Towards highly adaptive data-intensive systems

Mori, Marco; Cleve, Anthony

Published in:
Lecture Notes in Business Information Processing

DOI:
10.1007/978-3-642-38490-5_36

Publication date:
2013

Document Version
Peer reviewed version

Link to publication

Citation for pulished version (HARVARD):
Mori, M & Cleve, A 2013, Towards highly adaptive data-intensive systems: A research agenda. in *Lecture Notes in Business Information Processing: CAiSE 2013 International Workshops.* vol. 148 LNBIP, Lecture Notes in Business Information Processing, vol. 148 LNBIP, Springer Verlag, pp. 386-401, 25th Conference on Advanced Information Systems Engineering, CAiSE 2013, Valencia, Spain, 17/06/13. https://doi.org/10.1007/978-3-642-38490-5_36

Towards Highly Adaptive Data-intensive Systems: A Research Agenda

Marco Mori[*] and Anthony Cleve

PReCISE Research Center, University of Namur
Rue Grandgagnage 21, 5000 Namur, Belgium
`{marco.mori,anthony.cleve}@unamur.be`

Abstract. Data-intensive software systems work in different contexts for different users with the aim of supporting heterogeneous tasks in heterogeneous environments. Most of the operations carried out by data-intensive systems are interactions with data. Managing these complex systems means focusing the attention to the huge amount of data that have to be managed despite limited capacity devices where data are accessed. This rises the need of introducing adaptivity in accessing data as the key element for data-intensive systems to become reality. Currently, these systems are not supported during their lifecycle by a complete process starting from design to implementation and execution while taking into account the variability of accessing data. In this paper, we introduce the notion of data-intensive self-adaptive (DISA) systems as data-intensive systems able to perform context-dependent data accesses. We define a classification framework for adaptation and we identify the key challenges for managing the complete lifecycle of DISA systems. For each problem we envisage a possible solution and we present the technological support for an integrated implementation.

Key words: data-intensive systems lifecycle, context-aware database, self-adaptive systems

1 Introduction

Data-intensive systems manage complex and huge amount of data that are suited for different types of users each performing tasks of different nature and possibly in different contexts. Most of the effort for designing, maintaining and evolving these systems depends on their complex interactions with big data sources. Taking this perspective, a relevant problem that need to be tackled in a systematic manner is how to ease the management of their variability in accessing data. This problem has started to be tacked in the literature of *context-aware databases* [1] by means of methodologies, techniques and tools for creating sub-portions of a global database based on different factors, i.e., current context, user tasks and user preferences [22, 7]. These techniques support variations of data that are not

[*] beneficiary of an FSR Incoming Post-doctoral Fellowship of the *Académie universitaire 'Louvain'*, co-funded by the Marie Curie Actions of the European Commission

performed in a systematic process where context-dependent variations of application behavior are propagated to data thus making it difficult to consistently change data.

In order to generate the data of interest at a certain context we should reason on the requirements that should be achieved in that specific context [38]. Through a systematic approach it should be possible to create context-dependent versions of data-related artifacts that will remain un-changed for the whole system lifetime. This first level of data-adaptivity (design-time), represents a possible solution which is suited for all the situations where there is no need of reconfiguring data and programs at run-time. Nevertheless, this solution becomes unappropriate in ubiquitous environments that are characterized by every-changing contexts leading to a two-fold problem. First, the set of contexts can be too large in terms of tasks that will be completed, different types of users accessing the same system and different situations in which the system will have to operate. Second, limited capacity devices may not be able to manage the needed big data source. Thus a second level of adaptivity (run-time) is needed for achieving reconfigurations of data-related artifacts in a systematic and continuous manner. In the literature of *self-adaptive systems* [34, 5, 17] different techniques support context-dependent behavioral adaptations both at design time and at run-time while not much attention has been devoted to data-manipulation adaptations. Data-intensive systems can benefit from these techniques to achieve design-time and run-time adaptations of data-manipulation programs while they can benefit from *context-aware database* approaches for propagating these variations to data-related artifacts.

Variability of accessing data poses the interest towards a new class of *data-intensive self-adaptive (DISA) systems* as systems able to ease the complexity of accessing data by means of creating context-dependent data-related artifacts at design time and by means of enabling their run-time reconfigurations. In this paper, we envisage a lifecycle process for DISA systems and we propose a research agenda organized according to the three main challenges that need to be addressed in order for these systems to become a reality; we consider a *design* process for DISA systems, *migration* of existing systems towards DISA systems and *monitoring/optimizing* run-time reconfigurations of DISA systems. In [24] we have defined a theoretical framework for supporting the design and configuration of new DISA systems. In this paper, we extend our previous results in a wider scope and we analyze the key challenges for DISA systems along with our methodological solutions for each of those. Beyond the creation of new systems, it is also important to consider legacy data-intensive systems. These systems need to be analyzed in order to evaluate their variability in accessing data. To understand this variability, it is necessary to extract their data-manipulation behavior by analyzing the system execution. Thus we consider the problem of process understanding in order to evaluate the convenience of migrating towards DISA systems. Finally, a migrated or a new DISA system have to be provided with a decision-making process to support its optimized run-time reconfiguration of programs and data-related artifacts.

We consider an e-health scenario where physicians, i.e, doctors, secretaries, nurses, radiologists and patients are involved in a set of care processes. Each physician is interested in a different excerpt of data. Secretaries are interested in administrative data, doctors are interested in case histories (with medical images) of patients, diagnosis and therapies, nurses and patients are interested in the application of the therapy while radiologists are strictly interested in case histories and basic patient information with the aim of capturing medical images. Further, there exist other factors that affect the portion of data of interest, namely the task the physician is performing, the device, the location or room where the system will run and the department to which the patient belongs. For instance, doctors access to high quality medical images only through desktop devices. They access to different set of information based on the activity they perform, i.e., check-up, visits, surgery operations and department administrator. Finally, if the doctor performs a visit from outside the hospital or in case of emergency, he should only visualize a textual representation of the case history. This scenario shows the context-dependent interest of users towards heterogeneous data. The application can be provided with the required data once for all the system lifetime or it may be necessary to reconfigure at run-time the data due to context variations.

In the remainder of this paper, Section 2 gives a detailed descriptions of DISA artifacts and a possible classification of data adaptations. Section 3 analyzes the three main challenges for supporting the lifecycle of DISA systems along with an integrated set of techniques to be adopted for implementing the process. For each problem we present contributions in the literature and our methodological approach. Finally, Section 4 discusses related work before conclusions and future directions are given in Section 5.

2 Framework basics

In this paper we adopt a feature engineering perspective [19, 37, 11] in order to represent the basic unit of behavior of DISA systems as *features*. In these systems most of the functionalities operates on data, thus we consider their corresponding features for our analysis. Section 2.1 discusses artifacts of a DISA system while Section 2.2 classifies adaptations to data-related artifacts.

2.1 DISA artifacts

Fig. 1 envisages the relationships between the artifacts of DISA systems, namely *context*, *features* and *data*. *Context* is characterized by means of a set of dimensions determining the current user situations, namely, user role, user task, device characteristics, location etc... Context states determines the set of features, i.e., *configurations*, that have to included into the application. Configurations defined according to a *feature model* [37] require a subset of data belonging to a big data source suited for all possible contexts. We consider different levels of abstraction of data: the *conceptual schema* is a Platform Independent Model (PIM) typically represented through an ER diagram containing entity types and relationships among entity types; the *logical schema* is a Platform Specific Model (PSM) containing tables and foreign keys which are the basis for defining database queries; finally *database instances* are data to be loaded into the device.

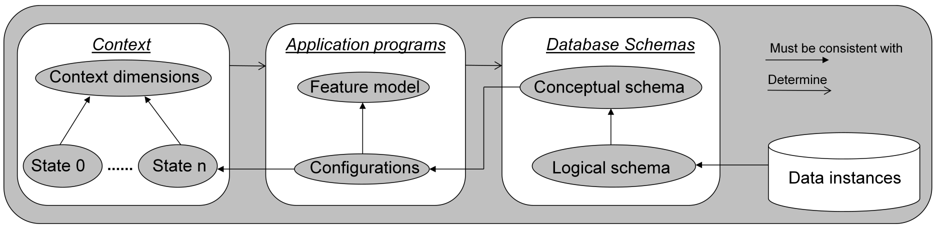

Fig. 1. DISA artifacts

In our perspective a *feature* links together variability of the application (in terms of its requirements) to the variability of data (in terms of data excerpts) as inspired by approaches where variability of requirements is linked to variability of a formal specification [8] or to the variability of source code [16]. Following the taxonomy proposed in [14] we distinguish among functional, performance and specific quality requirements which pertain to functional, performance and specific quality concerns of the application. Orthogonally to these requirements, constraints limit the solution space of functional, performance and specific quality requirements. Based on this taxomony we define a feature as triple $f = (R, P, V)$ where R is a functional, non-functional or a specific quality requirement (context independent), P is the presence condition, i.e. a contextual constraint requirement which expresses the applicability of the feature; V is the excerpt of data of interest for the feature defined in terms of entity types of the conceptual schema.

2.2 Data adaptations dimensions

In this section we present a classification of data adaptations based on which we characterize the adaptations we enable in our framework (Section 3.1). We define database adaptations according to the three following dimensions:

– *Structural dimension*, concerning modifications to database structures and data instances;
– *Semantic dimension*, i.e., evolution of the semantic of data;
– *Consistency dimension*, addressing the impact of adaptation to the correctness of data-related artifacts.

Structural dimension is concerned with the adaptation of the database structures. This regroups modifications applied to the conceptual, logical, physical schemas and data instances. we can classify database evolutions as:

- *Conceptual modifications* typically translate changes in the functional requirements of the information system into conceptual schema changes.
- *Logical modifications* do not modify the requirements but adapt their platform-dependent implementation in the logical schema.
- *Physical modifications* aim at adapting the physical schema to new or evolving technical requirements, like data access performance.
- *Data modifications* aim at translating variations of the schema to the database instance.

Semantic dimension captures the impact of a given database adaptation scenario on the informational content of the target database. In other words, it aims at indicating whether the adaptation involves: *Semantics-augmenting*, *Semantics-decreasing* or *Semantics-preserving* schema modifications.

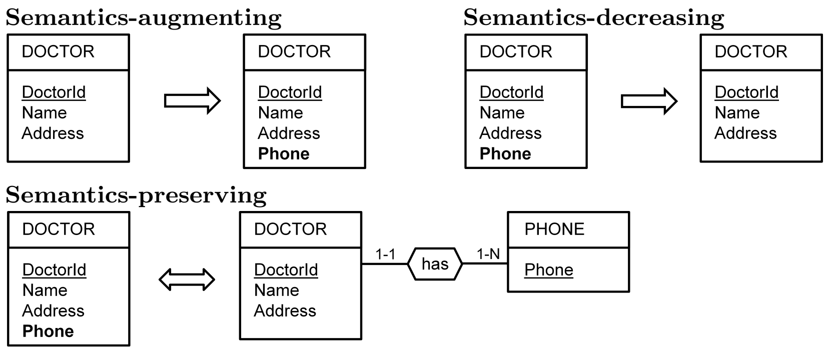

Fig. 2. Categories of schema modifications

Consistency dimension Whenever a variation to a data-related artifacts is performed we have to check the consistency of this variation considering two different problems:

- *intra-artifact* consistency: an adaptation may cause an existing consistency constraint to be violated within an artifact.
- *inter-artifact* consistency: a broken consistency link must then be reestablished by means of a *change propagation* adaptation. For instance, schema modifications at a given level of abstraction necessitate the adaptation of the schemas belonging to the other abstraction levels, and the data instances.

3 Variability for DISA systems

DISA systems have to be created either from zero or from legacy systems taking into account their variability in accessing data. In case of a new system it is necessary to consider which are the factors that tune the interest of the users towards different portions of data and how to propagate variations of this factors to variations of data. DISA systems should support design-time and run-time variability of data. The space of variability is determined at design time while during execution a certain configuration of data is created based on the current context. Only if context changes during the system lifetime we have to provide run-time variability in order to align data variations to context variations. These variations of data that are foreseen at design time are not enough if the system works in an un-predictable environment. Indeed there exist other variations to data that cannot be foreseen before they are needed. Indeed, in ubiquitous environments due to unforeseen situations it is not always possible to exactly determine the space of variability of a system before its execution. To this end, we envisage the adoption of a *design process phase* which is able to support *design-time variability*, *run-time variability* and *unforeseen run-time variability* of accessing data.

In case of the migration of a legacy system to a DISA system, we have to extract its data-manipulation behavior with the aim of understanding its variability in accessing data. This consists of a *process understanding phase* with the aim of evaluating if it is convenient or not to carry out the migration and determining the features of the system as the basis for its variability.

Finally, a DISA system has to perform run-time reconfigurations which are subject to performance degradations and scalability problems that need to be accurately tackled in order for a DISA system to be usable. In the remainder of this section we explain each key problem for a DISA system along with our envisaged solution: (i) *data-variability aware design process*, (ii) *data-variability aware process understanding*, (iii) *data-variability aware performance optimization.*

3.1 Data-variability aware design process

Context-dependent data accesses should be supported in a systematic manner during the design phase of a DISA system. To this end, it is necessary to align the 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 where many approaches identify the portions of data of interest based on a certain context model describing the current situation. Methodologies, techniques and tools have been defined following either a pruning [42, 43, 38, 36, 11] or a merging [2, 3, 22, 7, 35, 29] technique for creating a subset of a database. Altought it has been argued that it is important to link the variability of the application to variability of data [38], there is no approach that provides the data of interest based on changing application requirements. The literature of self-adaptive systems [34, 5, 17] provides the theoretical and methodological support for managing the variability of system requirements as a consequence of context variations. On the one hand, the literature of self-adaptive systems provides no support for propagating variations of requirements to data, while on the other hand variability of application requirements has not been considered in the literature of context-aware databases. To this end, taking inspiration by processes provided for self-adaptive systems [4, 17], we advice the adoption of a process for designing a DISA system, having as objective the variability of accessing data. We envisage the adoption of a framework that supports feature-based data tailoring by means of a *filtering design process* and a *run-time filtering process*.

The *design process* (Fig. 3) 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. Consequently designers organize the elicited requirements following a feature engineering perspective, through *features* and a *feature model* which entails the admissible configurations of features. 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. Finally, the *decision-making support* phase defines the applicability of each admissible configurations at each possible context state based on the presence condition of its entailed features.

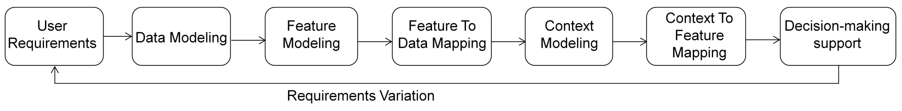

Fig. 3. Feature-based filtering design process

The *run-time filtering process* (Fig. 4(b)) provides the right data according to the features that have to be provided at the current context. Upon context variation an automatic derivation phase retrieves the most *suitable* (see Section 3.3) set of features to apply. According to these feature the process determines the set of entity types of the conceptual schema that should be included in the target view. The *data model validation* phase consists in modifying the conceptual schema in order to make it consistent with the large schema. Once this view has been created the *data model deployment* phase determines the corresponding logical schema and data instances. In Fig. 4(a) we have emphasized the stack for the reconfiguration of data. The target configuration of features is the input for determining the subset of the conceptual schema, which in turn is the input for determining the subset of the logical schema, which in turn determines the actual data. API's provided at each level support the propagation of variation from high-level conceptual schema till to data instances.

Designers should create DISA applications along with their context-dependent variability (Fig. 3) before putting the system in execution. At run-time based on the current context and user role, an automatic procedure should provide the subset of data of interest to the application. If context and user role remain the same for all the system lifetime we have only *design-time variability* of accessing data. On the contrary, if either user role or context change during the system lifetime, we need *run-time variability* of accessing data. In this case an automatic

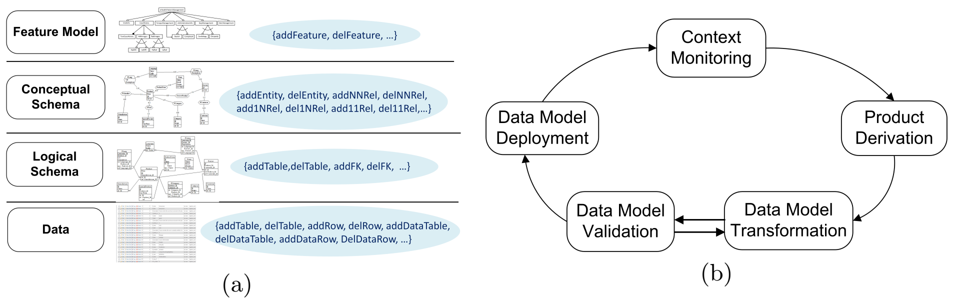
Feature Model
{addFeature, delFeature, ...}
Conceptual Schema
{addEntity, delEntity, addNNRel, delNNRel, add1NRel, del1NRel, add11Rel, del11Rel,...}
Logical Schema
{addTable,delTable, addFK, delFK, ...}
Data
{addTable, delTable, addRow, delRow, addDataTable, delDataTable, addDataRow, DelDataRow, ...}
(a)
Context Monitoring
Product Derivation
Data Model Transformation
Data Model Validation
Data Model Deployment
(b)

Fig. 4. Feature-based (a) reconfiguration stack and (b) run-time filtering process

procedure should enable the continuous run-time variations of data according to continuous context and user role variations (Fig. 4(b)).

This clearly distinction from design-time and run-time activities does not hold in ubiquitous environments which are mainly characterized by variations of requirements due to un-predictable contexts. In this case, the variability space determined at design time may have to be re-computed at run-time taking into account the variations to the requirement set to satisfy. The process we have envisaged in [24] is only able to support variations that are foreseen at design time. To this end, we enhance the process by enabling the *unforeseen run-time variability* of accessing data by means of re-iterating the steps of the process at run-time in order to satisfy a new set of requirements (Fig. 3). We consider only addition of requirements while we do not take into account deletion since the first poses the more difficult problems.

Feature evolution scenarios New requirements encapsulated into new features should be included into the data-intensive systems at run-time. We classify two co-evolution scenarios representing the addition of a new feature:

– Co-evolution between *requirements* and *context*: requirements of the application have to be provided based on the current context. On the one hand, whenever a new requirement has to be implemented into the system it may be necessary to modify the context. For instance, if the hospital buys a new machine for the radiologist department it may be necessary to add a new requirement to acquire a new kind of medical images. This implies the addition of a new applicability condition over the context to define when this requirement should be provided. Thus the context model has to be augmented with the new room where this new kind of images will be collected. On the other hand, whenever the dimensions of context are augmented, it may be necessary to modify the set of requirements. For instance if the set of user roles is augmented with the psychologist (context variation), we may have to consider a new requirement to support his activity.

– Co-evolution between *requirements* and *data*: requirements may require data in order to be fulfilled, thus they may co-evolve each other. Following the same example above, the new requirement to add for collecting new medical images requires new data portions where the images will be recorded. Thus data schemas and data instances have to be modified in order to include this new type of information. On the contrary, whenever data are evolved (e.g., augmented), new requirements may be required for accessing these data. Let us suppose to add to data the portion corresponding to the long hospital treatments. As a consequence we will have to add a new requirement to allow visualization of this new kind of treatments through the graphical unit interface (GUI).

Data adaptations in our framework In our framework we envisage the adoption of *conceptual*, *logical* and *data instances* modifications that are *semantics-decreasing* and *semantics-increasing*. In particular, in case of design-time variability we envisage the application of *semantics-decreasing* adaptations in order to produce a subset of information from the global source schema that are suited for a certain context. In case of run-time variability we envisage the adoption of *semantics-preserving* and *semantics-increasing* adaptations since switching from a context to another it is necessary to discard some information that are not anymore required while it is necessary to add other information. Both for design-time and run-time variability we envisage the adoption of *conceptual* modifications in order to align the platform-independent schema to the current set of requirements that have to be provided in a certain context. Consequently, we consider *logical* modifications in order to align the conceptual modification to the platform-dependent schema and finally *data instances* modifications for configuring the required database instance.

Data-related artifacts provided for a certain context are subsets of wider artifacts which include data necessary for all possible contexts. Whenever a variation to a data-related artifact is performed we have to check the correctness of this variation considering two different problems. First, we have to check if such variation is consistent with the rules defined for the definition of that specific model (*intra-artifact consistency*). Second, we have to check if the variation to the data-related artifact is consistent with its global source model (*inter-artifact consistency*). We achieve the first by performing variations that are correct by construction while we achieve the second by applying an adjustments phase to changed models [11]. Once a consistent model is obtained we apply bi-directional transformations [12, 39] in order to propagate variations from user-centric data models (i.e., the conceptual schema) till to data instances.

3.2 Data-variability aware process understanding

Extracting the data-manipulation behavior of the application is a complex task which supports designers in understanding a running data-intensive system. These systems carry out frequent interactions with a data source with the aim of fulfilling their requirements. Capturing these interactions is a promising approach for understanding the application behavior and for supporting three different types of activities: *discovering* the application behavior, *checking* the compliance of the application behavior w.r.t. contracts models, *enhancing* the application behavior. The *discovering* phase is required for legacy data-intensive systems for which documentation of processes is not available. Let us consider the e-health application implemented and working in hospital without documentation; for these application it is interesting to produce the behavioral models that are daily performed by physicians in order to document their actual activities. *Checking* the compliance of the application behavior is useful for both legacy and non-legacy data-intensive systems; let us consider an hospital manager which desires to monitor some parameters of the activities performed by third-party companies working at the hospital; managers agree on a written process and they would monitor that this process is correctly performed in reality. *Enhancing* the application behavior is another important activity for legacy and non-legacy data-intensive systems that allows the variation of the process by adding new instances of it, e.g., in a hospital the manager may want to add a new instance for the patient registration process where user is asked off-site to give a judgment of his experience at the hospital.

The activities we have described are at the core of process mining techniques whose aim is to discover, to check and to enhance real processes starting from event logs of information systems [41, 40]. Input events logs entail different instances of the same process as they can be recorded during system execution based on different types of information. Many approach presented in the literature [20, 31, 21, 33, 26] show the usefulness of process mining techniques for discovering, checking and enhancing processes in real working environments. The approach presented in [26] applies a process mining technique to logs of web service interactions, while approaches presented in [20, 31, 21] show the usefulness of process mining techniques in healthcare scenarios. Finally, the approach in [33] shows how to adopt a process mining technique to extract processes of wafer scanners to support testing held by a manufacturer. None of these approaches is designed for data-intensive systems. In order to adopt process mining techniques to data-intensive systems we have to determine higher-level data-manipulation functions starting from the interactions of the systems with its data sources. Approaches presented in the literature like [9, 10] support the extraction of semantic information starting from sequences of queries and relationships among those. Semantic information collected by following these approaches should be the basis for determining the events of a data-intensive application in terms of high-level data-manipulation functions. Upon the identification of these data-oriented events, we envisage the adoption of classical process mining techniques in order to produce data-oriented processes of the system.

Process mining for DISA systems can support the analysis of processes related to a single user or to a single group of users or it can support the analysis of multi-user of multi-group of users performing the same process. For example in the e-health scenario we may want to analysis the process of check-up visits for all the doctors and check the variability of accessing data for such a process. In addition we may want to analyze a process containing all the activities in which a single doctor is involved, i.e., check-up visits, surgery operations, department administrator activities, etc...

3.3 Data-variability aware performance optimization

DISA applications need to be reconfigured at run-time as a consequence of context-variations. Reconfigurations of data involve a set of conflicting requirements that should be carefully taken into account by the framework module implementing the variation of data. Among these requirements we have *stability* of data [18, 27], i.e., a measure defined as the ratio between variations of data (output) and variation of context (input), *user benefit*, i.e., a metric expressing the satisfaction of the user, and *reconfiguration cost* which is a metric defined over the operations that have to be completed for reconfiguring data. Let us consider the e-heath case study where the doctor changes his task from check-up activity to an emergency activity. This requires a reconfiguration of the database supporting the doctor mobile application in the emergency activity with a restricted set of data. This reconfiguration should be performed by considering the requirements above and in particular given more weight to the reconfiguration cost requirement since in an emergency situation it is better to have a low reconfiguration cost (e.g., very quickly) and low user benefit (e.g., limited patient information) instead of having high reconfiguration cost (e.g., wait too long) and high user benefit (e.g., rich set of patient information).

In Software Engineering conflicting criteria are combined together with different weights in a unique utility function that is optimized [44, 32, 28]. Approaches presented in the literature of self-adaptive systems show that these optimization approaches can benefit from predictive models of context. Indeed, looking at future context variations affecting the reconfiguration choices, it is possible to achieve better performance of the reconfiguration process. As presented in [25] the authors exploit a probabilistic user preference model for achieving reconfiguration of self-adaptive systems with better performance while in [6] the authors propose an approach for achieving better reconfiguration performance based on a predictive model concerning the availability of contextual resources. Following the idea of these approaches, our aim is to optimize the performance of the reconfiguration process for DISA systems by adopting a predictive process model containing information about current and future data accesses. As shown in Section 3.2, it is possible to extract processes of data accesses starting from historical information as the basis for getting a predictive model. Based on this model, we envisage the adoption of a multi-objective optimization technique with the aim of promoting better performance for the reconfiguration of data. This technique should support the decision-making process by enabling the evaluation of the most *suitable* configuration of data that should be adopted for *design-time* and *run-time adaptations.*

Let us consider the e-health case study where the doctor is changing his activity from visits management to department administrator and let us suppose that he remains department administrator for a short period of time before coming back to his visits. As soon as he becomes administrator, his application has to be reconfigured in order to include the data required for performing the new activity. Nevertheless it is not convenient to discard all the data required for performing the visits since the doctor will soon return to visit patients. To this end, as much as possible data regarding visits should be maintained into the device taking into account the future context variations.

3.4 Techniques for implementation

The lifecycle process we envisage for DISA systems can benefit from current practice technologies available in the literature. We present features by defining their requirements (e.g. as Linear Time Temporal Logic expressions), context requirements as predicates and data excerpts as sets of entity types of the conceptual schema. We formalize a *SAT problem* (e.g., JaCoP tool[2]) for evaluating the context states in which each configuration of features is admissible according to its context predicate. Since more than one configuration may be admissible at a certain context state, we envisage the adoption of *multi-objective optimization techniques* for selecting the best possible configuration of data. Once such a configuration is identified, we envisage the application of a *filtering technique* for creating the subset of the global conceptual schema suited for the best configuration at the current context [23]. We consider *schema transformation techniques* for making the subset of the conceptual schema consistent with the global schema. Then, through *bi-directional techniques* we propagate conceptual modifications to the logical schema and finally to data instances. We plan to model data-related artifacts with DB-MAIN tool[3] while we plan to adopt MySQL[4] DBMS for data instances. As far as the migration problem is concerned, we plan to adopt *query parsers* (e.g., *JSqlParser*[5]) for extracting semantic information of data-related events to be applied to a *Formal Concept Analysis technique* [13], (e.g., *colibri-java*[6]) with the aim of clustering queries which implement the same high-level data-manipulation function [15]. Once high-level data accesses have been identified, we plan to adopt the de-facto standard for process mining, i.e., *ProM* tool[7] to extract data-oriented processes of legacy data-intensive systems.

4 Related Work

To the best of our knowledge, a lifecycle process that supports creation, migration and optimization of DISA systems has not been yet proposed in the literature. This process should support design-time and run-time adaptivity of accessing data. Most of approaches presented in the literature consider only specific problems within the process and they provide only design-time variability;

[2] http://jacop.osolpro.com
[3] http://www.db-main.be
[4] http://www.mysql.com
[5] http://jsqlparser.sourceforge.net
[6] http://code.google.com/p/colibri-java
[7] http://www.promtools.org

they follow either a pruning (top-down) or a merging (bottom-up) perspective for creating the data of interest (subset) from a global data model. In [2], the authors propose a merging approach for tailoring the logical schema to the current context instance which is modeled separately from the schema. In [7], the authors work on a logical schema for producing the excerpt of data that fits the current context-dependent user preferences. In [42], the authors propose a filtering approach for creating a consistent excerpt of the conceptual schema starting from a required subset of it. In [38], the authors propose a feature-oriented approach for tailoring the data of interest from a conceptual schema. They model the variability of accessing data in terms of features but they do not link this variability to context variations. In [3], the authors present a design technique for creating very small databases from a big data source by considering conceptual and logical schema. Context variations are not taken into account, thus their approach does not support context-dependent run-time variations of data. As far as run-time variability is concerned, the approach presented in [30] shows how to achieve un-predictable variations of the context and how to propagate this variations to the relational database.

Altought adaptivity of data can be achieved by considering artifacts at different abstraction levels, it is not still clear which is the process to follow in order to create, to migrate and to optimize DISA systems that support design-time and run-time adaptivity of accessing data. A link between application variability and data variability has not been yet implemented in the literature making it difficult to propagate variations of context to variations of required data. Most of approaches support design-time variations of data while there is almost no support for foreseen and unforeseen run-time variations; the latter are becoming more and more important given that it is not always possible to provide the complete space of reconfiguration choices at design time.

5 Conclusions

We discussed the critical problems of data-intensive systems and the need of introducing adaptivity to ease the management of big amount of data for which a context-dependent approach makes sense, i.e., different users accessing the same system, heterogeneous environments where the application runs and heterogeneous processes to be completed over data. We presented a unique lifecycle process for DISA systems and we showed how to solve the three critical problems for the management of data variability. We proposed a methodological solution and a possible integrated implementation which exploits techniques presented in the literature. As for future work we will implement our integrated solution for the lifecycle of DISA systems and we will experiment it at a large scale with a real e-health system, e.g., OSCAR database[8] which contains a huge amount of data of interest for differents stakeholders performing heterogeneous care processes in different contexts.

[8] http://www.new.oscarmanual.org

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 L. Tanca. Carve: Context-aware automatic view definition over relational databases. *IS*, 38(1):45–67, 2012.
3. C. Bolchini, F. A. Schreiber, and L. Tanca. A methodology for a very small data base design. *Inf. Syst.*, 32(1):61–82, 2007.
4. Y. Brun et al. Engineering self-adaptive systems through feedback loops. In *Self-Adaptive Systems*, volume 5525, pages 48–70, 2009.
5. B. H. C. Cheng et al., editors. *Self-Adaptive Systems*, volume 5525 of *LNCS*, 2009.
6. S.-W. Cheng, V. Poladian, D. Garlan, and B. R. Schmerl. Improving architecture-based self-adaptation through resource prediction. In *SEFSAS*, pages 71–88, 2009.
7. P. Ciaccia and R. Torlone. Modeling the propagation of user preferences. In *ER*, pages 304–317, 2011.
8. A. Classen, P. Heymans, and P.-Y. Schobbens. What's in a feature: A requirements engineering perspective. In *FASE*, pages 16–30, 2008.
9. A. Cleve, J.-R. Meurisse, and J.-L. Hainaut. Database semantics recovery through analysis of dynamic sql statements. *J. Data Semantics*, 15:130–157, 2011.
10. A. Cleve, N. Noughi, and J.-L. Hainaut. Dynamic program analysis for database reverse engineering. In *GTTSE*, pages 297–321, 2011.
11. K. Czarnecki and M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In *GPCE*, pages 422–437, 2005.
12. K. Czarnecki, J. N. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. F. Terwilliger. Bidirectional transformations: A cross-discipline perspective. In *ICMT*, pages 260–283, 2009.
13. B. Ganter, R. Wille, and R. Wille. *Formal concept analysis.* Springer Berlin, 1999.
14. M. Glinz. On non-functional requirements. In *RE*, pages 21–26, 2007.
15. C. D. Grosso, M. D. Penta, and I. G. R. de Guzmán. An approach for mining services in database oriented applications. In *CSMR*, pages 287–296, 2007.
16. P. Inverardi and M. Mori. Model checking requirements at run-time in adaptive systems. In *ASAS '11*, pages 5–9, 2011.
17. 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.
18. G. Karsai, A. Ledeczi, J. Sztipanovits, G. Peceli, G. Simon, and T. Kovacshazy. An approach to self-adaptive software based on supervisory control. IWSAS'01, pages 24–38, 2003.
19. D. O. Keck and P. J. Kühn. The feature and service interaction problem in telecommunications systems. a survey. *IEEE TSE*, 24(10):779–796, 1998.
20. R. Mans, W. M. P. van der Aalst, R. J. B. Vanwersch, and A. J. Moleman. Process mining in healthcare: Data challenges when answering frequently posed questions. In *ProHealth/KR4HC*, volume LNAI, pages 140–153, 2012.
21. R. S. Mans, H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker. Application of process mining in healthcare - a case study in a dutch hospital. In *BIOSTEC (Selected Papers)*, pages 425–438, 2008.
22. 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.
23. A. Metzger et al. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *RE*, pages 243–253, 2007.

24. 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)*, volume 7706 of *LNCS*, pages 85–105. Springer, 2013.
25. 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*, volume 7041 of *LNCS*, pages 286–301, 2011.
26. H. R. M. Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah. Event correlation for process discovery from web service interaction logs. *VLDB J.*, 20(3):417–444, 2011.
27. R. Nzekwa, R. Rouvoy, and L. Seinturier. A flexible context stabilization approach for self-adaptive application. In *PerCom*, pages 7–12, 2010.
28. C. Parra et al. Using constraint-based optimization and variability to support continuous self-adaptation. In *SAC*, pages 486–491, 2012.
29. C. A. Parra, A. Cleve, X. Blanc, and L. Duchien. Feature-based composition of software architectures. In *ECSA*, pages 230–245, 2010.
30. E. Quintarelli, E. Rabosio, and L. Tanca. Context schema evolution in context-aware data management. In *ER*, pages 290–303, 2011.
31. Á. Rebuge and D. R. Ferreira. Business process analysis in healthcare environments: A methodology based on process mining. *Inf. Syst.*, 37(2):99–116, 2012.
32. B. Roy. *Multicriteria Methodology for Decision Aiding.* Kluwer Academic Publishers, 1996.
33. A. Rozinat, I. S. M. de Jong, C. W. Günther, and W. M. P. van der Aalst. Process mining applied to the test process of wafer scanners in asml. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 39(4):474–479, 2009.
34. M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *TAAS*, 4(2), 2009.
35. 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.
36. M. Schäler, T. Leich, M. Rosenmüller, and G. Saake. Building information system variants with tailored database schemas using features. In *CAiSE*, pages 597–612, 2012.
37. P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, 2007.
38. 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*, 2009.
39. 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.
40. W. M. P. van der Aalst. Process mining: Overview and opportunities. *ACM Trans. Management Inf. Syst.*, 3(2):7, 2012.
41. W. M. P. van der Aalst et al. Process mining manifesto. In *Business Process Management Workshops (1)*, pages 169–194, 2011.
42. A. Villegas and A. Olivé. A method for filtering large conceptual schemas. In *ER*, pages 247–260, 2010.
43. A. Villegas, A. Olivé, and M.-R. Sancho. On computing the importance of associations in large conceptual schemas. In *Conceptual Modelling and Its Theoretical Foundations*, volume 7260 of *LNCS*, pages 216–230, 2012.
44. P. Vincke. *Multicriteria Decision-Aid.* J. Wiley, New York, 1992.