PROFRAME: A Prototyping Framework for Mobile Enterprise Applications

Matthias Jurisch, Bodo Igler, Stephan Böhm

Faculty of Design – Computer Science – Media
RheinMain University of Applied Sciences
Wiesbaden, Germany
Email: {matthias.jurisch,bodo.igler,stephan.boehm}@hs-rm.de

***Abstract*—The rise of mobile device dissemination over the last few years and their short product life cycles require developers of enterprise applications to adapt to this situation. While the development of consumer apps is supported by many tools and processes, these can not be easily adapted to enterprise needs. Therefore, new process models for Mobile Enterprise Application development are required. In this paper, we present an approach based on Software Product Lines and Design Science Research that gathers information in the form of patterns from existing projects to ease the development of new applications. This information contains data from a user, organizational and technical perspective. The approach is currently a work in progress but will be further developed in the future.**

Keywords–Mobile Enterprise Applications; Design Science Research; Software Product Lines; User-centered Design; Prototyping; Pattern Inventories.

I. INTRODUCTION

Over the last few years, a rise in the dissemination of mobile devices could be observed. The development of applications for these devices is characterized by short product life cycles and high expectations regarding the usability of applications. On the other hand, development culture in large enterprises is often founded on precise specification and heavy-weight processes. Large enterprises usually have several requirements concerning compliance, security, linking to legacy back-end-applications and adherence to corporate design guidelines whereas in the mobile consumer market, these problems are not necessarily considered. In the mobile consumer market, new processes and concepts have been developed, which do not suit the needs for user-centric application development in large enterprises. Therefore, tools and approaches for *Mobile Enterprise Application* (MEA) development are required.

In this paper, we present a Prototyping Framework for Mobile Enterprise Applications (PROFRAME) containing a tool and process model using a prototyping approach that suits mobile enterprise applications. This process uses *structured and formal modeling* of application artifacts to support prototyping and takes organizational aspects of software projects into account. The process itself is based on the established methodology *design science research*.

The paper is organized as follows: Scientific work related to our project is presented in Section II. Our approach is described in Section III. Preliminary results are presented in Section IV. A conclusion and outlook are given in Section V.

II. RELATED WORK

According to [1] and [2], many enterprises still lack experience concerning the development of MEAs. While there are no established process models for MEA development, first research approaches for process models can be found in related literature. Dugerdil [3] presents an approach for transforming enterprise applications to mobile applications. An instrumentation framework that tries to ease the maintenance of MEAs is proposed in [4]. The management perspective of this problem is also represented in literature. Badami [5] examines this aspect from an organizational viewpoint and proposed the concentration of MEA development into "Mobile Centers of Excellence" that concentrate the competences of mobile experts inside enterprises.

The previously mentioned processes can be supported by tools. Existing prototyping tools (Kony, Verivo, Akula, SAP Mobile Platform) allow rapid prototyping. These tools can not always be used in MEA-contexts, since they are focused on predefined use cases or the integration of existing enterprise products.

No processes or tools that specifically support the development of MEAs can be found in literature or in practice. Therefore, new approaches are required that take the characteristics of MEAs into account. Central questions that need to be answered are how the approach can take the specifics of MEAs into account and how the approach can decrease the time and effort for development. These questions could be answered through reuse of artifacts from existing MEA projects. Frameworks or approaches focusing on the inventory and reuse of technical components, user interface patterns or organizational aspects can not be found in literature or practice. Especially organizational aspects are not represented in mainly software-focused work.

III. APPROACH

Our hypothesis is that in a pool of MEA projects, there are some artifacts or patterns from different domains (organizational, technical and UI design aspects) that can be reused in future projects. The idea of our approach is to analyze existing completed MEA-projects and extract these patterns and artifacts from them. Patterns are stored in a knowledge base that allows efficient searching and semantic modeling. This knowledge base can then be used for new MEA-projects that can reuse components of earlier MEAs.

The knowledge base will contain knowledge from (at least) three perspectives. These perspectives are a user-centric view on the application including its UI design, technical aspects concerning the platform and structure of the application and an organizational viewpoint on the project. This can be used to aid the reuse of artifacts from other projects. Reusing components from earlier work efficiently can help concentrating on aspects like user-centricity and reduce resources required to carry out

 ISBN: 978-1-61208-502-9

the project. This would make it easier for MEA-development to catch up with consumer app development.

To efficiently execute this process, a tool is required. The tool will enable the creation of new patterns, entering existing projects including the usage of patterns into the knowledge base and creating new projects, as well as a UI prototype from a wireframe-like UI editor. The information for new projects can then be used to find relevant existing projects including information about potential reusable artifacts and knowledge from existing projects. An example for the UI for entering information regarding new patterns is shown in Fig. 1. For each pattern, a name and a description (1), specific properties (2) and an image for graphical representation (3) can be added. If the pattern is a *UI pattern* (4), the image can be used inside the tool's UI editor during the creation of a new project.

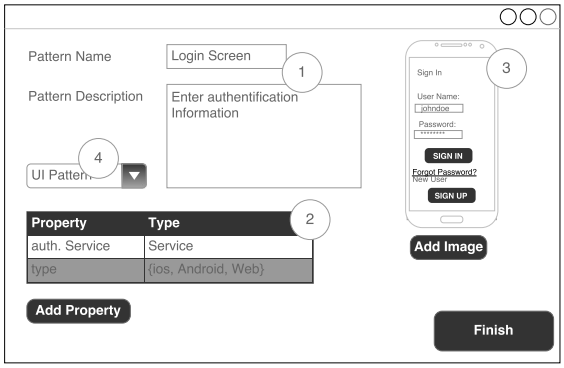

Figure 1. Prototype for Tool UI

Our approach combines aspects from the fields of software engineering, semantic technologies, IT-management and User-Centered Design. We divided the problem into four sub-problems, where we apply knowledge from these fields. The subproblems are:

1) identification of reusable software components in existing software,
2) adequate description, filing and semantic linking of information regarding these software components,
3) representing non-technical aspects in the knowledge base,
4) development of a tool that aides the application of the process with a high usability and
5) a process for a continuous construction of a suitable inventory.

To identify reusable components in software systems (problem 1), a detailed analysis of the software features and artifacts is required. Similar problems are addressed by *Software Product Lines*, which are discussed in Section III-A. Existing collections of reusable components exist as collections of *User Interface Design Patterns*, which is described in Section III-B. Creating an adequate modeling framework (problems 2,3 and 4) requires modern approaches from *Model driven Software Development* (Section III-C). To support interlinking of information from several domains and inference of new knowledge, *Semantic Technologies* (Section III-D) can provide useful technical concepts. For the development process (problem 5), domain feedback and inclusion of existing knowledge is important. These aspects are addressed in the area of *Design Science Research*, which is described in Section III-E.

A. Software Product Lines

Software Product Lines (SPL) [6] is an approach from the field of software engineering. Its main target is to ease the production of software variants while systematically reusing software components. The foundation for SPL is a catalog of artifacts. In production, these artifacts can be combined and adapted to produce new software variants.

SPL can contribute greatly to our approach. Product-feature trees that contain the features of a software product line, can be reused to describe features of MEAs in the knowledge base in our problem context. But SPL cannot be directly applied to our problem setting. SPL typically starts from scratch to develop a software product line, while our approach extracts patterns and artifacts from existing MEAs. Also, switching to a full-fledged SPL-approach would require a great deal of time and high effort, which is problematic for enterprises in the MEA-field given the dynamic market. Aspects that could be reused from SPL are Product-feature trees as well as reusable software components.

B. User Interface Design Pattern Libraries

Usage of patterns is state of the art in software engineering, especially for technical aspects of software design (e.g. [7]). Nowadays, patterns are used in all areas of software engineering. For our problem domain, user interface design pattern libraries [8] can provide an important contribution.

Besides pattern libraries for conventional desktop software, there are also pattern libraries for mobile applications (e.g. [9]). To use these concepts in the field of MEAs, they need to be transferred to the enterprise context. Relating these patterns to other system artifacts is important for the application of these concepts, too. Then, these patterns can be reused for prototyping, which could include the usage of technical system artifacts.

C. Model Driven Software Development

In Software Engineering, models are used for specification, analysis and design of systems. The models usually contain information about the structure or behavior of the described system and provide some kind of meaning (semantic) for this information. *Model Driven Software Development* (MDSD) deals with the automated generation of software components from these models [10]. This can help integrate domain experts into the development process who are not experienced with software development. The generated components can then be integrated into manually created software.

In the context of our problem domain, adequate models for describing system artifacts and software components including their dependencies and organizational aspects of the corresponding software projects are of importance. Whether existing modeling languages can be used to fulfill these requirements or a new language needs to be designed has to be examined.

D. Semantic Technologies

For the modeling, semantic modeling technologies based on ontologies can be used. Ontologies are defined as an "explicit specification of a conceptualization" [11]. Ontologies provide a mechanism to represent concepts and the relations between them. The semantics of the representation allow the inference of new knowledge from the ontology. Tim Berners-Lee and others proposed the semantic web, a technical approach to applying ontology-based technologies to the web

 ISBN: 978-1-61208-502-9

[12]. The semantic web is especially useful for interlinking information from different domains.

An excerpt from an example ontology is shown in Fig. 2. The excerpt shows a hierarchy of concepts. The concept `'App Component'` is a sub-concept of `owl:Thing`, which represents all concepts. `'App Component'` itself has several sub-concepts. In addition to the simple taxonomic structure shown in the example, ontologies support the definition of relations between instances of concepts and restrictions to these relations.

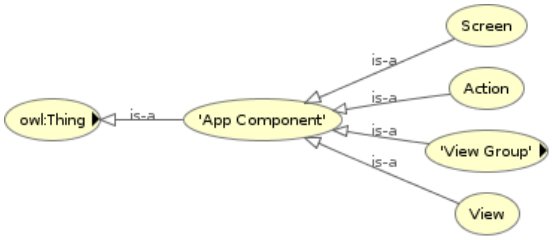

Figure 2. Example Ontology

In our approach, semantic web technologies will be used to provide a knowledge base that contains project information from completed MEA projects. The natural support for interlinking data from different domains will be useful when integrating the data from a user centric view, a technical artifact standpoint and an organizational perspective. These schemes can then be connected via so-called mapping ontologies. The separate development of ontologies for different domains might help bootstrapping the approach.

E. Design Science Research

Design Science Research (DSR) was proposed by Hevner et. al. [13]. It unites concepts from behavioral and design science in a cyclic model. The objective of this process is to improve the application domain by creating new artifacts (products and processes). The starting point of the process originates from requirements that stem from the application domain. Every artifact created in the process is validated against these requirements in the *relevance cycle*. Field tests are used to confirm that a created artifact meets the domain requirements. The artifacts are created in the *design cycle*, which also evaluates the artifacts based on the requirements. This cycle is the core of the process. The *rigor cycle* provides access to and updates a knowledge base that is used in the design cycle.

The DSR-model can be applied in two ways: The process model and other artifacts can be seen as artifacts of the DSR-process. Hence, DSR can be seen as the process model for our future research. The other perception would include the DSR-process into the created process model, where MEA-artifacts are seen as artifacts in the sense of DSR. The knowledge base can be structured as an ontology and the knowledge base used in the rigor cycle.

IV. Preliminary Results

Preliminary results for the usage of the tool and its integration into the app development process and the tools architecture were found. The concept for usage is described in Section IV-A and the tool's architecture is presented in Section IV-B.

A. Tool Usage Concept

There are two different modes when using the tool. The first mode is entering existing, finished projects into the tool's knowledge base. In this mode, the user can enter several details regarding the project. These details are separated into three views: the user-centric view of the application, including requirements and UI design, the technical viewpoint and an organizational perspective on the project.

The UI perspective contains details regarding the flow of screens and is represented in a wireframe-like editor. The user can annotate theses screens with patterns (see Fig. 1) to add more details to the knowledge base. The technical perspective can be used to specify, among others, a development platform, a deployment platform, used services and related artifacts. The IT-management perspective can be used to enter information regarding the organizational aspects of the project like the process model used, costs, development time and other organizational aspects. The information can be entered through form-based editors using attribute-value pairs.

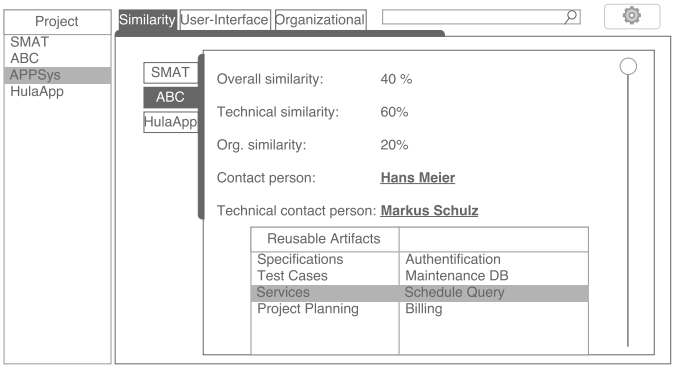

Figure 3. Screen Prototype for similarity View

The second mode allows the user to create a new project. The user can fill out the information that is known about the project with the same forms that are used to enter existing projects. The user can then open a similarity view that shows similarity scores for existing projects. A prototype for that view is depicted in Fig. 3. The user can see different similarity scores for the different perspectives as well as contact persons for different aspects of the project. In the bottom of the view is a table that contains reusable artifacts that can be filtered by categories. Moreover, the user can generate a prototype from the wireframe-like UI editor, which is not shown in this paper.

B. Tool Architecture

The tool's architecture is depicted in Fig. 4 as a Fundamental Modeling Concepts diagram. The two modes of operation presented in Section IV-A are both executed by the *PROFRAME User*. The *Project Participant* uses an ontology editor to edit the structure of the knowledge base and adds the initial project data concerning the technical, business- and user-related aspects. The ontologies are connected through a mapping ontology that relates the concepts from the different ontologies to each other, using the natural interlinking support of ontologies. Ontologies can also ease the variability of the knowledge base. From the ontology, an MDSD-Framework can generate the *PROFRAME Project Tool*, which can then be used by the *PROFRAME User*.

 ISBN: 978-1-61208-502-9

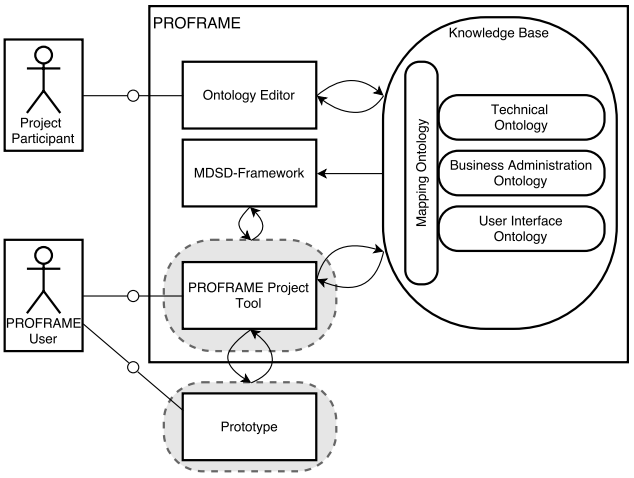

Figure 4. Tool Architecture

The *PROFRAME User* can enter the project data for a new project into the *PROFRAME Project tool*, which is also represented as the content of the ontology. The user enters data regarding desired process model, platform, etc. A wireframe that uses existing patterns can also be used as input. From the wireframe, the tool can generate a prototype that can be used to demonstrate the capabilities of the planned application to customers. To further plan the project, the user can use the screen shown in Fig. 3, which can be used to, e.g., find contact persons with relevant information and potential reusable artifacts. The *PROFRAME project tool* is also responsible for computing the similarities between planned and existing projects to be shown in a similarity view reusing approaches from the field of recommendation systems for software engineering. An implementation could use case based reasoning, where the similarities between cases can be computed using word similarity in the descriptions of the projects, similar to the work presented in [14]. Using structural similarity measures could also be of use. Another promising strategy for finding similar projects is the use of rule-based recommender systems [15], that explicitly state the relation between properties of the projects.

V. Conclusion and Outlook

In this paper, we have presented an approach to tackle the difficulties of mobile enterprise application development. Our approach combines methods from the domains of software product lines, user interface design pattern libraries, model driven software development, semantic technologies and design science research to improve the development process. The approach has the advantages that it allows early prototyping and high reuse of existing artifacts. Therefore, it can lead to reduced costs for MEA projects. But most importantly, the approach organizes the knowledge from different MEA projects. The approach could prevent the loss of knowledge, when employees leave the company or are transfered to other positions. This could also reduce costs for companies employing this approach. Also, the approach could ease communication between teams inside the company.

In the future, we plan to implement our approach with an industry partner that uses mobile centers of excellence to evaluate and improve it. User motivation including the perceived usefulness and ease of use of the approach will be central to the evaluation. The outcome of the process will be a pattern inventory, a modeling language, a prototyping tool and a cost-benefit analysis of the overall approach. Also, our approach can lead to new scientific results on how to represent data regarding software development projects that take a technical, business-related and user-centric perspective into account. A process based on design science research for MEAs could be another interesting outcome. Another research question is whether the prototyping aspect of the process can improve the user-centricity of the resulting MEAs which will also be evaluated in the future.

References

[1] A. Giessmann, K. Stanoevska-Slabeva, and B. de Visser, "Mobile enterprise applications–current state and future directions," in System Science (HICSS), 2012 45th Hawaii International Conference on, Jan 2012, pp. 1363–1372, Jan 2012.

[2] Gartner, "Gartner says demand for enterprise mobile apps will outstrip available development capacity five to one," https://www.gartner.com/newsroom/id/3076817, website [retrieved: June, 2016], 2015.

[3] P. Dugerdil, "Architecting mobile enterprise app: A modeling approach to adapt enterprise applications to the mobile," in Proceedings of the 2013 ACM Workshop on Mobile Development Lifecycle, ser. MobileDeLi '13. New York, NY, USA: ACM, 2013, pp. 9–14, 2013.

[4] M. Pistoia and O. Tripp, "Integrating security, analytics and application management into the mobile development lifecycle," in Proceedings of the 2Nd International Workshop on Mobile Development Lifecycle, ser. MobileDeLi '14. New York, NY, USA: ACM, 2014, pp. 17–18, 2014.

[5] S. A. Badami and J. Sathyan, "micE Model for Defining Enterprise Mobile Strategy," Int. J. on Recent Trends in Engineering and Technology,, vol. 10, no. 1, p. 9, Jan 2014.

[6] K. Pohl, G. Böckle, and F. J. v. d. Linden, Software Product Line Engineering: Foundations, Principles and Techniques. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[7] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture - Volume 1: A System of Patterns. Wiley Publishing, 1996.

[8] M. Welie, G. C. Veer, and A. Eliëns, Tools for Working with Guidelines: Annual Meeting of the Special Interest Group. London: Springer London, 2001, ch. Patterns as Tools for User Interface Design, pp. 313–324, 2001.

[9] "Mobile patterns," http://www.mobile-patterns.com/, website [retrieved: June, 2016], 2016.

[10] T. Stahl, M. Voelter, and K. Czarnecki, Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, 2006.

[11] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Int. J. Hum.-Comput. Stud., vol. 43, no. 5-6, pp. 907–928, Dec. 1995.

[12] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," Scientific American, vol. 284, no. 5, pp. 34–43, May 2001.

[13] A. R. Hevner, "A three cycle view of design science research," Scandinavian Journal of Information Systems, pp. 87–92, 2007.

[14] P. Gomes, F. C. Pereira, P. Paiva, N. Seco, P. Carreiro, J. Ferreira, and C. Bento, "Selection and Reuse of Software Design Patterns Using CBR and WordNet," SEKE'03, 2003.

[15] F. Palma, H. Farzin, Y. G. Guéhéneuc, and N. Moha, "Recommendation system for design patterns in software development: An DPR overview," RSSE 2012, pp. 1–5, 2012.

 ISBN: 978-1-61208-502-9