Flexible Development of Variable Software Features for Mobile Business Applications

Viktor Kolokolov
WSN Lab
TU Darmstadt, Germany
viktor.kolokolov@wsn.tu-darmstadt.de

Paul Baumann
WSN Lab
TU Darmstadt, Germany
paul.baumann@wsn.tu-darmstadt.de

Silvia Santini
WSN Lab
TU Darmstadt, Germany
santinis@wsn.tu-darmstadt.de

Stefan T. Ruehl
TU Clausthal, Germany
stefan.t.ruehl@tu-clausthal.de

Stephan A. W. Verclas
T-Systems International GmbH
Darmstadt, Germany
stephan.verclas@t-systems.com

ABSTRACT

With recent advances in development and deployment of mobile business applications (MBAs) based on the hybrid Web approach (hybrid MBAs) enterprises around the world well recognize new potentials to mobilize their business processes (BPs). Variability has a natural appearance in complex environments of different enterprises, where even similar BPs can have varying facets on the cross-enterprise scale. Yet, despite this fact current development tools for hybrid MBAs are lacking systematic variability management. Further, the literature on this particular technological landscape is scarce. We highlight in this paper emerging importance of this research field and describe its context and a research methodology. We propose an SPL-based approach to tackle considerable variabilities of hybrid MBAs.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements / Specifications – Tools; D.2.2 [**Software Engineering**]: Design Tools and Techniques – Computer-aided software engineering (CASE); D.2.13 [**Software Engineering**]: Reusable Software – Domain engineering, Reuse models

General Terms

Design, Documentation

Keywords

Mobile Business Applications, Hybrid Web, Software Product Lines, Variability Modeling

SPLC 2013 workshops, August 26 – 30 2013, Tokyo, Japan
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2325-3/13/08...$15.00.
http://dx.doi.org/10.1145/2499777.2500712

1. INTRODUCTION AND MOTIVATION

The research proposed in this paper is very beneficial for the SPL engineering (SPLE) community and industry. In particular, we make in this work the following contributions:

- Problem analysis of variability-aware development of hybrid MBAs:
  - Typical variation points and their occurrence in the technological environment they are settled in;
  - Characteristic relations among variation points.
- Systematic software reuse approach for hybrid MBAs:
  - Conceptual variability design;
  - Configuration and derivation methodology.

Mobile software technologies are making significant advances facilitating daily business tasks through novel applications. These technologies are changing the way people access and process information. A rich variety of mobile business applications (MBAs) enables new possibilities for enterprises to make their business processes (BPs) ubiquitously supported. Herein, an MBA is an application that supports a BP and reflects or inherits its characteristics, such as a structure of the BP, i.e., arranged steps (activities of a workflow of the BP), which constitute it, constraints, different ways how the BP can be executed, involved backend systems. Unlike MBAs, mobile non-business applications provide rather a limited or isolated functionality, such as a calculator or a game.

Currently, there are multiple development paradigms for MBAs, e.g., native and Web applications [10]. Due to the peculiarities of the underlying technologies, each development approach has inherent advantages and drawbacks. The key advantage of Web applications over natively build mobile applications is their cross-platform compatibility. Furthermore, they are in general cheaper, easier, and faster to realize [10]. Yet, Web applications are often hampered by the limited capabilities of mobile browsers and cannot access low-level hardware and software resources of the hosting devices[1]. Instead, native applications leverage the capabilities

[1]Presently changes emerge along with HTML5.

of a mobile device providing a more compelling user experience. Further, they can run off-line, but are also more expensive to maintain for every target platform.

A capability to coalesce Web applications and a native code, e.g., written in Java for Android or in Objective-C for iOS can expressively reduce development time and costs keeping at the same time the main advantages of both approaches. This motivates the increasing use of the hybrid Web approach into development of MBAs. Hybrid MBAs run on mobile devices wrapped around a native container and exploit its browser engine to process HTML5, Cascading Style Sheets (CSS) and JavaScript. Herein, a Web-to-native abstraction layer provides access to features of the native platform. This layer can be incorporated in Mobile Enterprise Application Platforms (MEAPs), e.g., Sybase Unwired Platform (SUP) [3], which also address appropriate levels of security and management in deploying mobile applications. These technologies, which are at present in use, and environmental aspects define hybrid MBAs and govern their distinct technological scope as compared to other development paradigms.

Further, on the cross-enterprise scale there can be potential differences in the workflow of a hybrid MBA supporting a BP, such as multiple authentication mechanisms, variations of data input methods, optional or alternative data pre- and processing activities. Typically, hybrid MBAs inherit also BPs' constraints, which can be enforced within single activities of the workflow during its execution. For instance, a constraint can be defined at an activity to check allowed formats and values of manually inputted data or to verify completeness of required data and its compliance with defined BP norms. Furthermore, there is a need for various customizations, such as branding or internationalization.

Unfortunately, with currently available development tools of MEAPs for hybrid MBAs it is not possible to systematically document neither variable workflow features arising on the cross-enterprise scale nor further considerable variabilities, which are not related to the workflow level. This complicates the adoption of hybrid MBAs across varying BP environments. Therefore, systematic variability management for hybrid MBAs that is lacking in modern MEAPs is needed.

This paper is structured as follows: Section 2 describes the background and the context, Section 3 describes the research methodology, Section 4 considers typical variation points of hybrid MBAs. Section 5 presents a conceptual variability design. In Section 6 realization aspects of an SPL for hybrid MBAs are discussed. Future work and our conclusions are highlighted in Sections 7 and 8.

2. BACKGROUND AND RELATED WORK

The idea of software reuse is nearly as old as software engineering itself. Some early attempts to reuse software artifacts had usually a small-scale, ad-hoc nature without systematic variability analysis. The origin of software product lines (SPLs) [26], as a systematic reuse approach encompassing all relevant assets throughout the entire software development life-cycle, is rooted in the early 1990s. One of the first contributions to it was the introduction of the Feature-Oriented Domain Analysis (FODA) [20]. Thereafter it underwent various extensions [7] and has been studied in several research fields: embedded systems [12], Service Oriented Architecture (SOA) [5], and Web applications [34, 6]. To the best of our knowledge, the use of SPLs to support development of hybrid MBAs has not been discussed in the literature yet.

Development of hybrid MBAs represents a convergence point for advantages of both native and Web-based technologies. A hybrid application is primarily built using widespread Web technologies, such as HTML5, CSS and JavaScript. It is then wrapped inside a native application (container) with access to features of the native platform that it runs upon. PhoneGap[2] is an example of a framework that facilitates creation of hybrid mobile applications.

With optimized JavaScript, CSS designed to realize beautiful layouts, and compliant HTML5 code that is platform agnostic, thus working on all major platforms, it is possible to create sophisticated mobile applications without sacrificing native capabilities. Further, it is known that native applications are installed on the mobile device, whereas HTML5 applications reside on a Web server. Increasing development and deployment of hybrid application can be supported by a MEAP, which incorporates a native application (e.g., the Hybrid Web Container (HWC) of the SUP [3, p. 21]) typically acting as a shell over the *UIWebview* in iOS or *WebView* in Android embedding Web content.

An analytical company Gartner predicts [4] that more than 50% of mobile applications deployed by 2016 will be hybrid, and has developed a concept [2, p. 2] for enterprises, which consider the MEAP approach for their mobile initiatives. According to that concept, an enterprise that plans to introduce three MBAs or to utilize three mobile Operating Systems (OSs) or to integrate at least three back-end data sources should deploy a MEAP. Drawing upon this concept Gartner says [2, p. 4] that *95% of organizations will choose a MEAP or packaged mobile application as their primary mobile development platform.* This emerging technological environment presents new opportunities for software reuse.

Using the SPL concept [26] it is possible to introduce appropriate variability management into the context of hybrid MBAs to meet demanding customization needs of BPs across various enterprises. At the same time the technological basis of an SPL should facilitate an agile adoption of new requirements in evolving BP environments. Therefore, a provider of hybrid MBAs can turn into a *lissome* BP enabler for enterprises with a new quality of industrial software production. In this vein, an SPL-based approach should enable flexible development of hybrid MBAs with increased efficiency of software reuse.

Finally, the recent research literature on SPL approaches includes materials for modern software systems, which accommodate complex characteristics of variability. Such systems address variability problems cutting across multiple layers [29], spanning different views [5], handling the product configuration in stages [13, 33]. Such concepts can be applicable also to hybrid MBAs to facilitate their variability management in the sophisticated environment they are settled in.

3. RESEARCH METHODOLOGY

The research described herein is conducted within the framework of a cooperation project with our industrial partners in the *Mobile Devision* at T-Systems[3]. The project

[2]http://www.phonegap.com (Last access: July 7, 2013.)
[3]http://www.t-systems.de (Last access: July 7, 2013.)

aims at designing methods for systematic variability management of hybrid MBAs and producing development artifacts supporting their reuse-oriented realization and flexible adoption across different enterprises.

This endeavour corresponds to the *formulative* research approach that is identified based on an analysis of articles in both *Software Engineering* and *Information Systems* as one of the major research approaches [24]. It focuses on formulating methods, processes, algorithms, as well as models, and strives to achieve better ways to build software systems [16, p. 502]. The formulative approach is also known to be in software engineering the most prevalent one [16, p. 501]. Further, it is similar in *Information Systems* to the *design science* research methodology that is technology-oriented and *attempts to create things that serve human purposes* [22, p. 253]. Thus, design science consists of two basis activities: *building* an artifact and *evaluating* it [22, p. 254]. Such artifacts for hybrid MBAs will include, e.g., variability modelling concepts and design patterns for their implementation, a software reuse methodology.

Evidently, prior to building an artifact formulation of its atributes has to be done, which can be documented in research documentation. A common and unified way to create software engineering documentation provides the Unified Modelling Language (UML) [1]. Thus, by using activity diagrams (ADs) of the UML it is possible, e.g., to represent (variable) workflows [17] of software systems, which support BPs, or by utilizing class diagrams to create a meta-model that formalizes a variability modelling approach [5]. Further documentations will be done pictorially, in prose, as code patterns and formulae, whenever appropriate.

Further, introducing an SPL requires some additional up-front investment that is necessary for building reusable assets, transforming the organisation and preparing it for the transition. Herein different strategies exist to adopt an SPL [26, p. 395-400], e.g., the *big-bang strategy* or *incremental introduction*. In order to mitigate costly up-front investments this work utilizes the strategy of the incremental introduction. It starts, usually, as a single work group doing SPLE and needs a limited amount of money and time spent on the transition within any particular period [26, p. 408].

The domain engineering phase of the SPLE will focus on creating models of BP domains supported by hybrid MBAs. This includes BP models and domain requirements analysis. There are known approaches for BP-oriented software systems, which capture variability of activity diagrams (ADs) underlying BPs, e.g., by using UML stereotypes [27, 19] or custom UML extensions [8]. Recently such approaches have been identified by Heuer *et al.* as semi-formal [17] and, thus unsuitable for entirely automated processing with tools. The authors proposed a formal approach that defines variability relations between variable workflow features and edges in the Domain Activity Diagram (DAD) of an application.

Thus, variability of BP models for hybrid MBAs will be documented as Domain Activity Diagrams (DAD) based (for better automation) on the state-of-the-art, formal approach, the theoretical foundation of which has been proposed by Heuer at al. in [17]. It is based on Petri nets [25], which enables one to apply techniques for Petri nets analysis to DADs, e.g., for verification of structural flaws. Requirements analysis will be done via elicitation from customers and interviews with domain experts at T-Systems. The domain analysis will be performed to design common and variable assets for an SPL. The feature modelling approach [20] will be used as a visual formalism that documents variability and commonality in terms of features and provides high-level representations of the capabilities of reusable artefacts.

For an implementation of artifacts tools of the MEAP, i.e., the SUP [3], that is used in this project, will be exploited whenever possible for better integration. For instance, with the Mobile Workflow Form Editor (MWFE) that is available with the SUP one can design DADs, yet without variability relations[4]. Therefore, an SPL integrated into the MEAP's environment should enable variability management in workflows, and for further hybrid MBAs' considerable features.

Further, since the authors will tightly collaborate with their industrial partners during the project, this research poses a pragmatic, result-oriented view on research methodology [30] that aims at advances gained by adapting research results in technical areas, such as the context of hybrid MBAs. Therefore, this research will use the *Concept Implementation* research method [16, p. 501], i.e., *proof by implementation* that is accompanied by industrial acceptance considered as validation of the research [11, p. 351]. Thus, an SPL for hybrid MBAs will be implemented for partial evaluation.

Furthermore, *qualitative evaluation* of the SPL architecture will be conducted via scenarios [14],[26, p. 224]. The use of scenarios to evaluate software architectures is one of the best industrial practices [23, p. 34]. The most of existing evaluation methods and experiences for product-line architectures known in the literature and research are scenario-based [14, p. 178, 181-182]. Finally, the research will be supported by an empirical analysis in a real-world industrial context conducted through case studies in different BP domains.

4. TYPICAL VARIATION POINTS

In this section we highlight exemplary variation points of hybrid MBAs, and show where they typically occur. We also discuss their important relationships on an example.

In the research area of software reuse in the mobile domain there exists, e.g., a framework developed by Brans and Basole [9] to address a higher degree of software reuse in development of mobile applications; and a roadmap for research in mobile business [15]. The existing literature indicates that when applying mobile technology to an enterprise applications have to be developed with respect to specific requirements of that particular enterprise. Brans and Basole [9, p. 152] identified several reasons for that fact, which is relevant for hybrid MBAs and their technical setting. This includes different back-end systems among enterprises, variable work processes and constraints, various mobile devices and OSs in use.

Hybrid MBAs are typically designed once to run on multiple platforms and mobile devices, which is supported by platform agnostic Web technologies; and various back-end systems can be managed by the MEAP. Yet, modelling variable workflows in BP-oriented variability-aware software systems is evidently an indispensable, challenging task. For hybrid MBAs there are two distinctive types of variabilities on the workflow level, which have to be considered herein:

- *Variable activities*, e.g., those, which realize different authentication mechanisms or data input methods, and

[4]The MWFE does not provide this functionality.

- *Variable transitions* – optional possibilities to move between activities, e.g., a “go-back” transition in the workflow that might be necessary to go back and check (and also correct, if needed) some data inputted manually by the user at an earlier (variable) activity, yet can be removed, if this data is properly provided by an automatic means at the same workflow step with another activity.

Concrete features of these variability types change the structure of workflows of variability-aware hybrid MBAs. Such features represent a characteristic group of considerable features of hybrid MBAs. At the same time, there are further features, which do not influence hybrid applications' workflows, such as workflow constraints or languages of the graphical interface. These features belong to the second distinctive group of features.

For instance, varying workflow constraints[5] can be realized in JavaScript and integrated into the hybrid MBA's package. The SUP provides herein useful support – an automatically generated *Custom.js* file that has an event-driven architecture and contains various (stub) methods, which constraints can be integrated into. This file represents a *minimal core* that can be extended according to the *inject* pattern [32] whenever necessary constraints are selected for the application. Such workflow constraints are useful to check relevant BP conditions during the execution of hybrid MBAs, and can be defined at certain activities of the workflow. Activities can also have multiple constraints defined at them at once.

Evidently, activities and workflow constraints, which are defined at them, can pose variability relationships of the type *requires*, which span across the previously mentioned two characteristic groups of variable features. In order to be applied each defined constraint requires one or more activities to be selected (if they are variable) in the workflow, which it is associated with, otherwise it is not applicable. The same holds for native features (e.g., a camera of the mobile device), which can be used in certain variants of activities.

Therefore, from the configuration point of view it is important first to decide upon variants of features of the first category and to form a concrete workflow for the hybrid application prior to configuring features of the second category. Otherwise, one might end up selecting features, such as workflow constraints, which can later turn out to be not required for the workflow of a hybrid application that is being instantiated. Although it might be possible to automatically include required activities when workflow constraints are being selected by using dependencies defined in the feature model, forming a workflow by selecting features, which do not influence its structure, is either less or not intuitive.

Thus, deriving workflows of hybrid MBAs should be a primary task. Further configurations can be regarded as either more fine-grained settings applied to selected activities of the workflow of a hybrid MBA or as those, which correspond to a basis functionality that does not depend on features of the workflow level.

These instantiation aspects of hybrid MBAs have to be considered during the configuration and derivation process. This process can be realized in a two-staged fashion performed by an SPL.

[5]Workflow constraints are features of hybrid MBAs and should not be confused with the feature model's constraints.

5. CONCEPTUAL DESIGN

In the following we propose a conceptual variability modelling approach and discuss its important design decisions. We highlight locations of variation points within the technological environment that hybrid MBAs are settled in.

We apply to the variability problems concerned with the two described groups of features (Section 4) the *separation of concerns* (SoC) methodology [18]. In particular, we discern two concerns:

- Formation of variable workflows, and
- Feature modelling for hybrid MBAs.

We expect this separation to facilitate a two-staged configuration and derivation process for hybrid MBAs where each stage is dedicated to a given concern and focused only on the relevant parts of the feature space. Therefore, we introduce a *dualistic-view* variability model and define it as the composition of a *Business Process view* (*BP view*) and a *Feature view*, Figure 1.

Typically, a hybrid MBA can be represented as a workflow of interconnected activities (HTML5 *forms*) executed within a native application (container). Such workflows supporting the same BP can potentially comprise variable activities and/or transitions between activities on the cross-enterprise scale. Thus, we capture in the *BP view* those features, which change the structure of workflows, whereas the *Feature view* accommodates additional features for hybrid MBAs, such as workflow constraints or native features on the *application layer* (AL) as well as various back-end entities for productive data on the *data layer* (DL). The *data access abstraction layer* is a part of the MEAP; though there is no variability herein, it plays an important role for application deployment, object-relational mapping (ORM) and data access. Thus, the *dualistic-view* variability model enables us to capture all features integrally and link the *BP view* with AL and DL joined within the *Feature view* and define inter-view relations spanning the *Cross-view boundary*, Figure 1.

Though variation points of variability-aware hybrid MBAs are distributed across the systems' layers, they might still be tightly connected through inter-layer dependencies, which create a cross-layer cut of the *Feature view*, and inter-view ones, which span the *Cross-view boundary*. For instance, if data retrieved from the user input on the application level has a certain variable format, then on the data layer a corresponding database (an alternative would be a standard business management software system) has to be selected with

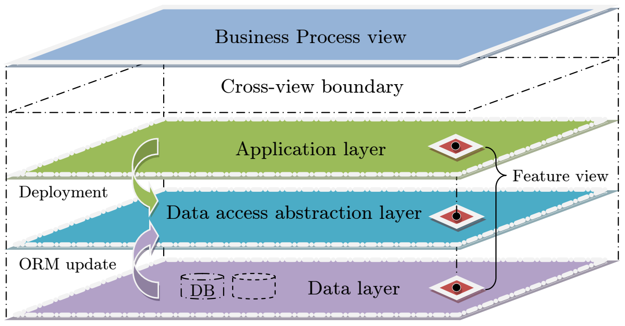

Figure 1: The conceptual variability model of hybrid MBAs with a dualistic-view composition.

required elements prepared in the relational scheme for incoming tuples. This can be realized with one variation point that describes the required data structure (format). Then code generation can be utilized to derive the user interface, the data structure, the database table creation statements (SQL code) (cf. [32, p. 4-5]) from the model of this variation point. Also the ORM on the data access abstraction layer has to be adjusted at the MEAP in order hybrid MBAs can store their data in a database or any other back-end entity. Furthermore, workflow constraints, which are modelled in the *Feature view*, require relations to certain activities in the *BP view*, which they are associated with.

The introduced views can be illustrated on a concrete exemplary application that supports a BP. Herein, MBAs for a widespread BP – *cash outlay request* (COR) can be utilized. A cash outlay is money employees of an enterprise pay for their operating activities. Such charges are covered in a COR that is processed and approved or denied by a department of the enterprise. MBAs can facilitate this BP by easing access to information and reducing the time effort spent on processing requests. Relevant documents, such as bills or tickets can be digitalized by using, e.g., the mobile device's camera. The approving entity can then process such requests with a mobile device ubiquitously. Thus, the BP is supported by two MBAs: 1) a *COR application* for issuing requests and 2) an *approval application* for processing them.

On the cross-enterprise scale different aspects of this BP can vary, which leads to variable features in the applications supporting it. Herein, the COR application can require, e.g., different authentication mechanisms: employees authenticate themselves by using their credentials or a certificate-based authentication process is utilized. Further, employees may require to identify themselves in the back-end system, which can be done manually or by using an electronic badge equipped with a Radio Frequency Identification (RFID) tag. Such variable features are realized with means of the MEAP (Section 6) as activities within the DAD of the COR application as required by the BP, and captured in the *BP view*.

Additionally to the features of the workflow level there are further features, such as the Graphical User Interface (GUI), internationalization, the camera with a fixed or variable resolution. Also workflow constraints can be used in the COR application to validate the user input, such as credentials and during manual identification with respect to proper formats, allowed values. These features are captured in the *Feature view*. Features of the concomitant approval application are captured in the same way.

Further, there are known multiple-view approaches, e.g., for general software architectures [21] and for variability modelling in SOA systems [5]. To address a particular set of problems Kruchten designed a 4+1 view model of software system architectures [21]. The author described that four specific views can be used to capture system design decisions and one view (use cases) can be utilized to integrally relate and unify them. Unlike Kruchten's model that tries to unify design decisions, our dualistic-view variability model strives to separate variability concerns. Thus, it enables us to mould MBAs on the workflow level as well as to configure them more fine-grained within the scope of single activities.

The authors in [5] designed a multiple-view variability model for SOA and defined inter-view relationships. They used a unifying feature view to avoid variability dispersion across different views. Similarly, we use a composite dualistic-view to jointly represent features of a hybrid MBA during feature modelling, and define their relationships (e.g., "requires"). However, during the derivation phase we apply a two-staged configuration process and apportion the entire feature space among stages. The authors used the UML's stereotype extension mechanism [1] to model variability in the *BP view* and tailor service activities into BPs. We realise our *BP view* in the formal fashion based on DADs cut into ADs, which can be supported by an automated tool.

Further, there is an attempt of a smart composition of reusable software components in mobile application product lines [28]. The authors consider the problem of porting mobile applications to heterogeneous devices, which may lead to a significant increase in software variability. In our work we utilize the hybrid Web approach that produces platform agnostic applications, which are written once, but run on all major devices and platforms due to portability of Web technologies. We focus on the variability aspects of hybrid MBAs arising on the cross-enterprise scale where BPs vary.

At the present time, we have produced the conceptual variability model for hybrid MBAs as described in this section. Research into typical variation points of hybrid applications has been performed. The *BP view* has been prototypically implemented as a part of the SPL (Section 6). We are working currently on a conceptual design of the *Feature view* and its realization. The *Feature view* will accommodate considerable features of hybrid MBAs, such as workflow constraints, languages, or native functionalities. During designing the *Feature view* we plan also to further explore variability aspects of hybrid MBAs. It can be advantageous to extend our current approach and introduce additional views, such as *an application view* and *a back-end system view*. This might help to "uncouple" features of the application layer from the ones related to the back-end system and to analyse their relationships.

6. REALIZATION ASPECTS

Since the SUP does not provide appropriate means to systematically capture considerable features of hybrid MBAs discussed in Section 4, we design our variability model upon the SUP and integrate an SPL into its technological environment. This section describes a feature modelling approach and a realization of the *BP view*.

For feature modelling and configurations of the SPL we utilize the *FeatureIDE* extensible framework [31] due to its direct integration with the Eclipse-based[6] Sybase Unwired Workspace (SUW) that is a part of the SUP. However, it does not support views for feature models explicitly. Nevertheless, the feature model for a hybrid MBA can be extended with two abstract features: *BPView* and *FeatureView*, which are the direct children of its root feature. By doing so we can fairly rapidly group the entire feature space into our two views (abstract features) within the *FeatureIDE's* graphical editor and avoid a time-consuming feature enumeration [18]. Each feature can belong to only one view.

The configuration of features is, again, accomplished with *FeatureIDE* that generates from the feature model hierarchical tree structures (branches) and enables one to do the configuration in two stages. During the first stage one configures all child features of the *BPView* and forms a concrete workflow for the hybrid MBA, and then proceeds to the *Fea-*

[6]http://www.eclipse.org (Last access: July 7, 2013.)

tureView branch. In this branch one can configure all further features, such as workflow constraints, native functionalities, or branding.

This realization of the dualistic-view composition of features always produces valid configurations, since *FeatureIDE* is aware of the entire feature model and enforces all constraints including those spanning the *Cross-view boundary* properly. It creates also a basis for proper two-staged product derivations.

Further, a *Sybase mobile workflow* is designed within the MWFE in order to realize the DAD of a hybrid MBA including all features of the *BP view* documented in the feature model. The SUP compiles the DAD into an HTML5 structure of the workflow, resulting *forms* of which contain all activities including transitions simultaneously. There are no yet variability relations defined in the DAD, since the MWFE does not provide this functionality. The process of variability definition, that is to be done in the DAD, is based on the formal approach proposed by Heuer *et al.* in [17].

During this process all features of the *BP view* are mapped to corresponding edges in the DAD that is represented with the UML semantics. Therefore, the DAD that is designed with the MWFE should be transformed by an adapter[7] for the representation with respect to the UML semantics. Herein, the SPL should provide regarding the variability definition process supporting means, e.g., a graphical mode for displaying the transformed DAD, and loading all features of the *BP view* from the feature model, which variability relationships have to be defined for.

In general the variability definition process needs to be done only once for the DAD of a BP domain; it can be realized as follows. The *BP view* features (activities/transitions) are loaded by the SPL along with the DAD of a hybrid MBA. Thereafter, each feature is graphically correlated to DAD's edges, which become variable. For each variable edge a relation is created that contains names of the compiled DAD's elements, which are interconnected by this (directed) edge.

In this way a binding connection between the feature model and the DAD representing the MBAs' variable workflow is established. Then, based on a feature selection a concrete activity diagram can be derived from the DAD for the hybrid application. This is done by removing from the DAD edges related to the features, which are not selected, along with corresponding activities.

Edges (transitions) between different activities of the DAD can be implemented as HTML5 *buttons*. Then, the SPL can parse HTML5 and remove from the Document Object Model (DOM) structure those *buttons*, the related edges of which are not present in the activity diagram derived for a specific hybrid application. Thereafter, HTML5 *forms* (activities), which do not contain any transitions, are removed from the HTML5 structure of the DAD as well.

7. FUTURE WORK

In our forthcoming work we will compete the implementation of the SPL for hybrid MBAs according to the presented conceptual design and conduct case studies in different BP domains. In addition to the formal realization of the *BP view* we will design a meta-model for our approach on the system level to formalize the approach. Qualitative evaluation of the SPL architecture will be performed via scenarios. During this process we will document gained knowledge that will be used to enhance the definition of concepts and implementation techniques for emerging hybrid MBAs.

Industrial case studies will be conducted with our partners at T-Systems. We will conduct multiple interviews with T-Systems experts in the *Mobile Enterprise* division in order to discuss variability of relevant BP domains. Based on the domain analysis common and variable assets will be identified, and then realized within the MEAP's technological environment.

[7]This transformation is rather straightforward.

8. CONCLUSIONS

In this paper we have discussed new opportunities for software reuse in the domain of hybrid MBAs. We have proposed a research methodology, and explored how variability problems of hybrid applications settled in the complex technological environment can be addressed with the SPL concept. Typical variation points, their occurrence and relationships have been highlighted. The dualistic-view variability model supporting the two-staged configuration and derivation process for hybrid MBAs has been designed. We hope that the described aspects of the important research field for software reuse can be beneficial for both academia and industry.

9. ACKNOWLEDGEMENTS

This work has been partially supported by the Collaborative Research Center 1053 funded by the German Research Foundation and by the LOEWE Priority Program Cocoon funded by the LOEWE research initiative of the state of Hesse, Germany.

I, Viktor Kolokolov, sincerely thank my parents, my loving mother Larisa and my dear father Vasiliy, and my sister Olga for your kindest support and encouragement during work.

10. REFERENCES

[1] OMG (2004), UML 2.0 superstructure specification, technical report. Technical report, Oct. 2004.

[2] Gartner, magic quadrant for mobile enterprise application platforms. URL: `http://www.gartner.com/id=1257121`, Dec. 2009.

[3] Sybase unwired platform 2.0: Fundamentals. URL: `http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc01204.0200/doc/pdf/sup_fundamentals.pdf`, Apr. 2011.

[4] Gartner says by 2016, more than 50 percent of mobile apps deployed will be hybrid. URL: `http://www.gartner.com/newsroom/id/2324917`, 2013.

[5] M. Abu-Matar and H. Gomaa. Variability modeling for service oriented product line architectures. In *Proceedings of the 2011 15th International Software Product Line Conference*, pages 110 –119, Aug. 2011.

[6] L. Balzerani, D. D. Ruscio, A. Pierantonio, and G. De Angelis. A product line architecture for web applications. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 1689–1693, New York, NY, USA, 2005.

[7] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, Sept. 2010.

[8] A. Braganca and R. J. Machado. Extending UML 2.0 metamodel for complementary usages of the <<extend>> relationship within use case variability specification. In *Proceedings of the 10th International on Software Product Line Conference*, SPLC '06, pages 123–130, Washington, DC, USA, 2006. IEEE Computer Society.

[9] P. D. Brans and R. C. Basole. A comparative anatomy of mobile enterprise applications: Towards a framework of software reuse. *Inf. Knowl. Syst. Manag.*, 7(1,2):145–158, Apr. 2008.

[10] A. Charland and B. Leroux. Mobile application development: web vs. native. *Commun. ACM*, 54(5):49–53, May 2011.

[11] L. Chen and M. Ali Babar. A systematic review of evaluation of variability management approaches in software product lines. *Inf. Softw. Technol.*, 53(4):344–362, Apr. 2011.

[12] K. Czarnecki, T. Bednasch, P. Unger, and U. Eisenecker. Generative programming for embedded software: An industrial experience report. In D. Batory, C. Consel, and W. Taha, editors, *Generative Programming and Component Engineering*, Lecture Notes in Computer Science, pages 156–172. Springer Berlin Heidelberg, Jan. 2002.

[13] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. In R. L. Nord, editor, *Software Product Lines*, Lecture Notes in Computer Science, pages 266–283. Springer Berlin Heidelberg, 2004.

[14] L. Etxeberria and G. Sagardui. Product-line architecture: new issues for evaluation. In *Proceedings of the 9th international conference on Software Product Lines*, SPLC'05, pages 174–185, Berlin, Heidelberg, 2005. Springer-Verlag.

[15] K. G. Fouskas, G. M. Giaglis, P. E. Kourouthanassis, S. Karnouskos, A. Pitsillides, and M. Stylianou. A roadmap for research in mobile business. *Int. J. Mob. Commun.*, 3(4):350–373, May 2005.

[16] R. Glass, I. Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44(8):491–506, June 2002.

[17] A. Heuer, C. J. Budnik, S. Konrad, K. Lauenroth, and K. Pohl. Formal definition of syntax and semantics for documenting variability in activity diagrams. In *Proceedings of the 14th international conference on Software product lines: going beyond*, SPLC'10, pages 62–76, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] A. Hubaux, P. Heymans, P.-Y. Schobbens, D. Deridder, and E. K. Abbasi. Supporting multiple perspectives in feature-based configuration. *Software & Systems Modeling*, pages 1–23, Nov. 2011.

[19] E. Kamsties, K. Pohl, S. Reis, and A. Reuys. Testing variabilities in use case models. In F. J. v. d. Linden, editor, *Software Product-Family Engineering*, number 3014 in Lecture Notes in Computer Science, pages 6–18. Springer Berlin Heidelberg, Jan. 2004.

[20] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Nov. 1990.

[21] P. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6):42 –50, Nov. 1995.

[22] S. T. March and G. F. Smith. Design and natural science research on information technology. *Decis. Support Syst.*, 15(4):251–266, Dec. 1995.

[23] M. Matinlassi, E. Niemelä, and L. Dobrica. Quality-driven architecture design and quality analysis method: A revolutionary initiation approach to a product line architecture. *VTT Technical Research Center of Finland, ESPOO2002*, 4(5):6, 2002.

[24] J. Morrison and J. F. George. Exploring the software engineering component in MIS research. *Commun. ACM*, 38(7):80–91, July 1995.

[25] C. A. Petri. Kommunikation mit Automaten. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM, No. 3 (1962); Also, English translation: Communication with Automata, Griffiss Air Force Base. New York. Tech. Rep. RADC-TR. 1 (suppl. 1), 1966.

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

[27] S. Robak, B. Franczyk, and K. Politowicz. Extending the UML for modelling variability for system families. In *Applied Mathematics and Computer Science*, volume 12, pages 285–298, 2002.

[28] R. E. V. Rosa and V. F. Lucena, Jr. Smart composition of reusable software components in mobile application product lines. In *Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering*, PLEASE '11, pages 45–49, New York, NY, USA, 2011. ACM.

[29] H. Schirmeier and O. Spinczyk. Challenges in software product line composition. In *42nd Hawaii International Conference on System Sciences, 2009. HICSS '09*, pages 1 –7, Jan. 2009.

[30] C. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, 1999.

[31] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich. FeatureIDE: an extensible framework for feature-oriented software development. *Science of Computer Programming*, 2012.

[32] M. Völter. Handling variability. Version 2.0, December 11, 2009.

[33] J. White, B. Dougherty, D. C. Schmidt, and D. Benavides. Automated reasoning for multi-step feature model configuration problems. In *Proceedings of the 13th International Software Product Line Conference*, SPLC '09, pages 11–20, Pittsburgh, PA, USA, 2009. Carnegie Mellon University.

[34] M. Yoshida and N. Iwane. An approach to the software product line system for web applications. In *International Conference on Computing Informatics, 2006. ICOCI '06*, pages 1 –6, June 2006.