

Context-Oriented Software Adaptation - A Brief State of the Art

Kim Mens, Anthony Cleve, Bruno Dumas

Abstract. In this document we provide an extensive, though not exhaustive, overview of existing related research in the areas of behavioural adaptation, user interface adaptation and database adaptation.

1. Behavioural Adaptation

This section provides an overview of the research that has been conducted on approaches that support the development of software systems allowing a consistent and dynamic adaptation of their behaviour according to their current internal state and the situation of their surrounding environment. System monitors and sensors provide information about the various situations to which software systems could adapt. Dynamic software adaptation of the system takes place with the objective of providing a more appropriate behaviour and quality of service of the software system with respect to the current situation of its execution environment. This requires such software systems to:

1. provide the necessary mechanisms for continuous adaptation, allowing components to be introduced into or removed from the system at any time, without stopping the system's execution;
2. ensure that the system's adaptations remain consistent with the expected behaviour of the system and other adaptations made to the system.

These characteristics are important to ensure that the system effectively changes to provide a more appropriate service with respect to its changing environment, and to ensure that it behaves as expected under all possible situations to which it may be adapted and all combinations of available adaptations. Different mechanisms have been proposed to enable dynamic software adaptation with respect to the environment. Ensuring consistency of these adaptations, however, is challenging, given that the number of possible adaptation combinations increases exponentially with the introduction of every new adaptation.

Regarding the mechanisms to achieve dynamic software adaptation, following Cardozo [BH1] we distinguish three categories, according to the abstraction level at which the particular mechanisms for achieving behavioural adaptation are implemented: architectural solutions, middleware solutions and language solutions. Alternative but equally interesting taxonomies that focus more on the composition mechanisms or programming technologies are given, respectively, by McKinley [BH2] and Bainomugsha [BH3].

Architectural Solutions

The category of architectural solutions rely essentially on using dedicated software architectures and designs to enable adaptation of behavioural adaptations at run time.

Design patterns [BH4, BH5], for example, provide recurring solutions to particular design problems, often related to reusability and maintainability. Particular design patterns, such as the state pattern, strategy pattern, decorator, abstract factories and dynamic proxies, provide dedicated solutions for enabling dynamic system adaptations. Although these patterns do enable behavioural adaptations of a system, the solutions are rigid in the sense that the adaptations need to be explicitly foreseen and the system needs to be structured in a particular way to accommodate these adaptations.

Dynamic software updates [BH6, BH7] address the need to upgrade the behaviour of software systems without stopping them, while respecting consistency and robustness of the system, for example when adding new features, bug fixes or patches to a software system. Most approaches for dynamic software upgrades support only forward evolution of adaptations however, that is, adaptations can be introduced but not removed from the system after they have been introduced.

Software product lines [BH8] are a structured approach to design not just a single but an entire product line of different software products. They support customization of the product line into a single product either statically, at design time, by selecting the appropriate components and features building up a product, or even dynamically by generating or transforming the required system components at run-time.

Such **dynamic software product lines** [BH9, BH10] rely on different techniques to enable such transformations, for example, regenerative, composable components construction, or incremental-move [BH8]. Using dynamic software product lines effectively allows the reconfiguration of systems at run time. However, again all system adaptations need to be foreseen in order to create the variability model describing the adaptations and to specify the concrete transformation strategy. Moreover, the points of the program in which adaptations should take place must also be defined beforehand.

The main drawback of these architectural solutions is thus that all adaptations need to be anticipated beforehand, which may be too strong a constraint for the setting of highly changing environments.

Middleware Solutions

The category of *middleware solutions* explores platforms and frameworks focused on reducing the complexity of building software systems that can dynamically adapt their behaviour, by shifting complexity from the application design towards a reusable middleware architecture. Most of the existing middleware approaches that deal with adaptation of a system at run time are targeted to work in a distributed environment. Since in this particular research project we are less interested in the technical communication issues between different remote components of a system, in this section we only discuss those aspects of middleware solutions related to dynamic adaptation of a system.

Dependency injection [BH11], sometimes referred to as Inversion of Control, achieves dynamic adaptation at the granularity of software components, allowing components to be replaced freely by other components, as long as they respect a common interface. Different components can be dynamically injected into the application according to the current situation. But as for the architectural solutions, such behavioural adaptations must be anticipated by the programmer, and the system components need to be designed with this kind of adaptation in mind. This approach is thus less suited for the definition of fine-grained behavioural adaptations, and provides little support for managing the subtle interactions that may occur between two independently injected components or to combine different behavioural adaptations for a same component.

Service-oriented architecture (SOA) [BH12] is a middleware approach for the construction of a system from loosely coupled services. Each component of behavioural functionality is designed and provided as an independent service of the system. Inter-service communication takes place through standard interfaces, usually using XML or a Web Services Description Language (WSDL). A limitation of SOA is that adaptations are defined at a coarse-grained level, allowing only complete services to be adapted. In recent years different approaches have been proposed to allow a dynamic interchange of services in SOA, such as context-aware mashups [BH13].

Cloud computing [BH15] provides an infrastructure where storage and computation services are leased to users and adapted dynamically to particular situations such as user load and available computation resources. It has opened the door to the new trend of **Software-as-a-Service** (SaaS) [BH14] where software products are provided to the end user as a combination of different accessible services. Combining these ideas also rise to our vision of **applications as feature clouds** [BH16], providing software services that are composed dynamically from a set of available fine-grained features. Such a feature cloud programming model can be realized by relying on context-oriented programming technology (see further), enabling new or adapted features to be added to or removed from running applications on demand, according to contextual information, and taking into account feature dependencies.

Event systems [BH17] provide a middleware platform to program systems in terms of actions that are executed automatically whenever certain conditions are satisfied. This makes them an interesting candidate to build systems supporting dynamic adaptations. However, they present some drawbacks for defining adaptations at different levels of granularity, and for defining adaptation events independently of the core functionality of the system.

Self-adaptive systems [BH18], also known as autonomic computing or self-managing systems, are conceived as closed-loop systems with feedback from their self and from the surrounding context. Müller et al. [BH19] advise highly dynamic software systems whose operations may need to adapt dynamically to changing requirements and uncertainty in their execution environments, to be instrumented with self-adaptation mechanisms to monitor selected requirements and environment conditions to assess the need for evolution, plan desired changes, as well as validate and verify the resulting system.

Language Solutions

The category of language solutions explore programming paradigms that provide language facilities to enable the definition and introduction of adaptive behaviour in software systems.

Metaprogramming facilities [BH20] extend existing programming languages with the ability to write programs that can reason about and manipulate themselves. As such, they can be used to write programs that can dynamically modify their own behaviour. However, since they don't provide any dedicated higher-level abstractions to represent contexts and behavioural adaptations to contexts, they are mostly useful as an underlying technique based on which more high-level language solutions can be developed.

Aspect-oriented programming [BH21] tackles the problem of modularizing behaviour that cuts across the base decomposition of a software system. By providing a notion of *aspects* that can be woven inside the base code of the system (either statically or dynamically) it can be used to define and apply behavioural adaptations to a system's base behaviour, but still remains mostly a technical solution.

Context-oriented programming languages [BH22] take this idea of separation of concerns between the base behaviour of a system and its behavioural adaptations further, by providing dedicated language constructs for representing and managing contexts, context dependencies, and behavioural adaptations to context. As such they provide a high level of dynamicity for the introduction and withdrawal of fine-grained behavioural adaptations. Many COP languages have been proposed [BH22, BH23, BH24, BH25, BH26, BH27, BH28, BH29, BH30, BH31, BH32, BH33, BH34, BH35, BH36, BH37, BH38], most of which have been implemented as extensions to traditional object-oriented languages by relying upon metaprogramming or aspect-oriented programming techniques.

Feature-oriented programming [BH39, BH40] is akin to context-oriented programming [BH41, BH42] but grew from the software product line and feature modelling research communities, as a mechanism to write programs in terms of software features and their interactions, and even to activate and deactivate such features at runtime. We have recently shown that both paradigms show quite some similarities [BH42] and have started to explore a new generation of context-oriented programming languages by taking up ideas from feature-oriented programming [BH16]. Although many other programming paradigms exist that provide mechanisms for defining dynamic behaviour adaptations, such as **reactive programming** [BH3, BH43, BH44] and **agent-oriented programming** [BH45, BH46], because of the built-in abstractions it provides for dealing with context and adaptation to context, the paradigm of context-oriented programming seems to be one of the most promising ones to truly achieve context-oriented adaptation. Nevertheless, useful ideas regarding how and when to adapt and how to manage the interaction between different behavioural adaptations, can be borrowed from these other paradigms.

Partly orthogonal to the above classification of mechanisms to achieve dynamic behavioural adaptation, the class of dynamically adaptive software systems could also be classified according to the kinds of techniques, models and formalisms they provide for managing the consistency of such systems. In his PhD thesis on the identification and management of inconsistencies in dynamically adaptive software systems [BH1], conducted under the guidance of Prof. Kim Mens at UCL, Nicolás Cardozo conducted an exhaustive survey of models for conflict resolution and inconsistency management. He distinguished four different kinds of approaches. A first kind are the **architectural approaches** that use models to guarantee consistency of the system not only at design and development time, but also during its execution or evolution [BH47, BH48], to monitor and verify particular aspects of the system's behaviour. A second kind are the **formal approaches**, which rely on algebras (e.g., process algebras or coalgebras), logic (e.g., modal or temporal logic), or model checking (e.g., SAT solving), to prove or verify interesting properties of the system. A third kind are the **rule-based approaches** that rely on logic programming or other rule-based engines to manage and coordinate the behaviour of dynamically adaptive software systems. The fourth kind he distinguished are the **state-machine based approaches** that use automata and labelled transition systems, statecharts, dataflow graphs or Petri nets to model, verify and manage the expected states and actions of a dynamically adaptive software system. In particular, the formalism of Context Petri nets [BH1, BH49, BH50] enables the definition of adaptations and their allowed and expected interactions in a context-oriented software system, and provides certain consistency guarantees about the system at design-time, while also serving as a runtime mechanism to ensure that no inconsistencies are introduced. Other authors such as Filieri et al. [BH51] have explored **probabilistic approaches** to verify or guarantee the consistency of self-adaptive systems at run-time.

2. Adaptation of User Interaction

User interface adaptation is the process of modifying a user interface based on a number of parameters [AUI1, AUI2, AUI3]. The interface can be adapted either when creating the interface, or dynamically in real time. The input as well as the output channels of the interface can be adapted. Parameters used when defining how to adapt the interface typically include three aspects: the **user profile**, the **device profile**, and **environmental parameters** [AUI4], with fewer works also exploring the **social surroundings** of the user [AUI5, AUI6]. For example, an application could adapt displayed information based on the current activity of the user: if the user is in a car, the information presented could be related to current traffic jams while if the user is walking in a town, showing information on public transportation available in the immediate surroundings may be more appropriate.

Frameworks in User Interface Adaptation

When considering user adaptation, different ways of adapting the user interface can be considered. On the one hand, the user interface can be created in such a way that it adapts automatically. On the other hand, solutions exist where users are given the possibility to modify the user interface in a proactive way. Malinowski et al. [AUI7] proposed a complete taxonomy to describe user interface adaptation. Their taxonomy is based on four different stages of adaptation in a given interface: initiative, proposal, decision and execution. Each of these adaptations can be performed by the user, the machine or both. Researchers have also explored how frameworks could help the definition of adaptive interfaces [AUI8]. López-Jaquero et al. [AUI9] extend Malinowski's taxonomy into the **ISATINE framework** with seven stages of adaptation: the goals for user interface adaptation, the initiative for adaptation, the specification of adaptation, the application of adaptation, the transition with adaptation, the interpretation of adaptation as well as the evaluation of adaptation. Calvary et al. [AUI4, AUI10] presented a model-based user interface framework for **multi-target interfaces** (also called **plasticity** of user interfaces), allowing the description of design and runtime phases without taking into account specific implementation requirements. Multi-target refers to multiple contexts, where context denotes the context of use of an interactive system described in terms of a user, platform and environment model. The user model contains information about the current user(s), such as user preferences or limitations of disabled users. The platform model describes physical characteristics of the device the system is running on, including screen size or processor speed. Finally, the environment model contains information about social and physical attributes of the environment in which the interaction takes place. Beyond multi-target interfaces, multiple studies have explored how an interface can **migrate** from one device to the other [AUI11, AUI12] or even be **distributed** on multiple devices [AUI13, AUI14].

Adaptive Multimodal User Interfaces

While adaptation has been investigated mostly for traditional Window/Icon/Menu/Pointer (WIMP) interfaces [AUI15] and context-aware interfaces [AUI16], work devoted to the automatic adaptation of **multimodal interfaces** based on contextual information is more limited. Multimodal interfaces are user interface relying on multiple input and output modalities (such as speech, gesture, emotions...) to enable the communication between a user and her device [AUI17]. Since early 2000, researchers have been exploring the possibilities of linking **multimodal interfaces** and **user interaction adaptation** [AUI18]. Evidence that context influences a user's preferences while interacting with a mobile device has been found by Lemmelä et al. [AUI19]. Additionally, they identified which modalities and combinations of modalities best suit different situation requirements. For test purposes, an SMS application was developed and evaluated in a car as well as in a pedestrian context. Speech input was assigned as the default interaction technique for the car environment whereas 2D gestures (e.g. finger strokes) and motion gestures (e.g. tilt) were used in the walking environment. Doyle et al. [AUI20] conducted a review and analysis of existing map-based multimodal systems. They further proposed and evaluated a novel multimodal mobile geographic information system (GIS) called Compass. Parameters such as effectiveness and efficiency of unimodal interaction were evaluated. To deal with the complexity of working with digital maps on small screen displays, Ramsay et al. [AUI21] proposed the use of motion gestures like tilting an external device backwards and forwards to navigate within a map. Reufer et al. [AUI22] investigated different motion gesture actions by using built-in smartphone sensors. They proposed a multimodal game using speech, 2D gestures and motion gestures as input modalities with

different actions specifically assigned to each modality. For example, 2D gestures like touch on the display were used for selecting and erasing content whereas motion gestures such as rolling (rotating the device around the z-axis) or pitching (rotating the device around the x-axis) were used to control different game options. In Bühler et al. [AUI23] the major concern was the flexible control of interaction modalities so that users as well as the system were able to make transitions between the different modalities. They presented a conceptual framework that described five combinations of input and output modalities based on a user's level of attention within a car and a pedestrian setting. Last but not least, David et al. [AUI24] proposed a mobile middleware to facilitate the development and maintenance of adaptive mobile applications. An instant messenger prototype was built to illustrate their approach. The application allowed users to read and write SMS messages using either the device's keyboard or speech. The application adapted its user interface depending on the user's movements. These last two works are particularly interesting, as they focus on the adaptation of multimodal input. Based on the results of these different researches, in Dumas et al. [AUI25] we further explored the possibilities of **automatic adaptation of multimodal interfaces** with a calendar application accepting accelerometer-based movements, gestures, RFID-tagged objects and multi-touch as input modalities.

3. Database Adaptation

In this section we study the state of the art in database adaptation, starting from database evolution in the large, via more fine-grained static database evolution mechanisms and runtime database adaptation mechanisms, to more recent research on context-aware databases.

Database Evolution in the Large

The general problem of **database evolution** is a largely explored problem within the data modeling, database engineering and data management communities. Database migration, database merging and database restructuring are popular database evolution scenarios that involve not only changing the data components of applications, but also rewriting some parts of the programs themselves, even when no functional change occurs. In general, such evolution patterns involve the **joint modification** of several mutually dependent system components, namely the data structures and constraints (i.e., the **schemas**), the **mappings** between the schemas, the **data instances** and the **programs** (i.e. the **data manipulation behaviour**). When the system evolves, the **consistency** that exists between these those artifacts must be preserved.

Database Evolution via Coupled and Bidirectional Transformations

A promising approach to formalize and support such a co-adaptation process is by means of **coupled transformations** (a.k.a. **co-transformations**). The general concept of co-transformation was defined by Lämmel as follows: “*A co-transformation transforms mutually dependent software artifacts of different kinds simultaneously, while the transformation is centred around a grammar (or schema, API, or a similar structure) that is shared among the artifacts*” [DB1]. Generally, “*two or more artifacts of potentially different types are involved, while transformation at one end necessitates reconciling transformations at other ends such that global consistency is reestablished*” [DB2].

Several authors have investigated the use of coupled transformations in the domain of database evolution. The 2LT project [DB3] aims to formalize and to support such a particular instance of coupled transformations, namely **two-level transformations**, which involve a transformation on the level of types with transformations on the level of values and operations. This generic approach revealed to be applicable to the coupled transformation of database schemas, data instances, queries, and constraints [DB4].

The PRISM system [DB5, DB6] provides an integrated support to relational schema evolution that includes (1) a language for the concise specification of modification operators for relational schemas, (2) impact analysis techniques that allow to evaluate the impact of schema changes, (3) automatic data migration support, (4) optimized translation of old queries to work on the new schema and (5) full documentation of the changes involved by the evolution. Query adaptation derives from the schema modification operators and combines a technique called chase and back-chase for **query rewriting** and **SQL view generation**.

Terwilliger et al [DB7] propose a comprehensive approach to the **co-evolution** of **conceptual** schema, **physical** schema and **mappings** in the particular context of the evolution of systems based on Object-Relational-Mapping technology. Their main focus is on achieving some local consistency of the chosen mapping patterns over time.

In previous work [DB8], we proposed a co-transformational approach to data-intensive system evolution, by focusing on semantics-preserving database evolution scenarios such as platform migration and schema refactoring. According to this approach, each generic semantics-preserving schema transformation is associated to a set of abstract query transformation rules allowing to propagate successive database restructuring to the program level (thereby preserving global consistency). Those co-transformation rules can then be composed to support the successive application of schema refactorings. We show in [DB9] that the combination of **transformational and generative techniques** provides a sound basis for the co-evolution of database schemas and programs.

An interesting particular category of coupled transformations are **bidirectional transformations**. According to its most generally agreed definition [DB10], a bidirectional transformation (bx) between two sources of information A and B consists of a pair of unidirectional transformations: one from A to B and another from B back to A. In many cases, the flow of data from A to B dominates the flow of data from B to A. In these cases, A is called the input/source/master of the bx, and B is called the output/target/slave. The transformation from A to B is called the **forward transformation**, whereas the transformation from B to A is called the **backward or reverse transformation**.

Bidirectional transformations are often seen as a mechanism for **preserving the consistency** of two (or more) inter-dependent software artifacts over time. The use of bx has been actively investigated by researchers from many different areas including software engineering, database engineering, programming languages, and document engineering to solve diverse problems such as: (1) constructing updatable views in relational databases; (2) transforming, integrating, and exchanging data; (3) synchronizing replicas of data expressed in different formats; (4) computing and synchronizing views of software models; and (5) providing convenient interfaces for editing complicated data sources.

All the above-mentioned approaches supporting the coupled transformation of schemas, data and behaviour consider this co-adaptation problem **at compile-time** (not at runtime). In other words, the adaptation mechanisms require the system to be **shut down** before it can be adapted, recompiled and redeployed. Furthermore, to the best of our knowledge, the above-mentioned approaches to co-evolving databases and programs only consider **unidirectional** coupled transformations. In addition, most of those approaches are restricted to **information-preserving** database evolution scenarios, that involve **semantics-preserving** schema transformations.

On-the-fly Database Adaptation

Only a few authors have investigated methods, techniques and tools to support **on-the-fly schema updates**. Uddin et al [DB11] propose an approach allowing to apply schema changes at runtime, i.e., without the need to shut down the running data-intensive system. Idreos et al. [DB12] present a complementary **adaptive indexing** technique for optimizing data access performance at runtime without human intervention. Those runtime approaches mainly focus on schema and data adaptation; they do not propose solutions to support schema-behaviour co-adaptation at runtime.

Context-Aware Databases

Data-intensive systems manage increasingly complex and huge amounts of data that are suited for different kinds of users each performing tasks of different nature in different, changing contexts. Most of the effort for designing, maintaining and evolving these systems depends on their complex interactions with (big) data sources. From this perspective, another important problem that needs to be tackled is how to facilitate data-intensive system **variability** in terms of data and data access. This problem has recently started to be addressed in the literature of **context-aware databases** [DB13] by means of methodologies, techniques and tools for creating sub-portions of a global database based on different factors, such as current context, user tasks and user preferences [DB14, DB15]. Several approaches in the literature

consider external factors, i.e. the context, to provide the user with the most relevant data. Preliminary ideas concerning **context-aware data tailoring** have been introduced by Bolchini et al. in 2009 [DB13] along with a generic architecture to enable context-aware access to a data source.

Generic **feature-based** approaches have been proposed by the **software engineering** community with the aim of creating a target model starting from a source model. These approaches, which –as we will see shortly– have been a source of inspiration for the **database engineering** community, rely on either **pruning (filtering)** or **merging** techniques. For instance, in [DB16] the authors support a generic approach to generate UML product models by **pruning** the base model which implements all possible feature configurations. This method evaluates the presence conditions belonging to the fragments of the base model in order to assess which fragments to include in the product according to the selected features. This approach supports the consistency of the products with respect to the base model. Saval et al. [DB17] identify the main challenges and propose a set of good practices for implementing the feature-based **merging** approach whose aim is to create a target model by combining the model fragments belonging to the features to derive. The **merging** approach presented in [DB18] supports the product derivation of architectural models according to the target feature configuration.

In the research literature on context-aware databases, most of the approaches that provide context-based data variability also follow either the **filtering** or the **merging** perspective. Bolchini et al. [DB19] propose a **merging** method to extract the relevant portion of a relational schema based on a hierarchical context model [DB7]. Each context element corresponds to a different **view** of the global schema. By means of two operators, their method supports the creation of the view starting from basic excerpts of data belonging to active context elements. Villegas and Olive [DB21] define a **filtering** method to select an excerpt of the conceptual schema based on a set of entity types that are required in a certain context. The method creates a consistent excerpt of the conceptual schema by starting from an augmented set of entity types. The focus set is then incremented with the entity types that optimize a single utility function which combines the absolute importance of an entity type and its closeness to the focus set.

Ciaccia and Torlone [DB14] present an approach to select the **most suitable data** based on context-dependent **user preferences**. They propose a definition of context as a set of elements that are related hierarchically. Each context may have a different preference, each one expressed as a binary relation over the tuples of a database relation. The authors propose an operator to select the best tuples of a table according to a context-dependent preference relation.

Miele et al. [DB22, DB23] propose an approach to context-dependent **data personalization**, allowing the user to express preferences that specify which data (s)he is more/not interested in depending on each specific context. A partial order can be defined among the data, based on which only the top (most preferred) portion of the data are loaded in a given context. The complementary problem of modeling, composing and exploiting **user preferences** in the context of data management has been addressed by several authors in recent years. A survey of this domain is available in [DB24].

In recent work [DB25], we presented a **theoretical framework** for **database schema adaptation** relating context models, feature models and data models, by considering **two levels of abstraction** for the latter (conceptual and logical schemas). This feature-based, context-oriented adaptation framework includes, among others, three **generic algorithms** for adapting database schemas as a consequence of context variations. We also published preliminary yet promising results in the domain of **run-time reconfigurations** of context-oriented databases [DB26]. In particular, we identify a critical property of the system, i.e., **stability** as the capacity of the system to have the lowest possible variations of data w.r.t. variations of **user-driven data requirements**. This approach aims to optimize stability by exploiting a predictive model expressing future possible **variations of informational needs**. We have formalized, implemented and experimented a multi-objective decision-making process to determine the best reconfiguration of data depending on two relevant metrics representing the input data-oriented predictive model.

Bibliography

References to Research on Behavioural Adaptation

- [BH1] N. Cardozo. Identification and Management of Inconsistencies in Dynamically Adaptive Software Systems. PhD thesis, Université catholique de Louvain, Belgium, September 2013. Advisor: Prof. **K. Mens**.
- [BH2] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, B. H. C. Cheng. A Taxonomy of Compositional Adaptation. Technical Report MSU-CSE-04-17. Department of Computer Science and Engineering, Michigan State University, USA, May 2004 (updated July 2004).
- [BH3] Engineer Bainomugisha. Reactive Method Dispatch for Context-Oriented Programming. PhD thesis, Vrije Universiteit Brussel, Brussels, Belgium, December 2012.
- [BH4] E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison-Wesley Longman, 1995. ISBN: 0-201-63361-2.
- [BH5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. Pattern-Oriented Software Architecture Volume 1: A System of Patterns. John Wiley & Sons, August, 1996. ISBN: 0471958697.
- [BH6] D. Gupta. On-Line Software Version Change. PhD thesis, Indian Institute of Technology, Kanpur, India, November 1994.
- [BH7] Y. Vandewoude. Dynamically updating component-oriented systems. PhD thesis, Katholieke Universiteit Leuven, Belgium, March 2007.
- [BH8] C. Cetina, Ø. Haugen, X. Zhang, F. Fleurey, V. Pelechano. Strategies for variability transformation at run time. In Proceedings of the International Software Product Line Conference, pp. 61–70, Pittsburgh, PA, USA, 2009. Carnegie Mellon University.
- [BH9] S. Hallsteinsen, M. Hinchey, S. Park, K. Schmid. Dynamic Software Product Lines, IEEE Computer 41(4):93-95. IEEE Computer Society Press, April 2008.
- [BH10] J. Lee, K. C. Kang. A Feature-Oriented Approach to Developing Dynamically Reconfigurable Products in Product Line Engineering. In Proceedings of the International Software Product Line Conference, pp. 131–140, IEEE, 2006. ISBN: 0-7695-2599-7.
- [BH11] M. Fowler. Inversion of Control Containers and the Dependency Injection Pattern. January 2004. URL: <http://martinfowler.com/articles/injection.html>
- [BH12] T. Erl. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, August 2005. ISBN: 978-0131858589.
- [BH13] A. Brodt, D. Nicklas, S. Sathish, B. Mitschang. Context-Aware Mashups for Mobile Devices. In J. Bailey, D. Maier, K.D. Schewe, B. Thalheim, X.S. Wang, Web Information Systems Engineering (WISE 2008). Lecture Notes in Computer Science 5175, pp. 280-291. Springer, 2008.
- [BH14] T. Lixin Tao. Shifting paradigms with the application service provider model. IEEE Computer, 34(10):32-39. IEEE Computer Society, October 2001.
- [BH15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. A View of Cloud Computing. Communications of the ACM 53(4):50-58. ACM Press, April 2010.

- [BH16] N. Cardozo, **K. Mens**, S. González, P.-Y. Orban, W. De Meuter. Features on Demand. Proceedings of the International Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM, 2014.
- [BH17] P. Verissimo, A. Casimiro. Event-Driven Support of Real-Time Sentient Objects. In Proceedings of the International Workshop on Object-oriented Real-time Dependable Systems (WORDS). IEEE, January 2003.
- [BH18] M. Salehie, L. Tahvildari. Self-Adaptive Software: Landscape and Research Challenges. ACM Transactions on Autonomous and Adaptive Systems 4(2):14:1--14:42. ACM, May 2009.
- [BH19] H. Müller, N. Villegas. Runtime Evolution of Highly Dynamic Software. In T. Mens, A. Serebrenik, **A. Cleve**: Evolving Software Systems, pp. 229-264, Springer, 2014.
- [BH20] G. Kiczales, J. Des Rivieres. The Art of the Metaobject Protocol. MIT Press, 1991. ISBN: 0262111586.
- [BH21] Gregor Kiczales, John Lamping, Anurag Menhdhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier and John Irwin. Aspect-Oriented Programming. In Proceedings of ECOOP 1997, LNCS 1241, pp. 220-242. Springer-Verlag, 1997. ISBN: 978-3-540-63089-0.
- [BH22] R. Hirschfeld, P. Costanza, O. Nierstrasz. Context-Oriented Programming. Journal of Object Technology, 7(3):125–151, March-April 2008.
- [BH23] Pascal Costanza and Robert Hirschfeld. Reflective layer activation in ContextL. In Proceedings of the 2007 ACM symposium on Applied computing (SAC '07), pp. 1280-1285. ACM, 2007. DOI=10.1145/1244002.1244279
- [BH24] Carlo Ghezzi, Matteo Pradella, and Guido Salvaneschi. Programming language support to context-aware adaptation: a case-study with Erlang. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '10), pp. 59-68. ACM, 2010. DOI=10.1145/1808984.1808991
- [BH25] R. Hirschfeld, P. Costanza, M. Haupt. An Introduction to Context-Oriented Programming with ContextS. In R. Lämmel, J. Visser, J. Saraiva, Generative and Transformational Techniques in Software Engineering II, Lecture Notes in Computer Science 5235, pp. 396-407. Springer, 2008
- [BH26] J. Lincke, M. Appeltauer, B. Steinert, R. Hirschfeld. An open implementation for context-oriented layer composition in ContextJS. Science of Computer Programming (76)12: 1194-1209. Elsevier, December 2011. DOI=10.1016/j.scico.2010.11.013
- [BH27] G. Salvaneschi, C. Ghezzi, M. Pradella. Context-oriented programming: A software engineering perspective. J. Syst. Softw. 85(8):1801-1817. Elsevier, August 2012. DOI=10.1016/j.jss.2012.03.024
- [BH28] G. Salvaneschi, C. Ghezzi, M. Pradella. JavaCtx: seamless toolchain integration for context-oriented programming. In Proceedings of the 3rd International Workshop on Context-Oriented Programming (COP '11). ACM, 2011. DOI=10.1145/2068736.2068740
- [BH29] G. Salvaneschi, C. Ghezzi, M. Pradella. ContextErlang: introducing context-oriented programming in the actor model. In Proceedings of the 11th annual international conference on Aspect-oriented Software Development (AOSD '12), pp. 191-202. ACM, 2012. DOI=10.1145/2162049.2162072
- [BH30] S. González, **K. Mens**, P. Heymans. Highly Dynamic Behaviour Adaptability Through Prototypes with Subjective Multimethods. In Proceedings of the Dynamic Languages Symposium, pp. 77-88. ACM Press, 2007.
- [BH31] S. González. Programming in Ambience: Gearing Up for Dynamic Adaptation to Context. PhD thesis, Université catholique de Louvain, Belgium, October 2008. Advisor: Prof. **K. Mens**.

- [BH32] S. González, **K. Mens**, A. Cádiz. Context-Oriented Programming with the Ambient Object System. *Journal of Universal Computer Science* 14(20):3307-3332, 2008.
- [BH33] S. González, N. Cardozo, **K. Mens**, A. Cádiz, J.-C. Libbrecht, J. Goffaux. Subjective-C: Bringing Context to Mobile Platform Programming. In B. Malloy, S. Staab, M. van den Brand. *Proceedings of the International Conference on Software Language Engineering. Lecture Notes in Computer Science* 6563, pp. 246-265. Springer, 2011.
- [BH34] S. González, **K. Mens**, M. Colacioiu, W. Cazzola. Context Traits: Dynamic Behaviour Adaptation Through Run-time Trait Recomposition. In *Proceedings of the International Conference on Aspect-oriented Software Development (AOSD)*, pp. 209-220. ACM, 2013.
- [BH35] T. Aotani, T. Kamina, H. Masuhara. Featherweight EventCJ: A Core Calculus for a Context-oriented Language with Event-based Per-instance Layer Transition. In *Proceedings of the 3rd International Workshop on Context-Oriented Programming (COP)*, pp. 1:1-1:7. ACM, 2011.
- [BH36] M. Appeltauer, R. Hirschfeld, T. Rho. Dedicated Programming Support for Context-Aware Ubiquitous Applications. In *Proceedings of the The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, pp. 38-43. September 2008.
- [BH37] M. Appeltauer, R. Hirschfeld, M. Haupt, J. Lincke, M. Perscheid. A Comparison of Context-Oriented Programming Languages. In *Proceedings of the International Workshop on Context-Oriented Programming (COP)*, pp. 1-6. ACM, 2009.
- [BH38] M. Appeltauer, R. Hirschfeld, M. Haupt, M. Hidehiko. ContextJ: Context-Oriented Programming for Java. In *Journal of Computer Software of The Japan Society for Software Science and Technology*. June 2010.
- [BH39] Christian Prehofer. Feature-oriented programming: A new way of object composition. *Concurrency and Computation: Practice and Experience*, 13(5):465-501, April 2001.
- [BH40] Sebastian Günther, Sagar Sunkle. rbFeatures: Feature-Oriented Programming with Ruby. *Science of Computer Programming*, 77(3): 152–173, Elsevier, March 2012.
- [BH41] P. Costanza, T. D'Hondt. Feature Descriptions for Context-oriented Programming. In *Proceedings of the 2nd International Workshop on Dynamic Software Product Lines (DSPL'08)*, pp. 9-14, 2008.
- [BH42] Nicolás Cardozo, Sebastian Günther, Theo D'Hondt, **Kim Mens**. Feature-Oriented Programming and Context-Oriented Programming: Comparing Paradigm Characteristics by Example Implementations. *Proceedings of the International Conference on Software Engineering Advances (ICSEA 2011)*, pp.130-135. 2013.
- [BH43] Wan, Zhanyong and Hudak, Paul. Functional Reactive Programming from First Principles. In *Proceedings of the Conference on Programming Language Design and Implementation (PLDI), SIGPLAN Notices* 35(5):242-252. ACM, May 2000.
- [BH44] E. Bainomugisha, A. L. Carreton, T. van Cutsem, S. Mostinckx, W. De Meuter. A Survey on Reactive Programming. In *ACM Computing Surveys* 45(4):52:1--52:34. ACM, August 2013.
- [BH45] Y. Shoham. *Agent-Oriented Programming*. Artificial Intelligence 60:51–92. Elsevier, 1993.
- [BH46] Brian Henderson-Sellers, Paolo Giorgini. *Agent-Oriented Methodologies*. IGI Global, 2005. ISBN: 978-1-591-40581-8.
- [BH47] Ragnhild Van Der Straeten. *Inconsistency Management in Model-Driven Engineering*. PhD thesis, Vrije Universiteit Brussel, Brussels, Belgium, September 2005.

- [BH48] Bencomo, N., France, R.B., Cheng, B.H.C., Almann, U. (Editors.). *Models@run.time: Foundations, Applications, and Roadmaps*. Lecture Notes in Computer Science 8378. Springer, 2014.
- [BH49] N. Cardozo, S. Gonzalez, **K. Mens**, R. Van Der Straeten, T. D'Hondt. Modeling and Analyzing Self-adaptive Systems with Context Petri Nets. In *Proceedings of the Symposium on Theoretical Aspects of Software Engineering (TASE 2013)*, IEEE Computer Society, pp. 191-198, 2013.
- [BH50] N. Cardozo, S. Gonzalez, R. Van Der Straeten, **K. Mens**, J. Vallejos, T. D'Hondt. Semantics for Consistent Activation in Context-Oriented Systems. *Journal of Information and Software Technology*, Vol. 58, pp. 71-94. Elsevier, February 2015.
- [BH51] Antonio Filieri, Giordano Tamburrelli. Probabilistic Verification at Runtime for Self-Adaptive Systems. In *Assurances for Self-Adaptive Systems*, Javier Cámara, Rogério de Lemos, Carlo Ghezzi, Antónia Lopes. *Lecture Notes in Computer Science* 7740, pp. 30-59. Springer, 2013
- [BH52] A. Hubaux, A. Cleve, P. Y. Schobbens, A. Keller, O. Muliawan, K. Mens, D. Deridder, R. Van Der Straeten. Towards a Unifying Conceptual Framework for Inconsistency Management Approaches. Technical Report, University of Namur, 2009.
- [BH53] B. Desmet, J. Vallejos, P. Costanza, W. De Meuter, T. D'Hondt. Context-Oriented Domain Analysis. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, L. Vieu (Eds.), *Modeling and Using Context*. *Lecture Notes in Computer Science* 4635, pp. 178-191. Springer, 2007.
- [BH54] N. Cardozo, S. González, **K. Mens**, T. D'Hondt. Safer Context (de)Activation Through the Prompt-Loyal Strategy. In *Proceedings of the Workshop on Context-Oriented Programming COP'11*. ACM Press, 2011.

References to Research on User Interface Adaptation

- [AUI1] Billsus, D., Brunk, C. A., Evans, C., Gladish, B., & Pazzani, M. Adaptive interfaces for ubiquitous web access. *Communications of the ACM*, 45(5), pp. 34-38, 2002.
- [AUI2] Eisenstein, J., Vanderdonckt, J., & Puerta, A. Adapting to mobile contexts with user-interface modeling. *Third IEEE Workshop on Mobile Computing Systems and Applications*. pp. 83-92, IEEE, 2000.
- [AUI3] Paterno, F., & Santoro, C. One model, many interfaces. In *Computer-Aided Design of User interfaces*. pp. 143-154, Springer Netherlands, 2002.
- [AUI4] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. In *Interacting with Computers* 15, 3, pp. 289-308, 2003.
- [AUI5] Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D. Context is Key. *Communication of the ACM*, Volume 48, Issue 3, ACM Press, pp. 49-53, 2005.
- [AUI6] Carmagnola, F., Cena, F., & Gena, C. User model interoperability: a survey. *User Modeling and User-Adapted Interaction* 21.3, pp. 285-331, 2011.
- [AUI7] Malinowski, U., Thomas, K., Dieterich, H., and Schneider-Hufschmidt, M. A Taxonomy of Adaptive User Interfaces. In *Proceedings of HCI 1992*, pp. 391-414, 1992.
- [AUI8] Motti, V. G., & Vanderdonckt, J. A computational framework for context-aware adaptation of user interfaces. In *Seventh International Conference on Research Challenges in Information Science (RCIS 2013)*, pp. 1-12, IEEE, 2013.
- [AUI9] López-Jaquero, V., Vanderdonckt, J., Montero, F., and González, P. Towards an Extended Model of User Interface Adaptation: The Isatine Framework. In *Engineering Interactive Systems*, vol. 4940 of LNCS. Springer, pp. 374-392, 2008.

- [AUI10] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., FLorins, M. & Vanderdonckt, J. Plasticity of user interfaces: A revisited reference framework. In *Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design*. 2002.
- [AUI11] Berti, S., & Paternò, F. Migratory multimodal interfaces in multidevice environments. In *Proceedings of the 7th international conference on Multimodal interfaces*, pp. 92-99, ACM, 2005.
- [AUI12] Grolaux, D., Van Roy, P., & Vanderdonckt, J. Migratable user interfaces: Beyond migratory interfaces. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 422-430, IEEE, 2004.
- [AUI13] Elmqvist, N. Distributed user interfaces: State of the art. In *Distributed User Interfaces*, pp. 1-12, Springer London, 2011.
- [AUI14] Vanderdonckt, J. Distributed user interfaces: how to distribute user interface elements across users, platforms, and environments. *Proc. of XI Interacción*, 20, 2010.
- [AUI15] Cesar, P., Vaishnavi, I., Kernchen, R., Meissner, S., Hesselman, C., Boussard, M., Spedalieri, A., Bulterman, D. C., and Gao, B. Multimedia Adaptation in Ubiquitous Environments: Benefits of Structured Multimedia Documents. In *Proceedings of DocEng 2008 (São Paulo, Brazil)*, pp. 275-284, 2008.
- [AUI16] Adrienko, N. and Adrienko, G. Spatial generalization and aggregation of massive movement data. In *IEEE Transactions on Visualization and Computer Graphics*, 17(2), pp. 205-219, 2011.
- [AUI17] Dumas, B., Lalanne, D., and Oviatt, S. Multimodal Interfaces: A Survey of Principles, Models and Frameworks. In *Human Machine Interaction: Research Results of the MMI Program*, pp. 3–26, Springer, 2009.
- [AUI18] Duarte, C., & Carriço, L. A conceptual framework for developing adaptive multimodal applications. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pp. 132-139, ACM, 2006.
- [AUI19] Lemmelä, S., Vetek, A., Mäkelä, K., and Trendafilov, D. Designing and Evaluating Multimodal Interaction for Mobile Contexts. In *Proceedings of the 10th International Conference on Multimodal Interfaces (ICMI 2008, Chania, Greece)*, pp. 265-272, ACM, 2008.
- [AUI20] Doyle, J., Bertolotto, M., and Wilson, D. A Survey of Multimodal Interfaces for Mobile Mapping Applications. In *Map-based Mobile Services*, L. Meng, A. Zipf, and S. Winter, Eds., *Lecture Notes in Geoinformation and Cartography*, pp. 146-167, Springer, 2008.
- [AUI21] Ramsay, A., McGee-Lennon, M., Wilson, G., Gray, S., Gray, P., and De Turenne, F. Tilt and Go: Exploring Multimodal Mobile Maps in the Field. In *Journal on Multimodal User Interfaces*, 3, pp. 167-177, Springer, 2010.
- [AUI22] Reufer, T., Panknin, M., and Geiger, C. SensoDroid: Multimodal Interaction Controlled Mobile Gaming. In *Proceedings of the 13th International Conference on Humans and Computers (HC 2010, Aizu-Wakamatsu, Japan)*, pp. 32-36, 2010.
- [AUI23] Bühler, D., Minker, W., Häussler, J., and Krüger, S. The SmartKom Mobile Multi-Modal Dialogue System. In *Proceedings of ISCA Tutorial and Research Workshop (ITRW) on Multi-Modal Dialogue in Mobile Environments (Irsee, Germany)*, 2002.
- [AUI24] David, L., Endler, M., Barbosa, S. D. J., and Filho, J. V. Middleware Support for Context-Aware Mobile Applications with Adaptive Multimodal User Interfaces. In *Proceedings of U-Media 2011, 4th International Conference on Ubi-Media Computing (São Paulo, Brazil)*, pp. 106-111, 2011.

- [AUI25] Dumas, B., Solórzano, M. and Signer, B. Design Guidelines for Adaptive Multimodal Mobile Input Solutions. In Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI 2013, Munich, Germany), 2013.
- [AUI18] Duarte, C., & Carriço, L. A conceptual framework for developing adaptive multimodal applications. In Proceedings of the 11th international conference on Intelligent user interfaces, pp. 132-139, ACM, 2006.
- [AUI26] Henriksen, K., Indulska, J. Modelling and Using Imperfect Context Information. In Workshop on Context Modeling and Reasoning (CoMoRea), 2nd IEEE Conference on Pervasive Computing and Communications (PerCom 2004, Orlando, Florida, USA), pp. 33-37, 2004.
- [AUI27] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., & López-Jaquero, V. UsiXML: a language supporting multi-path development of user interfaces. In Engineering human computer interaction and interactive systems, pp. 200-220, Springer, 2005.
- [AUI28] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R.M. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In Proceedings of INTERACT'95, pp. 115-120.
- [AUI29] Costa, D., and Duarte, C. Adapting multimodal fission to user's abilities. Universal Access in Human-Computer Interaction. Design for All and eInclusion, pp. 347-356, Springer, 2011.
- [AUI30] Gajos, K. Z., Czerwinski, M., Tan, D. S., & Weld, D. S. Exploring the design space for adaptive graphical user interfaces. In Proceedings of the working conference on Advanced visual interfaces, pp. 201-208, ACM, 2006.
- [AUI31] Ghiani, G., Paternò, F., Santoro, C., & Spano, L. D. UbiCicero: A location-aware, multi-device museum guide. Interacting with Computers, 21(4), pp. 288-303, 2009.

References to Research on Database Adaptation

- [DB1] R. Lämmel. Coupled Software Transformations (Extended Abstract). In First International Workshop on Software Evolution Transformations, Nov. 2004.
- [DB2] R. Lämmel. Transformations everywhere. Science of Computer Programming. The guest editor's introduction to the SCP special issue on program transformation.
- [DB3] J. Cunha, J. Saraiva, and J. Visser. From spreadsheets to relational databases and back. In Proc. of PEPM '2009, pp. 179–188. ACM, 2008.
- [DB4] J. Visser. Coupled Transformation of Schemas, Documents, Queries, and Constraints. ENTCS, 200(3):3–23, 2008.
- [DB5] C. Curino, H. J. Moon, A. Deutsch, C. Zaniolo. Update Rewriting and Integrity Constraint Maintenance in a Schema Evolution Support System: PRISM++. PVLDB 4(2): 117-128, 2010.
- [DB6] C. Curino, H. J. Moon, C. Zaniolo: Graceful database schema evolution: the PRISM workbench. PVLDB 1(1): 761-772 , 2008.
- [DB7] J. F. Terwilliger, P. A. Bernstein and A. Unnithan. Automated Co-evolution of Conceptual Models, Physical Databases, and Mappings. In Proc. of ER'2010. Lecture Notes in Computer Science 6412, pp. 146-159. Springer, 2010.
- [DB8] **A. Cleve** and J.-L. Hainaut. Co-transformations in database applications evolution. In GTTSE'2005, Lecture Notes in Computer Science 4143, pp. 409–421. Springer, 2006.
- [DB9] **A. Cleve**, A.-F. Brogneaux and J.-L. Hainaut. A conceptual approach to database applications evolution. In Proc. of ER'2010, Lecture Notes in Computer Science 6412, pp. 132–145. Springer, 2010.

- [DB10] K. Czarnecki, J. Foster, Z. Hu, R. Lämmel, A. Schürr, and J. Terwilliger. Bidirectional Transformations: A Cross-Discipline Perspective. In *Proc. of ICMT'2009*, pp. 260-283. Springer, 2009.
- [DB11] I. Neamtii, J. Bardin, Md. Reaz Uddin, Dien-Yen Lin, Pamela Bhattacharya. Improving Cloud Availability with On-the-fly Schema Updates. In *Proc. of the 19th International Conference on Management of Data (COMAD 2013)*, December 2013.
- [DB12] S. Idreos, S. Manegold, H. A. Kuno, G. Graefe: Merging What's Cracked, Cracking What's Merged: Adaptive Indexing in Main-Memory Column-Stores. *PVLDB* 4(9): 585-597, 2011.
- [DB13] C. Bolchini, C. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, L. Tanca. And what can context do for data? *CACM*, 52(11):136-140, 2009.
- [DB14] P. Ciaccia, R. Torlone. Modeling the propagation of user preferences. In *ER*, pp. 304-317, 2011.
- [DB15] D. Martinenghi, R. Torlone. A logical approach to context-aware databases. In A. D'Atri, M. De Marco, A. M. Braccini, F. Cabiddu (eds.), *Management of the Interconnected World*, pp. 211-219. Physica-Verlag HD, 2010.
- [DB16] K. Czarnecki, M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. In *GPCE*, pp. 422-437, 2005.
- [DB17] G. Saval, J. P. Puissant, P. Heymans, T. Mens. Some challenges of feature-based merging of class diagrams. In *VaMoS*, pp. 127-136, 2009.
- [DB18] C. A. Parra, **A. Cleve**, X. Blanc, L. Duchien. Feature-based composition of software architectures. In *ECSA*, pp. 230-245, 2010.
- [DB19] C. Bolchini, E. Quintarelli, R. Rossato. Relational data tailoring through view composition. In *ER*, pp. 149-164, 2007.
- [DB20] E. Quintarelli, E. Rabosio, L. Tanca. Context schema evolution in context-aware data management. In *ER*, pp. 290-303, 2011.
- [DB21] A. Villegas, A. Olive. A method for filtering large conceptual schemas. In *ER*, pp. 247-260, 2010.
- [DB22] Antonio Miele, Elisa Quintarelli, Emanuele Rabosio, Letizia Tanca: ADaPT: Automatic data personalization based on contextual preferences. *ICDE 2014*: 1234-1237
- [DB23] Antonio Miele, Elisa Quintarelli, Letizia Tanca: A methodology for preference-based personalization of contextual data. In *ER*. 2011.
- [DB24] Kostas Stefanidis, Georgia Koutrika, and Evaggelia Pitoura. 2011. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst.* 36, 3, Article 19.
- [DB25] M. Mori and **A. Cleve**. Feature-based adaptation of database schemas. In *Proceedings of the 8th International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2012)*, *Lecture Notes in Computer Science* 7706, pp. 85-105. Springer, 2013.
- [DB26] M. Mori, **A. Cleve**, and P. Inverardi. A stability-aware approach to continuous self-adaptation of data-intensive systems. In *Proceedings of the 2nd International Conference on Context-Aware Systems and Applications*, pp. 304-315. Springer, 2013.
- [DB27] M. Mori and **A. Cleve**. Towards highly adaptive data-intensive systems: A research agenda. In *Proceedings of First International Workshop on Variability Support in Information Systems (VarIS 2013)*, *Lecture Notes in Business Information Processing* 148, pp. 386-401. Springer, 2013.
- [DB28] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, L. Tanca. A data-oriented survey of context models. *SIGMOD Record* 36(4): 19-26, 2007.

[DB29] S. Castro, S. Gonzalez, **K. Mens**, M. Denker. DynamicSchema: A Lightweight Persistency Framework for Context-Oriented Data Management. In Proceedings of the Workshop on Context-Oriented Programming (COP), 2012.