A Framework to Support the Development and Evolution of Self-adaptive Data-intensive Systems

Marco Mori and Anthony Cleve
PReCISE Research Center, University of Namur
Email:{mmo,acl}@info.fundp.ac.be

***Abstract*—Nowadays ubiquitous software systems have to meet user expectations while considering an ever-changing environment. The increasing space of possible contexts and the limited capacity of mobile devices make no longer possible to incorporate all necessary software alternatives and the required data for all possible contexts. Thus, upon variations to user task, role, preferences or physical environment, the current software alternative and the required data have to be reconfigured.**

This talk supports the variation of the current software alternative and data by defining the mapping between user requirements and database excerpts. We support the creation of the most suitable, consistent and sufficient excerpt of data taking into account the current context and its possible future changes. The aim of this research is to create a theoretical and practical framework that supports the development and the evolution of adaptive data-intensive software systems for ubiquitous environments.

I. Setting the context

Nowadays software systems have to meet user expectations taking into account heterogeneous environments and changing user needs. In this context, the literature of *self-adaptive systems* [13], [3], [6] provides the theoretical and methodological support for managing the variability of system behavior as a consequence of context variations. At design-time, software engineers define different software alternatives that provide the means for satisfying user requirements in different contexts. At run-time the adaptivity is supported by switching to the most suitable software alternative based on the current context. Different software alternatives possibly need distinct partial portions of a global database which supports the whole set of user requirements. The more the application is adaptive, the more will be the number of its variants and the different portions of data each variant needs. In this scenario it is not feasible to consider a unique global database for all possible alternatives for different reasons. First, a unique database is more difficult to understand and to evolve than a subset of it. Second, a single big database for all the possible contexts can not fit the limited space capacity of mobile devices on which the software alternatives will run.

To address these problems we have to align variability of requirements to the variability of databases by defining traceability links between software functionalities and databases excerpts. Variability of data has been considered in the literature of *context-aware databases* [1], where the most suitable portion of a database is provided to the application considering the current context, user tasks and preferences [7], [4]. Methodologies, techniques and tools have been defined following either a pruning [18], [16], [5] or a merging [2], [11], [14] technique for creating a subset of a database. However, there is still a gap between variability of data and variability of requirements [16]. On the one hand, variability of application requirements has not been considered in the literature of *context-aware databases*, while on the other hand the literature of *self-adaptive systems* provides no support for propagating variations of requirements to data. At the intersection of the two research areas, our research goal is to create a framework that supports the development and the evolution of new *self-adaptive data-intensive systems* and the migration of non self-adaptive data-intensive systems for ubiquitous environments.

We focus on data and in particular on the problem of finding the most suitable portion of data to support the current set of requirements that have to be provided by the application in the current context.

II. Framework Basics

Users of ubiquitous applications use mobile devices with limited memory and computation capacity while they are interested in continuously changing excerpts of data based on different factors, namely user tasks, user roles, device characteristics and physical environment. Since a global database that supports all users activities for all the possible contexts cannot be entirely loaded, we have to find the most suitable excerpt of data for the current software variant that implements the currently required user requirements. To address this problem we consider a chain of relationships among *context*, *software functionalities* and *data*. *Context* determines which is the set of *software functionalities* that have to be provided by the application; *software functionalities* implement a subset of user requirements and they require a certain sub-set of *data*. Starting from the current context we should determine a *sufficient* and *consistent* sub-set of the global database: *sufficient* in terms of data required in the current context and *consistent* with respect to the global database constraints.

Our framework supports the context-dependent variability of data together with the application variability following a feature engineering perspective [15], [5]. We represent each application functionality as a single unit of behavior, called *feature*, which contains a business requirement, a contextual condition to activate the feature and the required data, e.g., expressed as subsets of concepts belonging to the large database schema. Variability of the application and data is supported

by different configurations of features each corresponding to a consistent portion of the global database.

As far as data are concerned, we consider different levels of abstractions: the conceptual schema is the basis to define the data that are related to each single feature in terms of entity types, or concepts; the logical schema is derived from the conceptual schema and constitutes a first step to the deployment of data to the target platform; the physical schema describes indexes to achieve better performance in accessing data; finally, database instances are the data to be loaded into the device.

Our framework supports feature-based schema filtering by means of a *filtering design phase* and a *filtering process phase*. The *filtering design phase* supports the variability of accessing data by establishing the applicability for all the possible feature configurations. It starts by defining the whole set of requirements along with the corresponding global database for all possible contexts. Then, designers organize the elicited requirements following a feature engineering perspective, through *features* and a *feature model* which entails the admissible configurations. At this point designers define a mapping between identified features and portions of the global schema, and they identify the contextual dimensions that affect the interest towards different portion of data. Finally designers define a presence condition for each feature to evaluate if data required by a feature should be included or not in the subset of the database. At run-time, the *filtering process phase* provides the right data according to the features that have to be provided in the current context. Upon context variation an automatic derivation phase retrieves the most *suitable* set of features to apply. Starting from these features the corresponding data are filtered from the global database in order to produce a consistent view. The framework supports the reconfiguration process by applying variations to database instances, conceptual schema, logical schema and physical schema. This process may generate inconsistencies that appear within each model and between models. We avoid the first by applying proper adjustments phases and by following semantically defined construction rules. We solve inconsistencies between models by co-evolving them. To this end, we apply bi-directional transformations among models [17] and model driven engineering techniques.

In [8] we have define the basic elements of the framework and implemented the filtering process of the large conceptual schema by means of two different algorithms. This paper proposes a definition of sufficient and consistent sub-schema with respect to the global schema. In order to discover which is the most *suitable* configuration of features to apply, we are currently exploiting predictive task models containing information about data accesses with the aim of providing better performance (stability, responsiveness, etc...) to the reconfiguration process (e.g., [10], [9]). To this end, we have formalized a multi-objective optimization technique that exploits current and probable future information needs to assign a fitness value at each admissible configuration thus supporting the decision making process.

As for future work, we will implement the design and run-time process as they have been already defined in [8] and we will introduce a notion of *uncertainty* within our artifacts of interest in order to address possible variations that cannot be predicted at design-time. To this end, we will support the addition/deletion of features (data) at run-time, the variation of the task model by enabling the addition/deletion of states and variations to their probability distributions, and finally we will also support variations to the set of context dimensions [12]. We will also apply consistency checks to the artifacts reconfigured at run-time in order to prevent databases inconsistencies such as the violation of integrity constraints and inconsistencies among database representations at different abstraction levels.

REFERENCES

[1] C. Bolchini, C. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca. And what can context do for data? *ACM*, 52(11):136–140, 2009.

[2] C. Bolchini, E. Quintarelli, and R. Rossato. Relational data tailoring through view composition. In *ER*, pages 149–164, 2007.

[3] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, editors. *Software Engineering for Self-Adaptive Systems*, volume 5525 of *LNCS*, 2009.

[4] P. Ciaccia and R. Torlone. Modeling the propagation of user preferences. In *ER*, pages 304–317, 2011.

[5] K. Czarnecki and M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In *GPCE*, pages 422–437, 2005.

[6] P. Inverardi and M. Mori. A software lifecycle process to support consistent evolutions. In *Self-Adaptive Systems*, volume 7475 of *LNCS*, pages 239–264, 2012.

[7] D. Martinenghi and R. Torlone. A logical approach to context-aware databases. In A. D'Atri, M. De Marco, A. M. Braccini, and F. Cabiddu, editors, *Management of the Interconnected World*, pages 211–219. Physica-Verlag HD, 2010.

[8] M. Mori and A. Cleve. Feature-based adaptation of database schemas. In *Proc. of the 8th Int. Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2012)*, LNCS. Springer, 2012. to appear.

[9] M. Mori, F. Li, C. Dorn, P. Inverardi, and S. Dustdar. Leveraging state-based user preferences in context-aware reconfigurations for self-adaptive systems. In *SEFM*, pages 286–301, 2011.

[10] C. Parra, D. Romero, S. Mosser, R. Rouvoy, L. Duchien, and L. Seinturier. Using constraint-based optimization and variability to support continuous self-adaptation. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 486–491, New York, NY, USA, 2012. ACM.

[11] C. A. Parra, A. Cleve, X. Blanc, and L. Duchien. Feature-based composition of software architectures. In *ECSA*, pages 230–245, 2010.

[12] E. Quintarelli, E. Rabosio, and L. Tanca. Context schema evolution in context-aware data management. In *ER*, pages 290–303, 2011.

[13] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *TAAS*, 4(2), 2009.

[14] G. Saval, J. P. Puissant, P. Heymans, and T. Mens. Some challenges of feature-based merging of class diagrams. In *VaMoS*, pages 127–136, 2009.

[15] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, 2007.

[16] N. Siegmund, C. Kästner, M. Rosenmüller, F. Heidenreich, S. Apel, and G. Saake. Bridging the gap between variability in client application and database schema. In *BTW*, pages 297–306, 2009.

[17] J. F. Terwilliger, A. Cleve, and C. Curino. How clean is your sandbox? - towards a unified theoretical framework for incremental bidirectional transformations. In *ICMT*, pages 1–23, 2012.

[18] A. Villegas and A. Olivé. A method for filtering large conceptual schemas. In *ER*, pages 247–260, 2010.