Computer Science and Information Systems 12(3):961–977 DOI: 10.2298/CSIS141031035N

Context Modeling based on Feature Models Expressed as Views on Ontologies via Mappings

Siniša Nešković[1], Rade Matić[2]

[1] University of Belgrade, Faculty of Organizational Sciences
Jove Ilića 154, 11000 Belgrade, Serbia
sinisa.neskovic@fon.bg.ac.rs
[2] Belgrade Business School, Kraljice Marije 73,
11000 Belgrade, Serbia
rade.matic@bbs.edu.rs

Abstract. This paper presents an approach for context modeling in complex self-adapted systems consisting of many independent context-aware applications. The contextual information used for adaptation of all system applications is described by an ontology treated as a global context model. A local context model tailored to the specific needs of a particular application is defined as a view over the global context in the form of a feature model. Feature models and their configurations derived from the global context state are then used by a specific dynamic software product line in order to adapt applications at runtime. The main focus of the paper is on the realization of mappings between global and local contexts. The paper describes an overall model architecture and provides corresponding metamodels as well as rules for a mapping between feature models and ontologies.

Keywords: context modeling, feature models, dynamic software product line, ontology, model mappings.

1. Introduction

The Internet of Things (IoT) vision assumes a very large number of "smart" devices or objects which can mutually interact and dynamically respond to events in their surrounding by adopting their behavior correspondingly [32], [27], [3]. From the software point of view, IoT consists of a large number of software applications (running on and managing "smart" objects) which react to users and their surroundings without user's explicit commands. In other words, IoT requires a development of a context-aware self-adapted system (CASAS) which possess an adaptation mechanism enabled to timely end highly efficient adapts applications at runtime according to changes in their context.

The development of such an adaptation mechanism is usually based on a single context model and rules that specify which configurations of the applications should run in every possible instance of the context [2], [5], [17], [12], [11], [13], [25]. However, in case of IoT, consisting of very large number of context-aware applications, it is very difficult and inefficient to use a single context due to its complexity. Such a single global context must include all information needed by all applications ("smart" objects), i.e. data about a large number of different situations and different users with different

interests and views. Thus, an adaptation of a single working application must deal with the entire context including a large amount of context classes and objects (both metadata and data) that are mostly irrelevant. Therefore, such application adaptation is difficult to specify and implement in an efficient manner. Having in mind that the number of smart objects in IoT is increasing exponentially, the problem of an efficient application adaptation is becoming crucial for the IoT vision to be realized to its full potential.

A possible solution to this problem could be to use separate local contexts tailored for each particular application. Having in consideration only the relevant context information will certainly reduce response time and increase the reasoning performance in the adaptation process. However, such solution is also connected with many difficulties due to synchronization and potential inconsistencies among a large number of different local contexts that possess overlapping contextual information. Therefore, although efficient, such solution would lead to an inadequate or incorrect behavior of "smart" objects, i.e. instead of solving the problem it would create an even more serious one.

This paper presents an approach to the problem of self-adaptation in such complex CASAS which is based on the usage of both global and local contexts. Global context is treated as an ontology describing contextual information required by all applications, whereas local contexts are derived as views over the global context tailored to the needs of particular applications. Views are defined through mappings (correspondences) between modeling elements of global and local contexts. Additionally, local context models in our approach are expressed in the form of feature models (FM) [21], which are commonly used in software product line engineering (SPLE) to enable efficient generation of application variants customized to specific needs of users. In our approach, derived feature models are used to instantiate variants of context-aware applications corresponding to a specific context state. Thereby, our approach relies on so called dynamic software product lines (DSPL) [14] as the main adaptation mechanism in CASAS.

The main focus of this paper is on describing the realization of mappings between global and local contexts. The rest of the paper is structured as follows. The next section gives an overview of work related to our research. In Section 3 our approach is described by an overall model architecture, a brief description of the adaptation process and detailed descriptions of models used to realize global and local contexts. An example to illustrate our approach is also given. Implementation aspects of a CASAS based on our approach are discussed in Section 4. Finally Section 5 concludes the paper.

2. Related Work

As the next evolution of the Internet, IoT envisions a world where all the objects around us are interconnected and behave smartly, i.e. they know what we like, what we want or need and can act accordingly. However, this vision poses a lot of technical challenges which makes its realization a difficult task. More about the IoT as a new computing paradigm, its potentials and possible applications, methods and models used to overcome challenges as well as existing middleware solutions can be found in [27]. One of the main challenges that this paper identifies is the context-awareness of smart

objects. Hence, a lot of research has been recently dedicated to context modeling and development of context-aware systems.

Regarding context modeling, several techniques are proposed in [27], [3], [2], [4], [33], [29]. Ontologies are the most expressive and most used technique for modeling contextual information. Based on semantic web languages such as RDF and OWL, the application of ontologies for context modeling provides a unique way to specify key concepts (as well as their instances) enabling thus reuse and sharing of information or knowledge about the context in distributed systems. However, ontologies are not without certain drawbacks [4], [18], [34], [35]. Most of suggested ontologies do not provide a clear description of contextual information. Although different ontologies have been proposed to model domain specific context information that are reusable in many domains, they cannot be reused directly as all of them have certain drawbacks in generality and/or dynamicity [2]. Additionally, as it is observed in [27], the processing of contextual information can be very resource intensive and slow when semantic web languages are used for expressing complex contextual ontologies (which in IoT is normally the case).

Approaches defined in [17], [9], [1], [16], [24], [30] use feature modeling as a technique for context modeling and development of context-aware self-adaptive systems in order to improve reusability and new configurability. A feature model (FM), firstly proposed by Kang et al. [21], consists of a hierarchy (represented by a tree diagram) of mandatory and optional features of some product within a specific domain. Since then it had many extensions proposed in order to enhance semantic expressiveness such as adding multiple feature instances and their cardinality constraints [7]. Group constraints, cloning, attributes and additional relationships are also advanced variability extensions because they give more descriptive power of feature modeling. A feature notation may also support concepts like annotations and FM references. FM references allow dividing large feature models into smaller ones. Annotations can capture additional information.

FMs are extensively used in software product line engineering (SPLE), an approach which intends to industrialize the development of similar software applications (a product family) by utilizing software production lines [26]. A software production line (SPL) represents a development platform consisting of all necessary production assets like tools, models, frameworks, processes, etc., which enable massive and efficient production of family members, i.e. applications tailored to the needs of a particular customer. FM with its mandatory and optional features is used to describe common and variable parts of a software product family. In this way FM defines the set of all configurations that are valid for applications in the particular domain. An application is produced by SPL through the process of configuration, a selection of features from the corresponding FM which suit the customer requirements.

In order to support adaptation at run time and enable context aware computing dynamic software production lines (DSPL) were introduced [12], [23], [15]. Most of DSPL approaches utilize two separate FMs. One FM is used to model the context while another FM represents functional and/or non-functional features of the software system (applications) operating in this context. These two FMs are usually related using feature dependencies expressed via requires and excludes relationships, but many of the approaches use some other special techniques to do that. For example, Fernandes et al. [9] provide a modeling notation that extends feature models to model context-aware systems. The general feature model is composed of a domain feature model with composition rules, a context feature model with context definitions, and context rules that

make the link between these two models. The formalism and notation used to represent feature models are not standard and their approach differs from our work in using a domain-specific language which allows a developer to specify context rules. Adaptation is guided by the context rules which specify how a specific context affects an application configuration in the domain, deciding about variant selection in a variation point.

A similar idea was shown in [1]. The paper addresses the reconfiguration of Dynamic Adaptive Systems where FMs are used to represent the context and the software system using standard formalism and notation. Except the feature dependencies between the contextual feature model and the application feature model that affect the selection of features, a group of contextual features determines product goals and attributes, which also constrain the selection of product features [24].

In [30] it is shown how requirements of runtime contexts are specified via require and exclude cross-tree constraints between the context model and the feature model i.e., context-aware DSPL variability specification. Based on this context-aware feature model (CFM), they introduce a transition system that provides appropriate reconfigurations at runtime. Every state represents potential configurations of the DSPL which satisfy a context or a set of context.

The framework explained in [17] is similar to our research because the view concept is used too. The difference is the technique how FM is expressed as a view on ontology. Their framework treats the context as a DSPL composed of a set of small contextual pieces, namely context primitives, which are elements of an ontology-based context model (OCM). Context feature model and OCM serve as a context model family. Context primitives are annotated using existence conditions and metastatements. Annotations are defined by features and feature attributes from the context feature model, and can be evaluated by considering a feature configuration. Based on this feature configuration, the corresponding context product (instance of a context model family) is generated automatically.

3. Our approach

Our research was primarily motivated by the idea that FMs can be expressed as views on ontologies [6] as well as that the derivation of a FM from an ontology should be very efficient in order to fulfill highly demanding requirements of context aware computing in IoT. Additionally, in order to support a wide range of existing and emerging technologies that can be used as an implementation platform, our approach is based on model driven engineering as development methodology [10]. This means that we use so called platform (i.e. technology) independent models (PIM), which are then transformed to highly efficient platform specific models (PSM) used for the implementation.

Contextual information shared by all system applications is described by an ontology treated as a global context model. Since usual ontology languages like RDF or OWL are not so efficient for large scale context information processing, we are using Entity-Relationship (ER) model instead as PIM for context modeling. ER model is a semantic data model with concepts which are semantically rich enough to express ontology [8], [19]. In addition, an ER model can be easily translated to some highly efficient implementation based on some DBMS.

To support efficient adaptation based on DSPL, we use local context models tailored to the specific needs of particular applications, which are defined as views over the global context in the form of FM. Views are defined through mappings (correspondences) between modeling elements of global and local contexts, i.e. between ER schema concepts and FM concepts. These definitions are then used to derive a FM configuration from the global context state used by DSPL in order to adapt applications at runtime. More about these implementation details is given in Section 4.

The rest of this section describes our approach in more detail. We first provide an overall model architecture, which identifies all PIM models (including their metamodels) and their relationships required in our approach. We also briefly describe the adaptation process based on the introduced models. Then we show how ER models are mapped to FM. At the end of the section we provide an illustrative example of our approach.

3.1. Model Architecture

The overall model architecture is shown as an UML package diagram in Figure 1. UML packages in the diagram represent models, whereas various relationships among models are represented by stereotyped dependency associations between corresponding packages. The diagram also classifies models in three different categories according to the time when they are created (represented as swim lanes in the diagram):

- *Runtime* category encompasses models which are automatically created and maintained during runtime of CASAS components and user applications.
- *Design time* category encompasses models which are created by developers during the design of CASAS components and user applications.
- *Metamodels* category encompasses models which are defined by our approach, i.e. metamodels which are introduced in the next subsection of the paper.

Since ER data model is used as an ontology definition language [8], [19], [31], [28], a global context model is defined using an ER schema. Hence, it must conform to ER metamodel. On the other hand, FMs are used to represent local context models which must conform to the corresponding FM metamodel.

FMs are defined as views on ontologies, namely, as projections of the ontologies from different viewpoints [6]. The views definition is given by a mapping model which maps the concept of an ER schema to concepts of a FM. The defined mappings have to follow rules and constraints, which are defined by the mapping metamodel.

An ER schema serving as the application global context model is created by system architects (chief system developers) usually using existing (one or more) ontologies, which are possible expressed in different ontology languages (e.g. OWL). These ontologies are then combined and tailored to the needs of the application and expressed as an ER schema. FMs are created by use case developers and application programmers, who are also responsible for defining mappings between the ER schema and FMs. Detailed description of the development methodology and organizational aspects of development are beyond the scope of this paper.

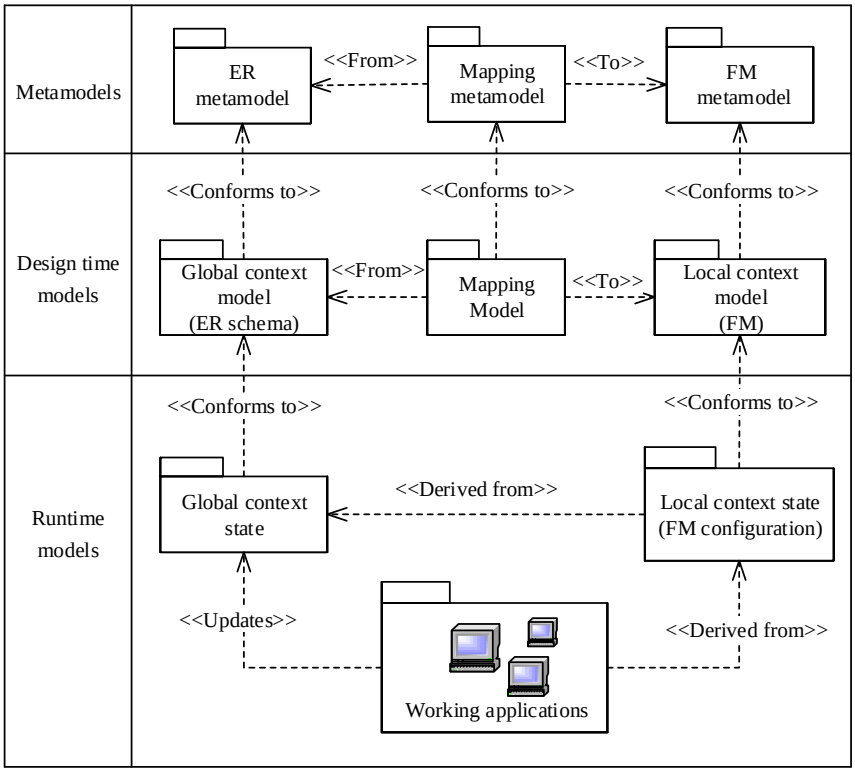

Fig. 1. Model architecture

At runtime level, a CASAS maintains a global context state, which keeps contextual information at the particular moment. The global context state is usually realized as some form of a database structured according to its ER schema defined in the design time. The database is updated, i.e. the global context state is maintained, by context-aware applications (sensors and smart objects) and/or other CASAS's run time components.

When significant changes of a context are detected, CASAS run time components will trigger an adaptation process which will instantiate (generate) affected running applications. The adaptation process consists of two main steps:

- *Derivation of local contexts*. Local context states for affected running applications are derived from the current global context state. They use the corresponding mapping models defined at the design time. According to SPL engineering principles, the local context state is represented as a FM configuration.
- *Instantiation of running applications.* Using a DSPL, affected working applications are instantiated based on corresponding FM configurations. Thus, a new instance (version) of the application is adapted to the current local context state.

The adaptation process is performed by a part of CASAS called Adaptation Manager. Due to space limitations, a detailed description of the adaptation process and Adaptation Manager is not included here.

3.2. Metamodels and Mapping Rules

ER metamodel, shown in Figure 2, is based on ER model defined in [22]. *ERConcept* represents the most abstract concept in the ER data model. It is specialized using more concrete ER concepts:

- *Entity* represents types of objects in a system. It is further specialized into *Kernel*, *Subtype*, *Aggregation*, and *Weak* entity types.
- *Relationship* is defined between two entity types.
- *Mapping* represents relationship roles as well as special relationships between specific entity types. *Min* and *Max* attributes specify lower and upper bound of its cardinality. *Mapping* is further specialized into more concrete subtypes: *OrdinaryMapping* (i.e. relationship role), *WeakMapping*, *AggregationMapping*, and *SpecializationMapping*.
- *Attribute* describes an entity type and *Domain* specifies the type value for an attribute.

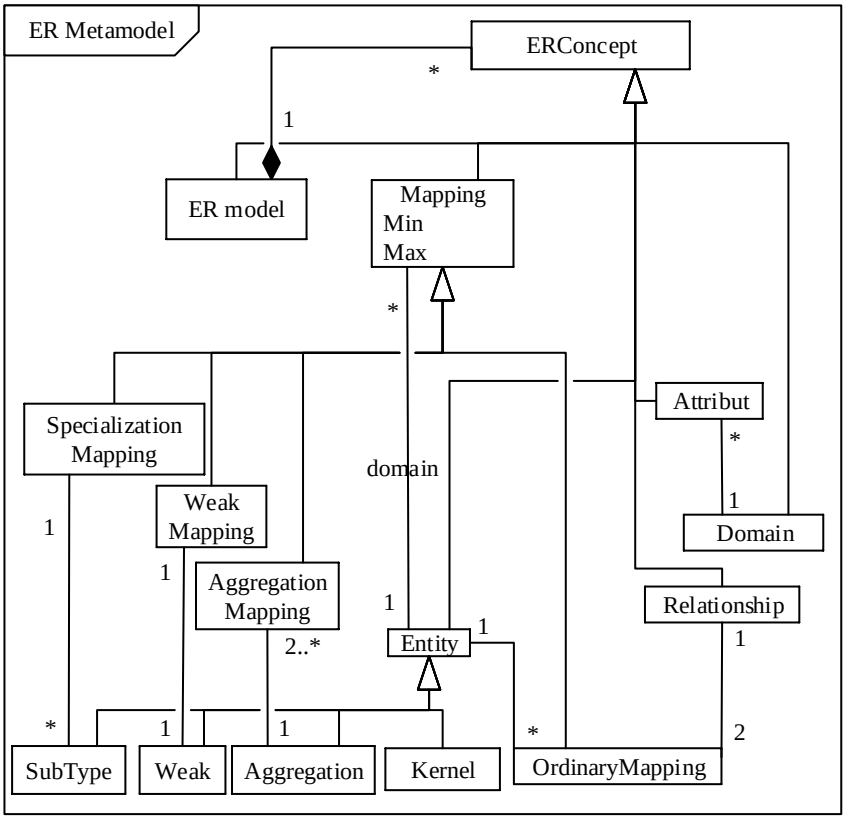

Fig. 2. ER metamodel

The version of FM we adopted in our approach is an enhanced version with cardinalities and attributes [7]. This version has an enhanced semantic expressiveness since it allows specification of constraints using cardinalities and supports also cloning of features (solitary subfeatures with MaxCardinality > 1). Additionally, the usage of attributes, instead of solitary subfeatures with parameters which is a semantically equivalent alternative, leads to more compact FM models. Thus, more efficient implementation is enabled.

The FM metamodel is shown in Figure 3. It is an original version developed by the authors of this paper independently of other FM metamodels available in the literature [7]. The main reason for developing our own metamodel version is to allow easier correspondences between concepts of ER models and FM, especially between ER mappings and FM relationships.

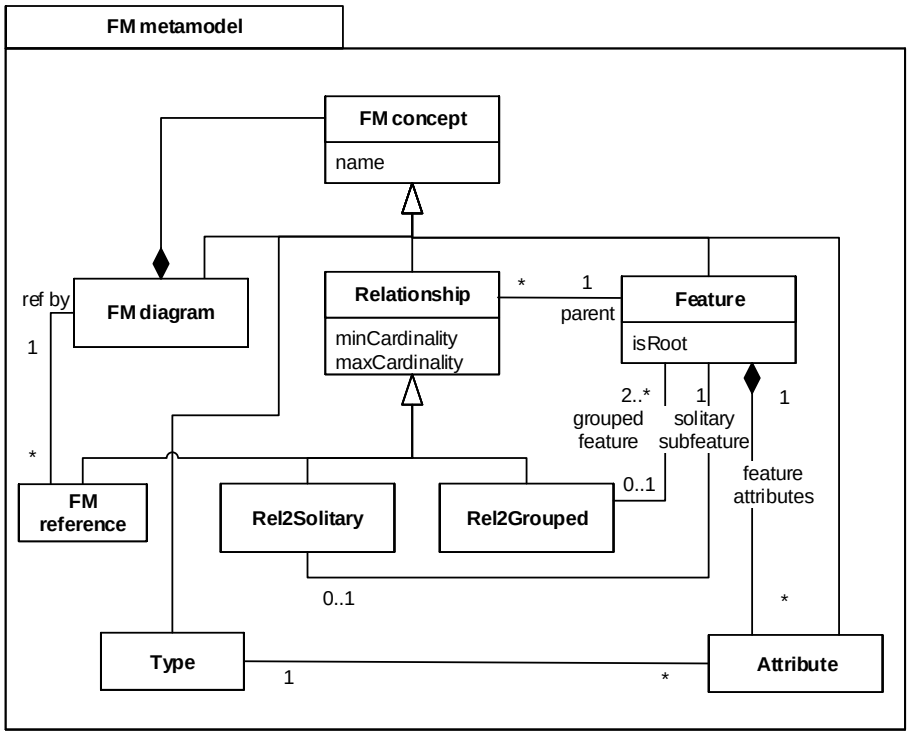

Fig. 3. FM metamodel

Its most abstract concept is FMConcept which is further specialized into more concrete concepts:

- *Feature diagram* represents a FM model which encompasses all other FM concepts belonging to it.
- *Feature* is a central FM concept. Its property *isRoot* defines a feature which is the root of a hierarchy of features comprising the given FM model. In a FM model only one feature can have this property set to true.
- *Relationship* represents a parent/child association between features forming a hierarchy of features. A relationship is subtyped to *Rel2Solitary* subclass,

representing an association between a parent feature and its solitary subfeature, and to *Rel2Grouped* subclass, representing an association between a parent feature and a group of features (also called grouped features). Properties *MinCardinality* and *MaxCardinality* enable expression of various constraints on relationships between features. In case of *Rel2Solitary* relationship, these properties determine whether a solitary feature is mandatory (value of *MinCardinality* is 1) as well as whether multiple features are allowed (*MaxCardinality* is higher than 1). In case of *Rel2Grouped* relationship, these properties determine whether a feature group is mandatory as well as exclusive (*MaxCardinality* is 1) or not (*MaxCardinality* is higher than 1).

- *FM reference* represents a reference to another FM enabling a division of large feature models into smaller ones. It is modeled as subtype of *Relationship*, thus enabling modelers to represent a subtree of features of some parent feature as a separate FM model.
- *Attribute* is defined as a separate FM concept due to efficiency, although it could be in essence represented as a solitary subfeature with a parameter.

The mapping between ER and FM concepts is determined by the following rules:

- **ER2FM** rule: Each ER model maps to a Feature model.
- **Entity-Feature** rule: Each *Entity* type (*Kernel*, *Subtype*, *Aggregation*, and *Weak*) maps to a Feature.
- **Mapp-Solitary** rule: Each *WeakMapping*, *AggregationMapping* and *OrdinaryMapping* maps to a *Rel2Solitary* relationship. Cardinalities of mappings between entities become cardinalities of the relationships linking corresponding features.
- **Spec-Group** rule: Each *SpecializationMapping* maps to a *Rel2Grouped* relationship. All entity subtypes of *SpecializationMapping* are mapped to features which are connected to the given *Rel2Grouped* relationship. Cardinalities of *SpecializationMapping* become cardinalities *Rel2Grouped* relationship
- **Ord-FMref** rule: If two entities have two or more relationships or if three or more entities form a circle then *FM reference* is created in order to maintain FM as a strict hierarchy (tree) of features. A *FM reference* should also be created in case of relationships linking an entity to itself (i.e. recursive relationships) for the same reason.
- **Attr-Attr** rule: Each *Attribute* maps to a feature *Attribute*.
- **Domain-Type** rule: Each *Domain* maps to a *Type*.

The Mapping metamodel, shown in Figure 4, defines allowed correspondences between ER concepts and FM concepts based on the aforementioned rules. A mapping between two concrete ER and FM models is represented by *ER2FM* class, which encompasses all correspondences between concrete elements of the ER and FM models made using the mapping rules. Class *Element* represents an instance of such correspondences. For each mapping rule there are appropriate subclasses of *Element*, which are named after the rule.

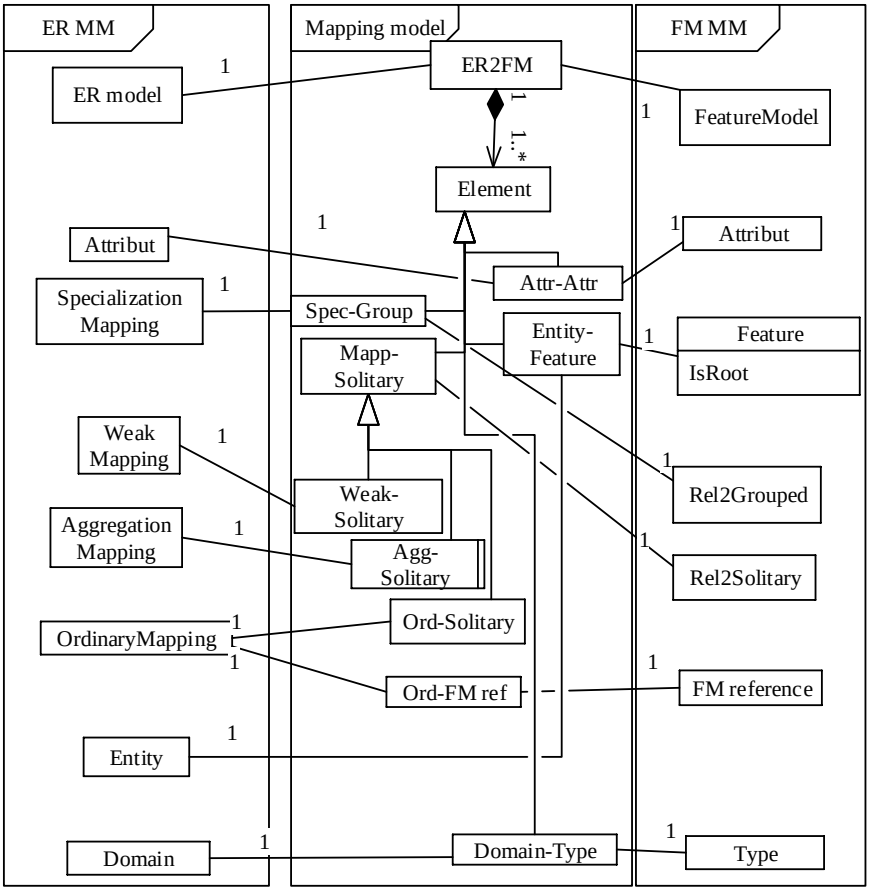

Fig. 4. Mapping metamodel

As it is obvious from the metamodel, mappings between concepts of ER model and FM are unique (non-ambiguous), except for OrdinaryMapping (ER relationship role). OrdinaryMapping can be mapped by *Mapp-Solitary* rule to a Rel2Solitary relationship, linking a feature and its solitary subfeature, or can be mapped to FM reference (a link to another FM model) by *Ord-FM-ref* rule. However, in case of recursive or circular ER relationships modelers are forced to use *Ord-FM-ref* rule instead of using *Mapp-Solitary* rule. So, the non-ambiguity of mapping ER concepts to FM concepts is also preserved in this case. In other cases developers can freely choose between these two rules, i.e. to choose *Ord-FM-ref* rule instead of *Mapp-Solitary* rule when they want to break a large FM model into smaller ones.

3.3. An Example

In this section we illustrate our approach by an example of a CASAS aimed to support a consortium of flower stores in a big city. The consortium has made an agreement with local taxi drivers to deliver flowers from the stores to their customers. When a store gets a flower delivery order from a customer, it creates a request which is sent to drivers from the store in order to select a driver assigned for the actual delivery. Drivers compete for the delivery by sending their current location. Depending of preferences of stores (e.g. automatic or manual delivery assignments to drivers, delivery confirmation required or not, etc.) and equipment options for drivers (e.g. whether the driver is equipped with a GPS mobile device), there can be many different variants of applications supporting stores and drivers. Here we give three use cases (UC) with contextual variants which are expressed using #if metastatements:

1. **Use case:** *Select driver for delivery*
   Actor: *Florist*
   - The system sends an offer for delivery to all drivers
   - The system registers the positive responses and the location of the driver
   - The system ranks all driver responses based on current driver distances from the store

   #if *FloristContext.AutoAssign* // Variant 1: Automatic Assignment
   - The system selects the driver with the best rank.

   #else // Variant 2: Manual assignment
   - Florist manually selects the driver from the driver ranking list.

   #endif

2. **Use case:** *Send bid for delivery*
   Actor: *Driver*
   - The driver receives a request for a delivery
   - The driver accepts the offer

   #if *DriverContext.Device.Mobile.GPS* // *Variant 1:* driver with GPS device
   - The system gets the current location from the driver's GPS device

   #else // Variant 2: driver is without GPS device
   - Driver enters and sends manually current location

   #endif

3. **Use case: Confirming delivery**
   Actor: *Driver and Customer*
   - **#if** *FloristContext.DeliveryConfirmation*

   // Variant 1: Store requires delivery confirmation
   - The driver asks the client to enter the confirmation code given to him by the store
   - Confirmation code is sent to the store

   #else // Variant 2: Store doesn't require confirmation
   - Use case does not exist for this driver.

   #endif

In order to do required adaptations, our CASAS system must keep the contextual information about stores and drivers. The (simplified) global context model is given in Figure 5. It includes information about stores, their preferences on task assignment and delivery confirmation as well as information about drivers working for a particular store. Drivers can be equipped with different devices, which can be mobile phones (with or without GPS) or GPS navigation devices. Additionally, each driver can have as his/her replacement another driver to whom requests are forwarded in case the initial driver is out of work.

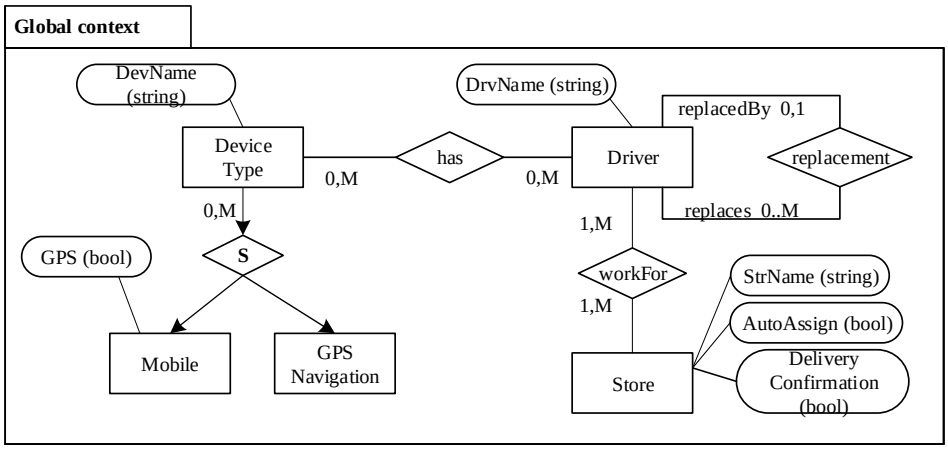

Fig. 5. An example of global context model

Since stores and drivers have two independent applications for realizing appropriate use cases, each application must have its own local context model. The two local context models include only relevant contextual information projected from the global context. These local contexts are shown in Figure 6. In addition to store's preferences, the FM for the local context of the store includes a list of drivers who work for this particular store (represented as solitary subfeature with cardinality 1..M). On the other hand, the FM for the local context of driver includes a list of devices a driver possesses (solitary subfeature with cardinality 1..M). Each device may have aditional subfeatures acorcoding to its type (represented as XOR grouped feature). Since each driver's replacement is also a driver with its own context, this situation is in the FM represented as a recursive FM reference to itself.

A part of the corresponding mapping model between the global context model and the local driver's context models of driver is given in the object diagram in Fig 7. ER_schema package contains instances of global context model from Fig 5, while package FM contains instances of FM Driver_Context from Fig 6. Package Mapping_model contains instances of the corresponding mapping rules by which concepts of the global context are mapped to the concept of the local context. Thus, features f1 (Driver) and f2 (DeviceType) are mapped from corresponding entities of the same name using Entity-Feature rule. On the other hand, relationship r2, which defines that f2 is a solitary subfetaure of f1, is mapped from o1 ordinary mapping (hasDeviceType) by rule OrdSolitary. FM reference fm1 (replacedBy) is mapped from ordinary mapping o2 using Ord-FM-ref rule.

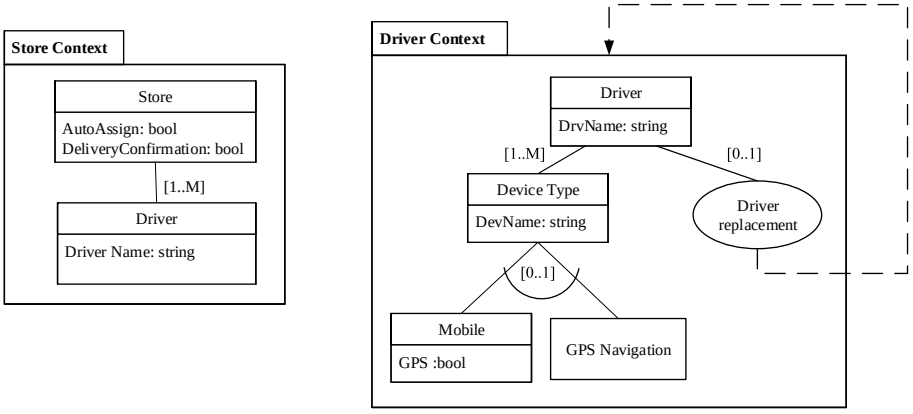

Fig. 6. FM Local contexts for stores and drivers

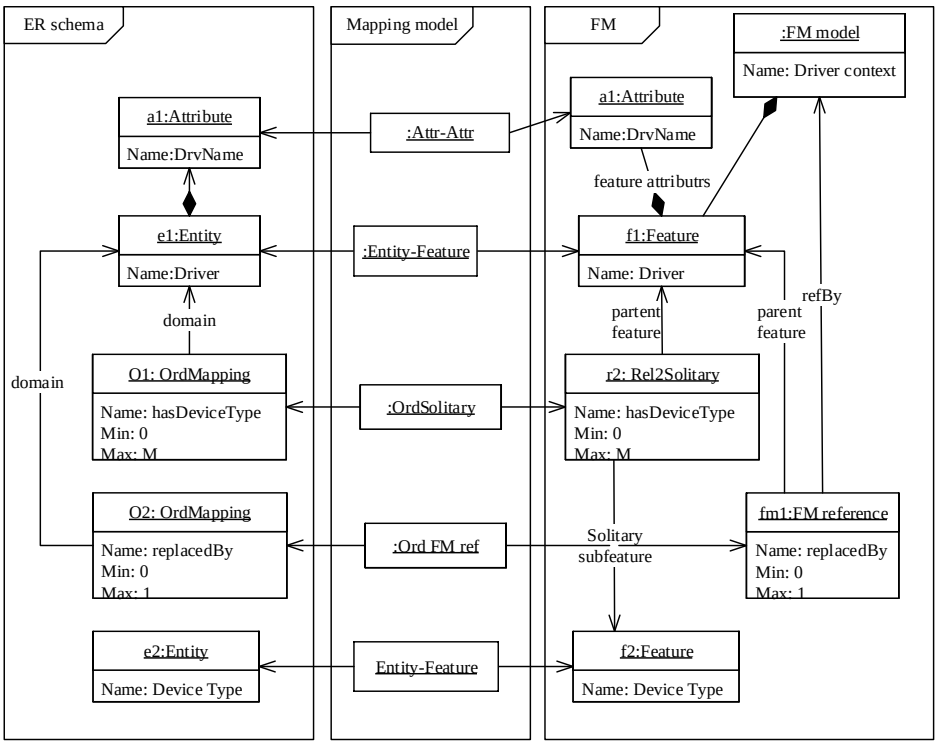

Fig. 7. A part of mapping model between global and local context

4. Implementation aspects

Since it is based on model driven engineering, our approach allows different implementation architectures and platforms. Here we describe one possible implementation used in our prototype system that is currently under development (shown in Figure 8).

For the realization of global context state we have chosen a relational database. We consider that database technology, including latest memory based RDBMS targeting mobile platforms, is superior in IoT environment where a large number of applications (sensors and other "smart" objects) concurrently access the global context state.

Regarding the realization of the local context state, we decided to use JSON objects [20] as a very efficient implementation of FM configurations. JSON format is used in many web platforms as an efficient alternative to XML format for transferring data between distributed applications. Additionally, JSON objects are easily transformed to native objects in many programming languages (Java, PHP, JavaScript, etc.).

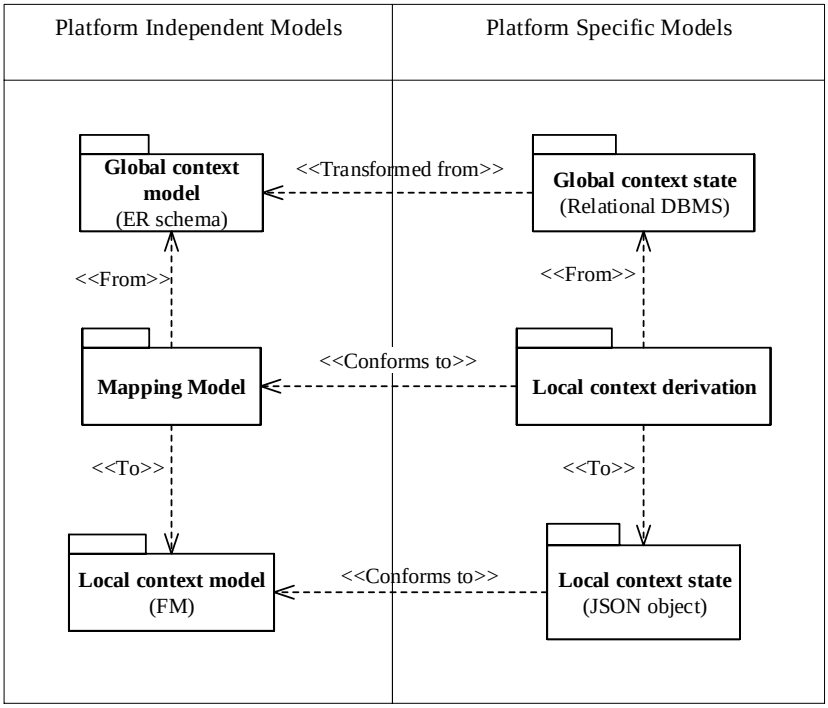

Fig. 8. Implementation architecture of prototype system

JSON objects are created by Adaptation Manager through the process called Local context derivation. Local context derivation is done according to Mapping Model by making appropriate queries over global context state (i.e. relational database) in order to create a JSON representation of local context state.

In order to adapt applications at runtime, our current prototype version of Adaptation Manager, which is built on the principles of DSPL, utilizes template engines as a technique to render application source code based on templates and JSON objects representing the local context state.

5. Conclusions

The main advantage of our approach stems from the utilization of both global and local contexts modeled by two different modeling techniques. Ontologies are superior for context modeling and realization of global context state, but not so suitable for the adaptation purposes in DSPL. On the other hand, feature models are suitable for the adaptation purposes, but no so adequate for global context modeling. Thus, our approach takes the best of both ontologies and feature models by using synergy effects.

Comparing to other existing approaches, the key benefit of our approach is in the adaptation process. It can be much more efficient due to smaller, less complex and better tailored local context models. This efficiency is achieved without sacrificing the advantages of ontologies. Thanks to the view-based approach, context can be easily shared between different applications increasing reuse of context information and reducing their complexity. Separating local contexts tailored for each particular application and considering only the relevant context will keep integrity of context information with no redundancy, reduce response time and increase the reasoning performance which is a critical concern in IoT environment.

An additional level of efficiency is also achieved by applying model driven development. It allows developers to use high level modeling techniques in the design time, while for implementation technology platforms can be used which are the most efficient in the given circumstances. Our initial experience with our first prototype implementation is very encouraging.

While some of the issues have been resolved, much work is yet to be done. In future research we would like to thoroughly verify benefits in efficiency of our approach over traditional approaches for context modeling and implementation in a large scale IoT environment. Future work also includes developing a full feature DSPL for a context-aware self-adaptive system based on our approach. We plan to generate logic and source code of Adaptation Manager such as local context derivation. We also plan to extend our developer prototype with complex event processing in order to better support and improve efficiency of reasoning and run time adaptations.

References

1. Acher, M., Collet, P., Fleurey, F., Lahire, P., Moisan, S., Rigault, J.P.: Modeling Context and Dynamic Adaptations with Feature Models. In Int'l Workshop Models@run.time at Models 2009 (MRT'09). (October, 2009)
2. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 263–277 (June 2007)

3. Bellavista, P., Corradi, A., Fanelli, M., Foschini, L.: A survey of context data distribution for mobile ubiquitous systems, ACM Computing Surveys (CSUR), v.44 n.4, p.1-45, (August 2012)
4. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modeling and reasoning techniques. Pervasive and Mobile Computing, vol.6 no.2, p.161-180, (April, 2010)
5. Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F. A., Tanca, L.: A data-oriented survey of context models. ACM SIGMOD Record, Vol. 36, No. 4, (December 2007)
6. Czarnecki, K., Chang Hwan, P.K., Kalleberg, K.T.: Feature Models are Views on Ontologies. Proceedings of the 10th International on Software Product Line Conference, p.41-51, August 21-24, (2006)
7. Czarnecki, K., Helsen, S., Eisenecker, U.: Staged Configuration Using Feature Models, R.L. Nord (Ed.): SPLC 2004, LNCS 3154, pp. 266-283 (2004)
8. Devedžić, V.: Understanding ontological engineering. Communications of the ACM, vol.45 no.4, (April 2002)
9. Fernandes, P., Werner, C., Teixeira, E.: An Approach for Feature Modeling of Context-Aware Software Product Line. Journal of Universal Computer Science, Special Issue on Software Components, Architectures and Reuse, vol. 17, no. 5, pp.807-829, (2010)
10. France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. Future of Software Engineering. IEEE Computer Society, (2007)
11. Geihs et all, K.: Software engineering for self-adaptive systems. Springer-Verlag, Berlin, Heidelberg, Chapter Modeling of Context-Aware Self-Adaptive Applications in Ubiquitous and Service-Oriented Environments, (2009)
12. Gomaa, H., Hussein M.: Dynamic software reconfiguration in software product families. In: van der Linden, F. (Ed.), Software Product Family Engineering. Lecture Notes in Computer Science 3014, Springer-Verlag, Berlin, Heidelberg. pp. 435-444, (2004)
13. Hallsteinsen, S., Geihs, K., Paspallis, N., Eliassen, F., Horn, G., Lorenzo, J., Mamelli, A., Papadopoulos, G.A.: A development framework and methodology for self-adapting applications in ubiquitous computing environments. Journal of Systems and Software, v.85 n.12, p.2840-2859, (December 2012)
14. Hallsteinsen, S., Hinchey, M., Park, S., Schmid, K.: Dynamic Software Product Lines. Computer, Vol. 41, No. 4, 93–95. (2008)
15. Hallsteinsen, S., Stav, E., Solberg, A., Floch, J.: Using Product Line Techniques to Build Adaptive Systems. In Proc. Int'l. Software Product Line Conf. (SPLC), pages 141–150. IEEE CS, (2006)
16. Hartmann, H., Trew, T.: Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains. In 12th International Software Product Line Conference, IEEE (2008), pp. 12-21, Ireland, (September 2008)
17. Jaroucheh, Z., Liu, X., Smith S.: CANDEL: Product Line Based Dynamic Context Management for Pervasive Applications. International Conference on Complex, Intelligent and Software Intensive Systems (ARES/CISIS 2010), IEEE CS, 209-216. (2010)
18. Jaroucheh, Z., Liu, X., Smith, S.: Mapping features to context information: Supporting context variability for context-aware pervasive applications. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on (Vol. 1, pp. 611-614). IEEE, (Avgust 2010)
19. Jarrar, M., Demey, J., Meersman, R.: On Using Conceptual Data Modeling for Ontology Engineering. In Spaccapietra, S., March, S., Aberer, K., (Eds.): Journal on Data Semantics (Special issue on Best papers from the ER, ODBASE, and COOPIS 2002 Conferences). LNCS, Vol. 2800, Springer, pp.:185-207. (October 2003)
20. JSON Tutorial , http://www.w3schools.com/json/.
21. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Software Engineering Institute, Carnegie Mellon University,Tech. Rep. CMU/SEI-90-TR-21, (Novembar 1990)

22. Lazarević, B., Marjanović., Z., Aničić, N., Babarogić, S.: Introduction to Databases, Beograd, (2006)
23. Lee, J., Kang, K.C.: A Feature-Oriented Approach to Developing Dynamically Reconfigurable Products in Product Line Engineering. In Proc. Int'l. Software Product Line Conf. (SPLC), pages 131–140. IEEE CS, (2006)
24. Lee, K., Kang, K.C.: Usage context as key driver for feature selection. In J. Bosch and J. Lee, editors, Software Product Lines: Going Beyond, volume 6287 of Lecture Notes in Computer Science, pages 32-46. Springer Berlin Heidelberg, (2010)
25. Morin, B., Barais, O., Jézéquel, J-M., Fleurey, F., Solberg, A.: Models@ run. time to support dynamic adaptation. Computer 42, no. 10: 44-51, (2009)
26. Northrop, L. M.: SEI's Software Product Line Tenets. IEEE Software, v.19, n.4, p.32-40, (July 2002)
27. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the Internet of Things: A survey, IEEE Commun. Surveys Tuts., vol. PP, no. 99, pp.1 -41 (2013)
28. Rajiv Kishore, K., Hong, Z., Ramesh, R.: A Helix-Spindle model for ontological engineering. Communications of the ACM, vol.47 no.2, p.69-75, (February 2004)
29. Raptis, D., Tselios, N., Avouris, N.: Context-based design of mobile applications for museums: a survey of existing practices. In Proc. 7th Int. Conf. on Human computer interaction with mobile devices & services, pages 153–160, (2005)
30. Saller, K., Lochau, M., Reimund, I.: Context-aware dspls: model based runtime adaptation for resource-constrained systems. In SPLC'13, pages 106-113. ACM, (2013)
31. Sanchez, D.M., Cavero, J.M., Martinez, E.M.: The road toward ontologies. In ONTOLOGIES : A handbook of priciples, concepts and applications in information systems, R. Sharman, R. Kishore, and R. Ramesh, Eds. London: Springer, pp. 3-20, (2006)
32. Schmidt, A.: Implicit human-computer interaction through context. 2nd Workshop on Human Computer. Interaction with Mobile Devices. (1999)
33. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In 1st Int. Workshop on Advanced Context Modelling, Reasoning and Management, (2004)
34. Vanathi, B., Uthariaraj, V.R.: Collaborative Context Management and Selection in Context Aware Computing. In Communications in Computer and Information Science, 1, Vol. 133, Advanced Computing, Part 4, Springer-Verlag Berlin Heidelberg, pp: 348-357. (2011)
35. Vanathi, B., Uthariaraj, V.R.: Hybrid hierarchical context representation in a context aware system. In Proc. of the 2nd International Conference on IT and Business Intelligence (ITBI'10), IEEE and IEEE Computational Intelligence Society, Nagpur, (2010)

Siniša Nešković is a lecturer of information systems at the University of Belgrade, Faculty of Organizational Sciences. He is the leader of the Laboratory for Information Systems, a research group working on the development of software tools, frameworks and components used for building complex information systems. His research interests include information systems development, business process modeling and automation, model driven development, software product line engineering, advanced software architectures, information retrieval and integration on the Web.

Rade Matić is the head of Informatics and E-business Department at the Belgrade Business School, a higher education institution for applied studies. He currently teaches courses in Design of Information Systems and Management Information Systems at BSc level. His research interests include information systems development, business process modeling and analysis, context modeling, software product line engineering and business intelligence.

Received: October 31, 2014; Accepted: April 17, 2015.