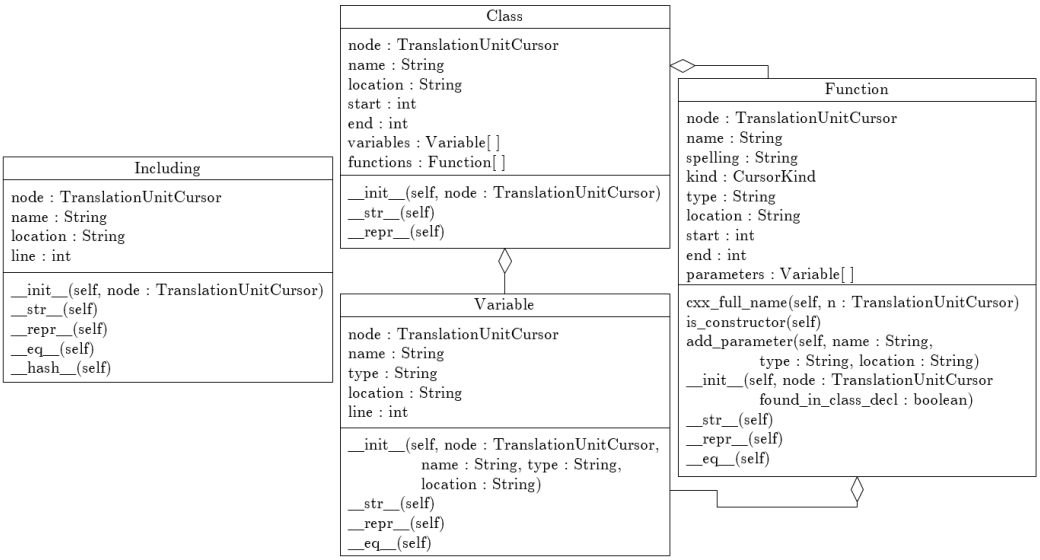


Figure C.1.: ReWaMP C++ parser classes

Table C.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 *data-rwmpId*.
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.
Public Class Fields	All variables declared in a class are public accessible without getter or setter.
1 DoDataExchange	There is only one method DoDataExchange() in one view, and it is located in the main file. Further the only parameter is always called as in the MFC definition 'pDX'.
::GetFocus	In legacy code '::GetFocus()==' or '::GetFocus()!=' is only used in if-clauses.
.Format	'.Format' only appears at CString variables with '%d'.

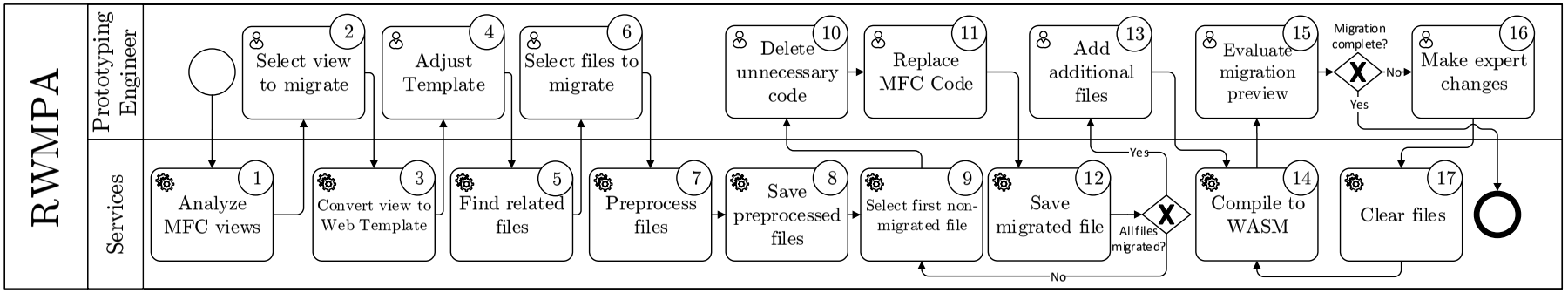


Figure C.2.: RWMPA Process

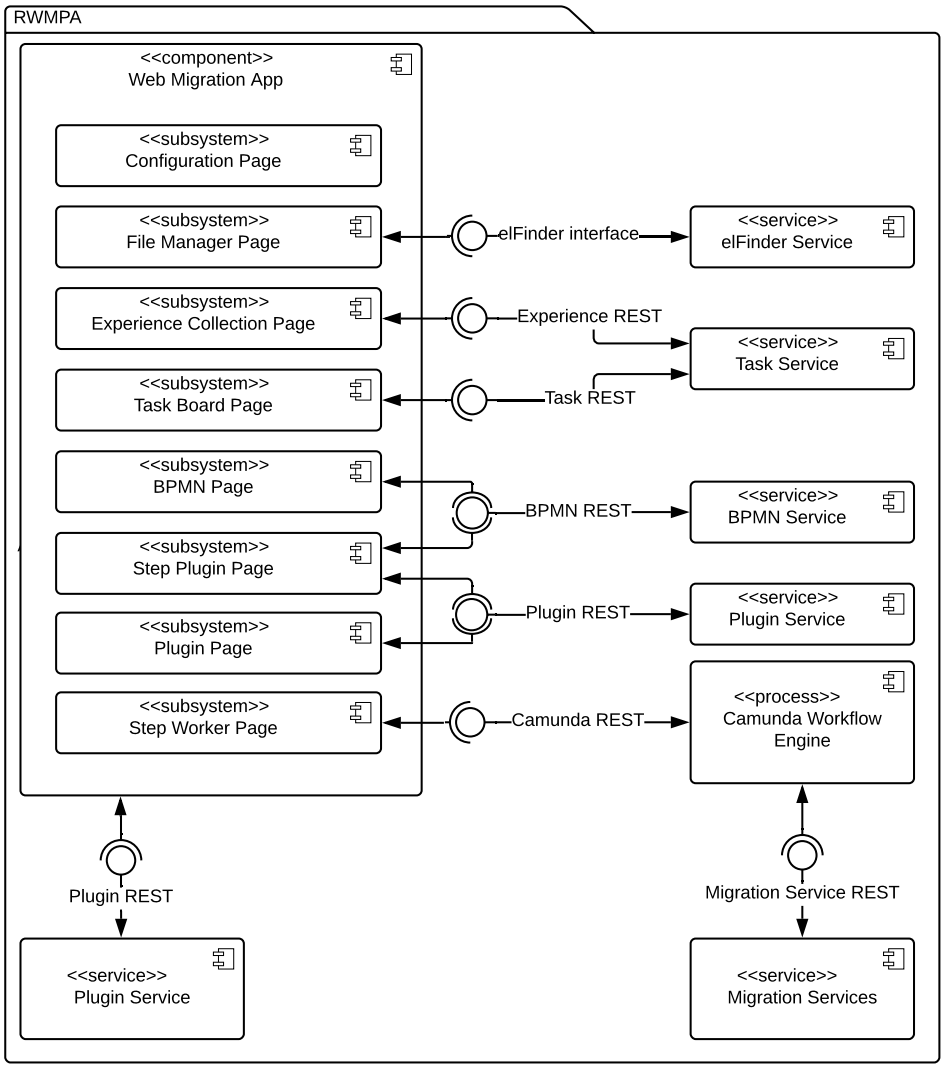


Figure C.3.: RWMPA Architecture

ReWaMP Evaluation Materials

D

This appendix contains additional materials used in the ReWaMP evaluation experiments of this thesis.

Table D.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 program- ming 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.

D.1 ReWaMP Questionaire

The following questionaire was used in the experimental evaluation of ReWaMP in section 6.4.1. Its structure and order of items represents the Google forms questionaire as seen by the test subjects. The specific questions used in the results analysis in section 6.4.1 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

D.2 ReWaMP Evaluation Data

Table D.2 contains the raw evaluation data of the empirical evaluation of ReWaMP, table D.3 the raw evaluation data from time measurements and table D.4 from effort measurements.

latex:fix pagebreak: longtable should just break

Table D.2.: ReWaMP Full Empirical Evaluation Data

Item	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 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
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

Questionaire responses, S1-S4 Self Assessment, F11-F25 Functionality, Q1-Q6 Quality, W1-W11 WASM-T, Abbreviations: Y - Yes, N - No

Table D.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 D.4.: ReWaMP Full Effort Evaluation Data

e^c_{31}	e^u_{31}	e^d_{31}	e^c_{32}	e^u_{32}	e^d_{32}	e^c_{41}	e^u_{41}	e^d_{41}	e^c_{42}	e^u_{42}	e^d_{42}	e^c_{51}	e^u_{51}	e^d_{51}	e^c_{52}	e^u_{52}	e^d_{52}	e^c_{61}	e^u_{61}	e^d_{61}	e^c_{62}	e^u_{62}	e^d_{62}
9	81	60	12	32	43	3	11	45	1	33	94	9	0	0	60	0	0	273	1	0	164	9	4
9	81	60	12	32	43	4	10	43	2	30	96	6	0	0	57	0	0	263	1	0	154	9	4
9	81	60	12	32	43	2	10	37	0	36	93	9	0	0	60	0	0	273	1	0	164	9	4
9	81	60	12	32	43	1	14	37	0	36	89	6	0	0	57	0	0	263	1	0	154	9	4
9	81	60	12	32	43	2	14	35	0	30	88	6	0	0	57	0	0	263	1	0	154	9	4
9	81	60	12	32	43	2	6	35	0	32	90	6	0	0	60	0	0	263	1	0	164	9	4

Measured efforts in SLOC for tasks 3-6, e^a_{ij} indicates effort for action a on task i on view j, actions can be c - create, u - update, d - delete

RWMPA Evaluation Materials

E

This appendix contains additional materials used in the RWMPA evaluation experiments of this thesis.

Table E.1.: RWMPA 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 RWMPA 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 RWMPA workflow: *evaluate migration using a preview*.
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.

E.1 RWMPA Questionaire

The following questionaire was used in the experimental evaluation of RWMPA in section 6.4.2. Its structure and order of items represents the Google forms

questionaire as seen by the test subjects. The specific questions used in the results analysis in section 6.4.2 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 single-choice items. The original questionaire 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.

E.2 RWMPA Evaluation Data

Table E.2 contains the raw evaluation data of the empirical evaluation of RWMPA, table E.3 the rwa data of the time measurements.

Table E.2.: RWMPA Full Empirical Evaluation Data

S1	S2	S3	S4	S5	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	T1	T2	T3	T4	T5	T6
7	2	2	1	Student	5	2	4	5	4	1	5	4	16	Adq.	5	4	5	1	2	1
10	2	3	1	Student	4	3	4	3	4	2	5	5	11	Adq	5	4	4	1	2	2
9	2	3	2	Researcher	3	3	4	4	4	1	4	3	11	Adq	3	3	3	4	3	1
15	3	2	1	SW Architect	4	3	3	4	4	3	2	5	16	TH	3	3	5	1	2	1
15	5	2	1	SW Engineer	3	4	3	3	3	1	4	3	16	Adq	4	4	5	2	2	2
9	2	2	1	Student	5	2	5	4	5	1	5	5	11	Adq	5	4	5	2	1	1
7	3	1	1	Studen	4	3	5	4	4	2	4	4	16	Adq	4	4	5	3	2	2

Questionaire responses, S1-S5 Self Assessment, W1-W10 Workflow, T1-T6 Tasks, Abbreviations: SW - Software, ADQ - Adequate, TH - Too high

Table E.3.: 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	t6.2	t10.2	t11.2	t13.2	t15.2	t16.2
255	131	90	273	1298	150	284	121	11	37	17	34	775	22	190	704
274	20	110	125	1525	41	155	97	3	2	56	56	503	25	212	745
275	58	36	187	1153	50	129	118	4	58	63	29	378	16	110	803
334	94	86	123	1093	66	258	105	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
230	270	183	333	1150	97	435	141	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
283	135	127	280	1703	241	284	249	13	22	13	12	697	10	280	959
227	123	114	179	1510	283	210	352	3	21	82	8	651	17	207	1045

Times for manual tasks in seconds, ti.j - time for task i on view j, view 1: u_{cal}, view 2: u_{new}, N/A - not available

F

UI Transformer Evaluation Materials

This appendix contains additional materials used in the UI Transformer evaluation experiments of this thesis.

F.1 UI Transformer Questionaire

The following questionaire was used in the experimental evaluation of RWMPA in section 6.4.3. Its structure and order of items represents the Google forms questionaire as seen by the test subjects, the numbering of items is equivalent to the numbers used in the results analysis in section 6.4.2. 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 questionaire 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.

F.2 UI Transformer Evaluation Data

Table F.1 comprises the raw evaluation data of the performance evaluation and table F.2 of the empirical evaluation of UI Transformer.

Table F.1.: UITransformer Performance Evaluation Data

i	$\#c$	d	t_1	t_2	t_3	t_4	t_5	t_6	t_{6c}	t_7	t_8	t_9	t_{10}
1	23	0	52991	68519	104155	109407	141808	39902	935	124686	112758	125425	203952
2	3	0	16279	15478	28198	36613	45745	12307	82	37245	27874	36292	57693
3	20	1	54291	65815	100514	103873	124564	39784	1217	123189	113321	115981	172955
4	10	1	34085	33375	51327	56723	83637	25141	506	68264	54769	65621	100792
5	24	1	70494	93683	136509	138137	225771	49583	2040	171170	128068	158620	208725
6	16	0	43013	46811	72981	89444	132452	31102	597	97152	74642	88178	142336
7	17	0	45297	52942	74969	77915	108202	32471	620	88533	74403	83224	136986
8	11	0	34845	36229	57637	63544	89369	24338	361	73976	59159	55261	111850
9	9	1	30242	36700	60202	66611	108856	23185	436	77622	49378	75228	119490
10	44	0	90796	120752	172615	164410	280721	72676	2647	206337	169370	184588	321439
11	21	0	54963	68549	90554	104358	171898	38218	850	114961	102480	101643	178151
12	8	0	27163	26802	41923	50756	73983	19923	231	57956	41112	55115	95696
13	13	0	37060	39273	61597	64808	116680	26815	419	84170	65159	73083	128390
14	34	1	89702	126699	174827	182356	293741	66420	3239	238330	178803	216341	302040
15	18	0	47694	53825	77801	83897	133893	34807	689	100739	77360	80813	165599
16	18	0	47282	55174	76268	86177	127002	34545	674	101801	78763	89394	142913
17	17	0	45746	53665	81786	83927	134995	33211	623	103131	74524	93218	156199
18	26	0	66019	80763	117248	121745	179148	46401	1136	146010	116134	122263	224872
19	15	0	40564	46082	63594	70852	110924	30169	524	83245	64305	76411	128667
20	45	0	104418	148559	204541	211551	355712	74151	2645	269510	194894	227905	374662
21	25	0	65858	76207	111911	112844	186624	46118	1088	138377	96956	115449	205943

i	$\#c$	d	t_{1}	t_{2}	t_{3}	t_{4}	t_{5}	t_{6}	t_{6c}	t_{7}	t_{8}	t_{9}	t_{10}
22	17	1	54653	66542	97983	105352	148268	41912	1691	114799	92023	90833	169773
23	9	0	28687	28396	44376	48115	71211	20889	273	55916	42943	50188	87356
24	15	0	41959	46156	71397	71739	116134	30523	532	89160	63933	73353	144680
25	24	0	63211	72244	122268	111309	162568	43332	1051	137168	103691	118977	214945
26	12	0	35509	39687	58831	62721	112413	25747	391	72834	57766	75042	128375
27	7	0	24856	23416	41805	45161	60975	18517	204	53822	39313	46733	79825
28	23	0	58854	67693	114172	107139	176296	42199	984	117695	94551	107629	229665
29	2	0	13826	11275	19827	26228	32520	11046	62	26766	20692	24792	42196
30	18	0	45411	53378	82518	83866	129468	34244	661	105192	80027	95523	182402
31	4	0	18772	17917	30647	37269	49093	14107	107	38928	30270	37001	63937
32	13	0	36239	39781	61251	70590	99885	27315	454	77543	59531	67263	119802
33	15	0	39338	44354	61204	68855	118620	30102	525	82338	65778	71539	138199
34	5	0	20911	18955	33715	40593	60701	15633	126	43591	35030	38422	65649
35	33	0	78749	98299	127206	150440	200350	57718	1662	163958	124326	164820	249250
36	33	0	79973	110052	141222	150619	250577	57913	1711	196452	144534	167886	287207
37	44	0	101350	134914	205079	182565	272043	73661	2756	232212	172857	251890	347843
38	26	1	64573	90133	125346	127640	166848	47770	1675	155965	133154	147034	238048
39	130	0	343834	454761	532608	519752	1040988	312285	15843	745364	635362	663514	1165719
40	44	0	99871	139913	182475	191001	331185	72570	2651	242200	186781	209142	355776
41	14	0	39149	42180	65693	74368	114644	28589	477	85711	62154	70282	141866
42	59	0	132403	196759	265300	274095	484474	101199	4297	370666	258822	307986	499852
43	17	0	45233	52882	80782	87955	134987	33102	624	97901	78447	89362	148623

i	$\#c$	d	t_1	t_2	t_3	t_4	t_5	t_6	t_{6c}	t_7	t_8	t_9	t_{10}
44	9	1	32129	35214	54621	62866	85028	23750	450	72674	49610	66375	122605
45	19	1	60210	84253	125042	123651	200841	44536	1996	150111	111212	149800	179487
46	30	1	85789	122708	179650	178399	302888	62849	3394	220465	175241	202110	255030
47	18	2	56963	75345	106238	114095	191564	43156	1759	147362	108226	133777	221927
48	24	1	70606	101166	135809	145739	158754	54125	2549	180167	138400	170755	290642
49	19	2	63455	86677	136162	140005	241658	47643	2315	178030	142071	164210	247198
50	22	1	66127	79544	109946	111316	181482	49496	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 F.2.: 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
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

G

This appendix contains additional materials used in the AWSM:CI evaluation experiments of this thesis.

G.1 Visual Similarity Questionaire

The following questionaire was used in the experimental evaluation of AWSM:CI in section 7.4.1. Its structure and order of items represents the printed questionaire 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?
   - A0[147]
   - 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)]

[147] this example shows the question for user interface u_1 encoded as A

3. 𝔖 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?

G.1.1 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

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 Moritz[148] 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, assign[149] Stammarzt Arzt Vier

G.2 Visual Similarity Evaluation Data

Table G.1 contains the raw evaluation data of the evaluation experiment of AWSM:CI.

[148] this task involved searching for the correct patient by name
[149] involves enabling selection by checking checkbox Stammarzt

Table G.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^*	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^*
3	2	2	3	2	2	4	4	4	1	5	4	0	3	1	2	1	5	4	2	4	2	2	3	4	0	4	2	2	3	1	2	1	5	5	1	4	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	2	2	3	2	3	3	2	2	4	2	3	4	1
1	2	5	1	3	5	1	2	5	1	2	3	1	1	2	5	3	1	2	2	2	4	1	3	3	3	3	1	5	4	1	2	4	1	4	1	2	4	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	2	4	3	3	2	3	1	4	4	1	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	1	4	4	1	4	4	2	3	3	1	4	4	2
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	1	5	5	2	2	3	1	4	5	1	3	5	2
2	4	4	2	2	2	2	3	4	3	1	2	2	3	3	2	4	1	2	2	4	2	3	2	2	2	2	4	4	5	1	2	4	2	3	2	4	4	2

Questionaire 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, 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).

Asset An 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), (cf. *intangible assets* ISO/IEEE, 2017b, entry 3.261)..

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).

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 5.

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 5.2.2.

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 6.

Erosion of Soft Knowledge The loss of soft knowledge "due to factors such as ageing, retirement, and staff changing jobs" over time (Khadka, Batlajery, et al., 2014).

Hybrid Web Application A Web Application that consists of parts of re-used legacy code, a transformed web UI and a generated RESTful API, cf. section 6.3.1.1..

Legacy Modernization see 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.

Prototyping see 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 8.

Rapid Web Migration Prototyping An adapted form of Rapid Prototyping transfered into the Web Migration domain, allowing to rapidly create web migration prototype, cf.Section 6.3.1.1.

Risk Management An "organized process for identifying and handling risk factors" (ISO/IEEE, 2017b, entry 3.3528).

Soft Knowledge The "existing knowledge in the form of skills and experiences within the technical staff (original developers, maintainer, users)" (Khadka, Batlajery, et al., 2014).

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).

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).

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 7.

Target Environment The specific environment for which a software is modified in Software Migration.

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), defined in Definition 4.

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.

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).

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).

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 6.3.1.1..

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.

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).

Bibliography

Printed References

Abhervé, Antonin, Andrey Sadovykh, Satish Srirama, Pelle Jakovits, 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. ISBN: 951–38–6009–4. 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. ISBN: 9781450325233.

Alavi, Maryam (June 1984). "An assessment of the prototyping approach to information systems development". In: *Communications of the ACM* 27.6, pp. 556–563. ISSN: 00010782.

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. ISSN: 0098-5589.

Allahbakhsh, M, B Benatallah, A Ignjatovic, Hamid Reza Motahari-Nezhad, 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, T. Abdollah, et al. (2008). "SOMA: A method for developing service-oriented solutions". In: *IBM Systems Journal* 47.3, pp. 377–396. ISSN: 0018-8670.

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. ISBN: 0-7695-1028-0.

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: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10845 LNCS, pp. 136–144. ISBN: 9783319916613.

– (2019). "Auto-Extraction and Integration of Metrics for Web User Interfaces". In: *Journal of Web Engineering* 17.6, pp. 561–590. ISSN: 1540-9589.

Bakaev, Maxim, Sebastian Heil, Nikita Perminov, and Martin Gaedke (2019). "Integration Platform for Metric-Based Analysis of Web User Interfaces". In: *To appear in: Proceedings of 19th International Conference on Web Engineering*. Daejeon, Korea: Springer.

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. ISBN: 978-1-5386-3488-2.

– (2017b). "Web Intelligence Linked Open Data for Website Design Reuse". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10360 LNCS, pp. 370–377. ISBN: 9783319601304.

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. ISBN: 978-1-5386-7054-5.

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 Michael 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. ISBN: 0130676349.

Benguria, Gorka, Brian Elvesæter, and Sylvia Ilieva (2013). *REMICS Handbook , Final Release*. Tech. rep.

Bernardi, Mario Luca, Giuseppe Antonio Di Lucca, and Damiano Distante (2009). "The RE-UWA approach to recover user centered conceptual models fromWeb applications". In: *International Journal on Software Tools for Technology Transfer* 11.6, pp. 485–501. ISSN: 14332779.

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. ISBN: 978-1-4673-2312-3.

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. ISBN: 0-8186-3700-5.

– (May 1994). “Program understanding and the concept assignment problem”. In: *Communications of the ACM* 37.5, pp. 72–82. ISSN: 00010782.

Bing Wu, Deirdre Lawless, Jesus Bisbal, Ray Richardson, 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. ISBN: 0-8186-8126-8.

Binkley, David (2007). “Source Code Analysis: A Road Map”. In: *Future of Software Engineering (FOSE ’07)*. IEEE Computer Society, pp. 104–119. ISBN: 0-7695-2829-5.

Bisbal, Jesús, Deirdre Lawless, Bing Wu, and Jane Grimson (1999). “Legacy information systems: issues and directions”. In: *IEEE Software* 16, pp. 103–111. ISSN: 07407459.

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. ISBN: 0-7695-1799-4.

– (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. ISBN: 0-7695-2027-8.

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. ISBN: 0-7695-2218-1.

Boehm, Barry W. (May 1988). “A spiral model of software development and enhancement”. In: *IEEE Computer* 21.5, pp. 61–72. ISSN: 0018-9162.

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. ISBN: 978-3-663-08951-3.

Borchers, Jens and Knut Hildebrand (1998). “Ein Vorgehensmodell für das Software Reengineering”. In: *Vorgehensmodelle für die betriebliche Anwendungsentwicklung*. Ed. by Ralf Kneuper, Günther Müller-Luschnat, and Andreas Oberweis. Wiesbaden: Vieweg+Teubner Verlag, pp. 152–167. ISBN: 978-3-663-05994-3.

Bow, Sing-Tze (2002). *Pattern Recognition and Image Preprocessing*. 2nd ed. CRC Press Inc. ISBN: 0824706595.

Bozzon, Alessandro, Sara Comai, Piero Fraternali, and Giovanni Toffetti Carughi (2006). “Conceptual modeling and code generation for rich internet applications”. In: *Proceedings of the 6th international conference on Web engineering - ICWE ’06*, p. 353. ISSN: 1041-1135.

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. ISSN: 1354-8565.

– (2013). *Crowdsourcing*. The MIT Press. ISBN: 9780262518475.

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. ISBN: 978-0071361156.

Brodie, M L and M Stonebraker (1995). *Migrating legacy systems: gateways, interfaces & the incremental approach*. 1st ed. Morgan Kaufmann Publishers Inc. ISBN: 978-1558603301.

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. ISSN: 09505849.

Brunelière, Hugo, Jordi Cabot, Javier Luis Canovas Izquierdo, Leire Orue-Echevarria, et al. (Aug. 2015). “Software Modernization Revisited: Challenges and Prospects”. In: *Computer* 48.8, pp. 76–80. ISSN: 0018-9162.

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. ISBN: 0-7695-2013-8.

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 Urbieta, 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. ISSN: 01641212.

CanforaHarman, 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. ISBN: 0-7695-2829-5.

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. ISBN: 978-3-540-76992-7.

Ceccato, Mariano, Massimiliano Di, Paolo Falcarin, Filippo Ricca, et al. (2014). “A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques”. In: *Empirical Software Engineering* 19.4, pp. 1040–1074.

Chechik, Gal, Varun Sharma, Uri Shalit, and Samy Bengio (Mar. 2010). “Large Scale Online Learning of Image Similarity Through Ranking”. In: *The Journal of Machine Learning Research* 11.3/1/2010, pp. 1109–1135. ISSN: 1532-4435.

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. ISSN: 1520-9202.

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. ISSN: 1092-8138.

Chikofsky, Elliot J. and James H. Cross (1990). “Reverse Engineering and Design Recovery: A Taxonomy”. In: *IEEE Software* 7.1, pp. 13–17. ISSN: 07407459.

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. ISBN: 978-1-4673-3076-3.

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.

CMU Software Engineering Institute (2013). *SMART-MP Workshop*.

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. ISSN: 0018-9162.

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. ISBN: 978-1-4244-1255-6.

Creswell, John W (2014). *Research design : Qualitative, quantitative, and mixed methods approaches*. 4. ed., in. Los Angeles: SAGE. ISBN: 9781452226101.

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. ISSN: 07407459.

Deerwester, Scott, Susan T. Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman (Sept. 1990). “Indexing by latent semantic analysis”. In: *Journal of the American Society for Information Science* 41.6, pp. 391–407.

Deng, Jia Deng Jia, Wei Dong Wei Dong, R. Socher, Li-Jia Li Li-Jia Li, et al. (2009). “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2–9. ISBN: 978-1-4244-3992-8.

Deshpande, Yogesh, San Murugesan, Athula Ginige, Steve Hansen, et al. (2002). “Web Engineering”. In: *Journal of Web Engineering* 1.1, pp. 3–17. ISSN: 1540-9589.

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. ISBN: 9781450371452.

Distante, Damiano, Gerardo Canfora, Scott 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. ISBN: 1595933751.

Distante, Damiano, T Parveen, and Scott 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. ISBN: 0-7695-2149-5.

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. ISBN: 0-7695-1804-4.

Distante, Damiano and Scott 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. ISSN: 1550-5502.

Distante, Damiano, Scott 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. ISBN: 0-7695-2536-9.

Domingos, Pedro (Oct. 2012). “A few useful things to know about machine learning”. In: *Communications of the ACM* 55.10, p. 78. ISSN: 00010782.

DSDM Consortium (2014). *DSDM Handbooks*. Tech. rep.

Eaddy, Marc, Thomas Zimmermann, Kaitlin D. Sherwood, Vibhav Garg, et al. (2008). “Do crosscutting concerns cause defects?” In: *IEEE Transactions on Software Engineering* 34.4, pp. 497–515. ISSN: 00985589.

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 (General Da*.

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, R 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. ISSN: 0162-8828.

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. ISBN: 978-3-319-51963-0.

– (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. ISBN: 978-3-319-94958-1.

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. ISBN: 9781612082165.

– (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. ISBN: 978-1-61284-259-2.

– (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. ISSN: 1942-2628.

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. ISSN: 20477473. 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. ISSN: 01676423.

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. ISSN: 18652034.

Gaedke, Martin (2000). “Komponententechnik für Entwicklung und Evolution von Anwendungen im World Wide Web”. dissertation. Universität Karlsruhe, p. 225. ISBN: 3826580591.

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. ISBN: 978-1-4244-1450-5.

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. ISSN: 01676423.

Gold, N. E., M. 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. ISSN: 0038-0644.

Gold, N. E., A. Mohan, C. Knight, and M. Munro (Mar. 2004). “Understanding service-oriented software”. In: *IEEE Software* 21.2, pp. 71–77.

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. ISBN: 0-7695-2861-9.

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. ISBN: 978-1-5386-7757-5.

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. ISBN: 978-1-4244-4897-5.

– (2009b). “Maintaining and evolving GUI-directed test scripts”. In: *2009 IEEE 31st International Conference on Software Engineering*. IEEE, pp. 408–418. ISBN: 978-1-4244-3453-4.

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. ISSN: 10959300.

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, Hua Wang, et al. Vol. 10041 LNCS. Cham: Springer International Publishing. ISBN: 9783319487397.

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 - Science and Technology Publications, pp. 147–158. ISBN: 978-989-758-300-1.

Heil, Sebastian and Martin Gaedke (2016). “AWSM - Agile Web Migration for SMEs”. In: *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*. Enase. Rome, Italy: SCITEPRESS - Science, pp. 189–194. ISBN: 978-989-758-189-2.

– (2017). “Web Migration - A Survey Considering the SME Perspective”. In: *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering*. SCITEPRESS - Science and Technology Publications, pp. 255–262. ISBN: 978-989-758-250-9.

Heil, Sebastian, Valentin Siegert, and Martin Gaedke (2018). “ReWaMP : Rapid Web Migration Prototyping leveraging WebAssembly”. In: *Proceedings of 18th International Conference on Web Engineering (ICWE2018)*. Vol. 10845 LNCS. Caceres, Spain, pp. 84–92. ISBN: 9783319916613.

– (2019). “Crowdsourced Reverse Engineering: Experiences in Applying Crowdsourcing to Concept Assignment”. In: *To appear in: Evaluation of Novel Approaches to Software Engineering, Revised Selected Papers, Communications in Computer and Information Science*. Springer.

Highsmith, Jim and Alistair Cockburn (2001). “Agile software development: the business of innovation”. In: *Computer* 34.9, pp. 120–127. ISSN: 00189162.

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. ISBN: 978-0-13-713012-2.

Howe, Jeff (2006). "The Rise of Crowdsourcing". In: *Wired* 14.6.

IDEO (2015). *The Field Guide to Human-Centered Design*. 1st ed. IDEO.org. ISBN: 978-0-9914063-1-9.

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. ISBN: 9780769551661.

IETF (1997). *RFC 2119 Key words for use in RFCs to Indicate Requirement Levels*. Tech. rep.

Jahanian, Ali, Phillip Isola, and Donglai Wei (2017). "Mining Visual Evolution in 21 Years of Web Design". In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '17*. New York, New York, USA: ACM Press, pp. 2676–2682. ISBN: 9781450346566.

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. ISSN: 2168-7161.

Jamshidi, Pooyan, Claus Pahl, Samuel Chinenyeze, 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. ISBN: 978-3-662-50538-0.

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. ISBN: 1-55860-486-3.

Kaindl, Hermann, Michal Smialek, Patrick Wagner, Davor Svetinovic, et al. (2009). *ReDSeeDS Requirements Specification Language Definition Deliverable D2.4.2*. Tech. rep.

Kan, Stephen H. (1996). *Metrics and Models in Software Quality Engineering*. 3. print. Addison Wesley Longman, Inc. ISBN: 0-201-63339-6.

Kappel, Gerti, Birgitt Pröll, Siegfried Reich, and Werner Retschitzegger, eds. (2006). *Web Engineering – The Discipline of Systematic Development of Web Applications*. Wiley. ISBN: 978-0470015544.

Karampaglis, Zisis, Anakreon Mentis, Fotios Rafailidis, Paschalis Tsolakidis, 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, Alexeis Garcia-Perez, et al. Springer Berlin Heidelberg, pp. 229–243. ISBN: 978-3-642-54338-8.

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. ISBN: 978-1-4673-7016-5.

Kassenärztliche Bundesvereinigung (2018). *IT in der Arztpraxis - Verzeichnis zertifizierter Software*. Tech. rep. Berlin: Kassenärztliche Bundesvereinigung, Dezernat Digitalisierung und IT.

Kazman, R., 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. ISBN: 0-8186-8967-6.

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. ISSN: 00010782.

Khadka, Ravi (2016). "Revisiting Legacy Software System Modernization". dissertation. Utrecht University. ISBN: 9789039365120.

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. ISBN: 9781450327565.

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. ISBN: 978-1-4577-0664-6.

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. ISBN: 9781466624887.

Khusidman, V and W Ulrich (2007). *Architecture-driven modernization: Transforming the enterprise*. Tech. rep.

Khusidman, Vitaly and William Ulrich (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. ISBN: 978-3-642-45397-7.

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. ISBN: 978-1-84628-922-4.

Kong, Jun, Omer Barkol, Ruth Bergman, Ayelet Pnueli, 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, Yosu Gorroñogoitia, 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. ISBN: 9781612082806.

Kumar, Ranjitha, Arvind Satyanarayan, Cesar Torres, Maxine Lim, 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. ISBN: 9781450318990.

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. ISBN: 9781450302289.

Latoza, TD Thomas D and André van der Hoek (2016). "Crowdsourcing in Software Engineering : Models , Opportunities , and Challenges". In: *IEEE Software*, pp. 1–13.

Lee, Changki and Gary Geunbae Lee (Jan. 2006). "Information gain and divergence-based feature selection for machine learning-based text categorization". In: *Information Processing & Management* 42.1, pp. 155–165. ISSN: 03064573.

Lewis, Grace, Edwin Morris, Liam O'Brien, Dennis 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 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. ISBN: 978-1-4799-2072-3.

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. ISSN: 01641212.

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. ISSN: 0883-9514.

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. ISSN: 1041-4347.

Liu, Weibo, Zidong Wang, Xiaohui Liu, Nianyin Zeng, et al. (Apr. 2017). "A survey of deep neural network architectures and their applications". In: *Neurocomputing* 234, pp. 11–26. ISSN: 09252312.

Lucia, Andrea De, Rita Francese, Giuseppe Scanniello, Genoveffa Tortora, Andrea De Lucia, et al. (Nov. 2008). "Developing legacy system migration methods and tools for technology transfer". In: *Software: Practice and Experience* 38.13, pp. 1333–1364. ISSN: 00380644. 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. ISBN: 0-7695-2354-4.

Lucia, Andrea De, Massimiliano Di Penta, Filippo Lanubile, 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. ISBN: 978-1-4244-3755-9.

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. ISSN: 1532060X. arXiv: 1408.1293.

MacKenzie, C Matthew, Ken Laskey, Francis McCabe, Peter F Brown, 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. ISSN: 00380644. 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. ISSN: 15335399.

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. ISSN: 01641212.

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. ISBN: 978-1-4244-2790-1.

Marcotte, Ethan (2010). "Responsive Web Design". In: *A List Apart* 306.

Marcus, Andrian, Andrey Sergeyev, Václav Rajlieh, 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. ISSN: 10951350.

Martin, James (1991). *Rapid Application Development*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc. ISBN: 0-02-376775-8.

Masak, Dieter (2006). *Legacysoftware*. Xpert.press. Berlin/Heidelberg: Springer-Verlag. ISBN: 3-540-25412-9.

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 and The Testing Problem (1998). “DifferentialTestingForSoftware”. In: *DIGITAL TECHNICAL JOURNAL* 10.1, pp. 100–107.

medatixx 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.

Mendelzon, A. and Johannes Sametinger (1995). “Reverse Engineering by Visualizing and Querying”. In: *Software Concepts and Tools* 16, pp. 170–182.

Meng, Xin, Jingwei Shi, Xiaowei Liu, Huifeng Liu, and Lian Wang (July 2011). “Legacy Application Migration to Cloud”. In: *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, pp. 750–751. ISBN: 978-1-4577-0836-7.

Menychtas, Andreas, Kleopatra Konstanteli, Juncal Alonso, Leire Orue-Echevarria, 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. ISSN: 1895-1767.

Menychtas, Andreas, Christina Santzaridou, George Kousiouris, Theodora Varvarigou, 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. ISBN: 978-1-4799-3036-4.

Merlo, E., P.-Y. Gagne, J.-F. Girard, K. Kontogiannis, et al. (1995). “Reengineering user interfaces”. In: *IEEE Software* 12.1, pp. 64–73. ISSN: 07407459.

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 Sadovykh (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 Vayssière. Vol. LNCS 6481. Springer Berlin Heidelberg, pp. 195–196. ISBN: 9783642247545.

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. ISBN: 978-1-4577-0879-4.

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. ISBN: 978-1-84628-922-4.

Müller, Hausi A., Jens H. Jahnke, Dennis B. Smith, Margaret-Anne Storey, et al. (2000). “Reverse engineering”. In: *Proceedings of the conference on The future of Software engineering - ICSE ’00*, pp. 47–60. ISBN: 1581132530.

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. ISBN: 978-1-61284-369-8.

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. ISBN: 978-3-642-31753-8.

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)*. Vol. 7977 LNCS, pp. 510–513. ISBN: 9783642391996.

Nebeling, Michael, Maximilian Speicher, and Mc 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. ISSN: 1939-1374.

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. ISBN: 978-0-7695-3755-9.

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. ISBN: 0897915755.

Nonaka, Ikujiro (2008). *The knowledge-creating company*. Harvard Business Review Press. ISBN: 978-1-4221-7974-1.

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, Hugo Brunelière, et al. (2014). *ARTIST Methodology M30*. Tech. rep., pp. 15–22.

Oulasvirta, Antti, Aliaksei Miniukovich, Gregorio Palmas, Tino Weinkauf, 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. ISBN: 9781450359498.

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.

Paul, Santanu and Atul Prakash (Mar. 1996). "A query algebra for program databases". In: *IEEE Transactions on Software Engineering* 22.3, pp. 202–217. ISSN: 00985589.

Pennington, Nancy (July 1987). “Stimulus structures and mental representations in expert comprehension of computer programs”. In: *Cognitive Psychology* 19.3, pp. 295–341. ISSN: 00100285.

Perez-Castillo, R., I. G-R de Guzman, Mario Piattini, and Christof Ebert (Nov. 2011). “Reengineering Technologies”. In: *IEEE Software* 28.6, pp. 13–17. ISSN: 0740-7459.

Pérez-Castillo, Ricardo, Ignacio García Rodríguez De Guzmán, and Mario Piattini (2011). “Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems”. In: *Computer Standards and Interfaces* 33.6, pp. 519–532. ISSN: 09205489.

Pérez-Castillo, Ricardo, María Fernández-Ropero, Ignacio Garcia-Rodríguez de Guzmán, and Mario Piattini (2011). “MARBLE. A business process archeology tool”. In: *IEEE International Conference on Software Maintenance, ICSM*, pp. 578–581. ISSN: 1063-6773.

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. ISSN: 20477473.

Perez-Castillo, Ricardo, Ignacio García-Rodríguez, 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. ISBN: 9781605581668.

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. ISSN: 09505849.

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. ISBN: 1-59593-170-8.

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. ISBN: 0-7695-1495-2.

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. ISSN: 10928138.

Razavian, Maryam (2009). “Knowledge-driven SOA migration”. In: *CEUR Workshop Proceedings*, pp. 13–18. ISBN: 9789088915895.

– (2013). “Knowledge-driven Migration to Services”. dissertation. Vrije Universiteit Amsterdam. ISBN: 9789088915895.

Razavian, Maryam and Patricia Lago (2010a). “A Frame of Reference for SOA Migration”. In: *Towards a Service-Based Internet*. Ed. by Elisabetta Di Nitto and Ramin Yahyapour. Vol. LNCS 6481. Springer Berlin Heidelberg, pp. 150–162.

Razavian, Maryam and Patricia Lago (2010b). “Towards a Conceptual Framework for Legacy to SOA Migration”. In: *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*. Ed. by Asit Dan, Frédéric Gittler, and Farouk Toumani. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 445–455. ISBN: 978-3-642-16132-2.

– (2010c). “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. ISBN: 9781450315685.

– (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. ISBN: 9783000306273.

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*. 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, ICWE 2014*. Ed. by S. Casteleyn, G. Rossi, and M. Winckler. Vol. 8541. Toulouse, France: Springer, Cham, pp. 494–497. ISBN: 978-3-319-08244-8.

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. ISBN: 978-3-319-04244-2.

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. ISSN: 18770509.

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 (ICWE)*, 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. ISBN: 978-3-642-13910-9.

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. ISSN: 03787206.

Roy Choudhary, Shauvik, Mukul R. Prasad, and Alessandro Orso (2014a). "Cross-platform feature matching for web applications". In: pp. 82–92.

– (2014b). "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. ISBN: 9781450326452.

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. ISBN: 978-1-4244-8630-4.

Rubin, Julia and Marsha Chechik (2013). "A Survey of Feature Location Techniques". In: *Domain Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 29–58. ISBN: 978-3-642-36653-6.

Rubner, Yossi, Carlo Tomasi, and Leonidas J Guibas (1998). "The Earch Mover's Distance as a Metric for Image Retrieval". In: *International Journal of Computer Vision* 40.2, pp. 99–121.

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. ISSN: 00380644.

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.

Salton, G., A. Wong, and C. S. Yang (1975). "A vector space model for automatic indexing". In: *Communications of the ACM* 18.11, pp. 613–620. ISSN: 00010782.

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. ISBN: 978-1-4799-3824-7.

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, Stefan Wild, et al. (2014). "Toward Collaborative Software Engineering Leveraging the Crowd". In: *Economics-Driven Software Architecture*. Elsevier, pp. 159–182. ISBN: 978-0-12-410464-8.

Sauro, Jeff and James R. Lewis (2016). *Quantifying the User Experience: Practical Statistics for User Research*. 2nd ed. Elsevier LTD, Oxford. ISBN: 978-0-12-802308-2.

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. ISBN: 978-1-4799-7329-3. arXiv: 1605.08642.

Scholkopf, B., C.J.C. Burges, F. Girosi, P. Niyogi, et al. (1997). “Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers”. In: *IEEE Transactions on Signal Processing* 45.11, pp. 2758–2765. ISSN: 1053587X.

Schueffel, Patrick (2016). “Taming the Beast: A Scientific Definition of Fintech”. In: *Journal of Innovation Management* 4.4, pp. 32–54. ISSN: 1556-5068.

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. ISBN: 0897917782.

Seacord, Robert C, Grace Lewis, and Daniel Plakosh (2003). *Modernizing legacy systems Software technologies, engineering processes, and business practices*. Boston: Addison-Wesley. ISBN: 9780321118844.

Semenenko, Nataliia, Marlon Dumas, and Tonis Saar (Sept. 2013). “Browserbite: Accurate Cross-Browser Testing via Machine Learning over Image Features”. In: *2013 IEEE International Conference on Software Maintenance*. IEEE, pp. 528–531. ISBN: 978-0-7695-4981-1.

Sneed, Harry M (1995). “Planning the reengineering of legacy systems”. In: *IEEE Software* 12.1, pp. 24–34. ISSN: 07407459.

Sneed, Harry M., Ellen Wolf, and Heidi Heilmann (2010a). *Softwaremigration in der Praxis: Übertragung alter Softwaresysteme in eine moderne Umgebung*. dpunkt Verlag. ISBN: 978-3898645645.

– (2010b). “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. ISBN: 978-3898645645.

Sosa Sanchez, 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. ISSN: 1548-0992.

Sosa-Sanchez, Encarna, Pedro J. Clemente, Miguel Sanchez-Cabrera, José María Conejero, 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. ISBN: 978-1-4799-5069-0.

Sosa, Encarna, Pedro J Clemente, José María Conejero, and Roberto Rodriguez-Echeverria (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. ISBN: 978-1-4799-1610-8.

Stapleton, Jennifer (1997). *DSDM, dynamic systems development method: the method in practice*. 1st ed. Addison-Wesley Professional. ISBN: 978-0201178890.

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. ISBN: 9781450327565.

Strauch, Steve, Vasilios Andrikopoulos, Thomas Bachmann, Dimka Karastoyanova, 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. ISBN: 978-0-7695-5095-4.

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. ISSN: 1573-7535.

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. ISBN: 0-7695-0303-9.

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. ISBN: 0-7695-1819-2.

Stroulia, Eleni and Tarja Systä (Apr. 2002). "Dynamic analysis for reverse engineering and program understanding". In: *ACM SIGAPP Applied Computing Review* 10.1, pp. 8–17. ISSN: 15596915.

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. ISBN: 978-1-4799-5063-8.

Tan, Wei, Yushun Fan, Ahmed Ghoneim, M. Anwar Hossain, and Schahram Dustdar (July 2016). "From the Service-Oriented Architecture to the Web API Economy". In: *IEEE Internet Computing* 20.4, pp. 64–68. ISSN: 1089-7801.

Tilley, S.R., Santanu Paul, and D.B. Smith (1996). "Towards a framework for program understanding". In: *WPC '96. 4th Workshop on Program Comprehension*, pp. 19–28. ISBN: 0-8186-7283-8.

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*. l. IEEE, pp. 35–42. ISBN: 978-1-4244-2776-5.

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. ISSN: 1042-1629.

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. ISSN: 10715819.

Tunkelang, Daniel (Jan. 2009). "Faceted Search". In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 1.1, pp. 1–80. ISSN: 1947-945X.

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. ISBN: 0-7695-2432-X.

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). “RDOTE – Publishing Relational Databases into the Semantic Web”. In: *Journal of Systems and Software* 86.1, pp. 89–99. ISSN: 01641212.

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. ISSN: 09574174.

Wagner, Christian (2014). *Model-Driven Software Migration: A Methodology*. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-05270-6.

Wallmüller, E (2001). *Software-Qualitätsmanagement in der Praxis: Software-Qualität durch Führung und Verbesserung von Software-Prozessen*. 2., völlig. München: Hanser. ISBN: 978-3-446-40405-2.

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, I. 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. ISBN: 0-7695-1727-7.

Warren, Ian (2012). *The Renaissance of Legacy Systems: Method Support for Software-System Evolution*. Springer Science & Business Media. ISBN: 9781447108177.

Wasserman, Anthony I. and David T. Shewmake (Dec. 1982). “Rapid prototyping of interactive information systems”. In: *ACM SIGSOFT Software Engineering Notes* 7.5, pp. 171–180. ISSN: 01635948.

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. ISSN: 15591131.

Weidema, Edgar R.Q. 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 CrowdSourcing in Software Engineering - CSI-SE '16*. New York, New York, USA: ACM Press, pp. 41–44. ISBN: 9781450341585.

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. ISBN: 9781450321617.

Winter, Andreas, C. Zillmann, A. Herget, Werner Teppe, 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.

Wood, David and Kyo Kang (1992). *A Classification and Bibliography of Software Prototyping*. Tech. rep. CMU/SEI-92-TR-013. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

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. ISBN: 978-1-4244-7475-2.

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. ISSN: 01641212.

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. ISBN: 978-1-4503-0942-4.

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. ISBN: 978-1-4799-4171-1.

Zillmann, C., Andreas Winter, A. Herget, Werner Teppe, 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. ISBN: 978-1-61284-259-2.

Standards and Specifications

ISO (2010). *INTERNATIONAL STANDARD ISO 9241-210 Ergonomics of human–system interaction — Part 210: Human-centred design for interactive systems*.

ISO/IEC (2009). *IFPUG functional size measurement method 2009*. Tech. rep. ISO/IEC 20926:2009.

ISO/IEEE (2006). *INTERNATIONAL STANDARD ISO / IEC 14764 Software Engineering — Software Life Cycle Processes — Maintenance*. Tech. rep.

– (2017a). *ISO/IEC/IEEE International Standard - Systems and software engineering - Measurement process*. Tech. rep.

– (Aug. 2017b). *ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary*. Tech. rep.

OASIS (2007). *Web services business process execution language version 2.0*.

Object Management Group (2011). *Architecture-driven Modernization : Abstract Syntax Tree Metamodel (ASTM)*.

– (2012). *Structured Metrics Metamodel (SMM)*.

Object Management Group (2014). *Model Driven Architecture (MDA) Guide rev.2.0*. Tech. rep. June, pp. 1–15.

– (2015). *Implementation of applications specified with IFML*. Tech. rep. February. Object Management Group.

– (2016a). *Architecture-Driven Modernization: Knowledge Discovery Meta-Model (KDM) Version 1.4*. Tech. rep. September.

– (2016b). *Query / View / Transformation Specification Version 1.3*.

Software Engineering Standards Committee of the and IEEE Computer Society (1998). *IEEE Standard for Software Maintenance*. Tech. rep.

W3C (2015). *W3C DOM4*.

– (2017). *CSS Grid Layout Module Level 1*.

– (2018a). *Web Components*.

– (2018b). *WebAssembly Core Specification*.

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*. (Visited on June 10, 2018).

Cunningham, Ward (1992). *The WyCash Portfolio Management System*. URL: `http://c2.com/doc/oopsla92.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-rates-insurance-legacy`.

– (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?assetid=30930`.

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`.

– (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).

Simpson, Alastair (2016). *An agile design prototype is worth a thousand user stories*. URL: `https://www.atlassian.com/blog/agile/agile-design-prototype` (visited on Feb. 17, 2019).

Statista (2018a). *Cloud Software as a Service - Statistics & Facts | Statista*. URL: https://www.statista.com/topics/3071/cloud-software-as-a-service-saas/.

– (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/.

Wessa, P (2017). *Multiple Regression (v1.0.48)*. URL: https://www.wessa.net/rwasp%7B%5C_%7Dmultipleregression.wasp/ (visited on Mar. 31, 2019).

List of Acronyms

ADM	Architecture-Driven Modernization.
ADMTF	Architecture-Driven Modernization Task Force.
API	Application Programming Interface.
AST	Abstract Syntax Tree.
ASTM	Abstract Syntax Tree Metamodel, OMG ADM standard.
ATL	ATLAS Transformation Language.
AWSM	Agile Web Migration for SMEs.
AWSMAP	AWSM Annotation Platform.
CIM	Computation-Independent Model.
CSS	Cascading Style Sheets.
EMP	Eclipse Modeling Project.
FHOG	Felzenszwalb histogram of oriented gradients.
GASTM	Generic Abstract Syntax Tree Metamodel, part of ASTM specification.
GUI	Graphical User Interface.
HCD	Human-Centered Design.
HCI	Human-computer interaction.
HMW	How Might We.
HTML	HyperText Markup Language.
HTTP	HyperText Transfer Protocol.
IaaS	Infrastructure as a Service.
IEC	International Electrotechnical Commission.
IEEE	Institute of Electrical and Electronics Engineers.
IFML	The Interaction Flow Modeling Language, OMG standard.
ISO	International Organization for Standardization.
ISV	Independent Software Vendor.
KDM	Knowledge Discovery Meta-Model, OMG standard.
MDA	Model Driven Architecture, OMG standard.

MDE	Model-Driven Engineering.
MDWE	Model-Driven Web Engineering.
MOF	Meta Object Facility, OMG standard.
MVC	Model-View-Controller.
OCR	Optical Character Recognition.
OMG	Object Management Group.
PaaS	Platform as a Service.
PIM	Platform-Independent Model.
PMS	Patient Management System.
PSM	Plattform-Specific Model.
QVT	Query/View/Transformation, OMG ADM standard.
ReMiP	Reference Migration Process.
ReWaMP	Rapid Web Migration Prototyping leveraging WebAssembly.
RIA	Rich Internet Application.
RWMPA	Rapid Web Migration Prototyping Assistance.
SaaS	Software as a Service.
SASTM	Specific Abstract Syntax Tree Metamodel, part of ASTM specification.
SME	Small and Medium-sized Enterprise.
SMM	Structured Metrics Metamodel, OMG ADM standard.
SOA	Service-oriented Architecture.
SOA-MF	SOA Migration Framework.
TUI	Text-based User Interface.
UI	User Interface.
UIX	User Interaction.
VUE-D	Visual UI Element Detector.
W3C	World Wide Web Consortium.
Web	World Wide Web.
WebML	The Web Modeling Language, see also IFML.

List of Figures

List of Tables

Declaration

I hereby declare that this dissertation entitled “Web Migration Revisited: Lowering the initial hurdle to commence a Web Migration” is my own work. All thoughts taken from other sources have been properly marked as such. This dissertation has not been previously published nor it is under review elsewhere.

Chemnitz, June 5, 2019

Sebastian Heil M.Sc.