Extending Capabilities with Context Awareness

Martin Henkel[1(✉)], Christina Stratigaki[2], Janis Stirna[1], Pericles Loucopoulos[2], Yannis Zorgios[2], and Antonis Migiakis[2]

[1] Stockholm University, Stockholm, Sweden
{martinh, js}@dsv.su.se
[2] CLMS Ltd, London, UK
xristinastrathgakh@gmail.com, {yz, a.migiakis}@clmsuk.com

Abstract. Organizations have the need to continuously adjust their capabilities to changes in the business context. If existing IT systems and associated development methods does not support this adjustment they need to be changed to do so. However, there exist specialized methods and tools that allow the design, and run-time monitoring of context information. In this paper an approach that allows existing systems to be extended with the management of context information is presented. The approach allows organizations to analyze the potential effect and effort of combing existing systems and tools with specialized tools that handle context information. The purpose of providing means for the integrated use of existing systems and specialized tools is to leverage the strength of both. The approach in grounded in and illustrated by a case of industrial symbiosis.

Keywords: Context analysis · Context monitoring · Capability modelling · Capability management

1 Introduction

Organizations work in changing environments, their capabilities need to be adjusted to meet the changes and in order to increase efficiency. Failure to adapt to changing environments can endeavour organizations profitability or even its existence [1]. The pace at which organizations need to adapt to changes is not slowing down. On the contrary, organizations that have a competitive advantage tend to keep that advantage for a shorter time [2]. Thus, organisations need to be aware of changes in their context and adapt accordingly. Central to being able to perform this adaptation is to know the context of the business, and to be able to measure and monitor it.

In this paper we present an approach that allows organisations to extend their existing IT systems and development tools with tools and methods that allow the analysis, design and monitoring of capabilities and their contexts. The approach makes use of the Capability Driven Development (CDD) [3] methods and associated design and run-time tools. The scope of this approach is (a) to make it possible to, on a case-by-case basis, outline the benefits and effects of extending existing systems and

J. Krogstie et al. (Eds.): CAiSE 2016, LNBIP 249, pp. 40–51, 2016.
DOI: 10.1007/978-3-319-39564-7_4

tools with the ability to design and monitor context information, (b) to make it possible to discern which method and software integrations that needs to be in place for such an extension, (c) to have a basis for the implementation of software components that are needed to perform changes to existing development tools and operational IT systems.

The approach is generic, but is specialized to make use of the CDD approach for capability and context analysis. A key component of CDD is the use of graphical models. The models used in CDD have a focus on describing organisational capabilities, their contexts and goals. CDD does not include specific methods and tools for the detailed creation of information systems. However it does provide tools that allow the analysis, design and run-time monitoring of context information. This can be put in contrast with traditional development tools that focus on components of software systems, such as, data objects, information management procedures, user interface components etc. Thus, combining the CDD approach with other development tools and systems gives the possibility to have support for capability analysis and monitoring (provided by CDD, or similar methods and tools) and the detailed implementation of information systems (provided by existing systems).

The generic approach presented is applied in an industrial symbiosis platform implemented by CLMS Limited. Industrial Symbiosis is an association between two or more industrial actors in which the wastes or by-products of one become the raw materials for another. This collaboration between two or more companies is called a s*ynergy*.

Traditionally, experts and consultants, involving highly complex, labour-intensive and error-prone tasks, performed the industrial symbiosis mediation process for the matching between producers and consumers manually. However, the company CLMS has implemented an *industrial symbiosis platform* that is improving industrial symbiosis by automating the process of mixing and matching the interests of different actors in the waste resource chain, and by providing knowledge-based support for managing resources and finding compatible ones. The platform suggests compatible matches to a company and facilitates the creation of synergies for these matches. The symbiosis platform is implemented using the state-of art model driven development tool zAppDev. As each step of the approach is described in subsequent section, we also describe its application in the symbiosis case.

In order to combine capability and context analysis with existing systems and development tools, both design-time and run-time options for integration need to be considered. We here outline four steps that can be followed to have a structured approach for selecting integration options, these are described in the following sections:

1. Perform capability and context analysis.
2. Select design time integration options.
3. Select run-time integration options.
4. Summarize tool support needed.

The rest of the paper is structured as follows. Section 2 is giving the theoretical background and the related work. Section 3 introduces the first step of the approach, capability and context analysis. Sections 3 and 4 describes subsequent steps in the approach, focusing on design and run-time options when combining existing systems with context aware design tools. Sections 6 and 7 contains summary and conclusions.

2 Related Work

In this paper we use the concepts of capabilities and context. These concepts have been described in the field of strategic management, enterprise modelling and in research on contextual systems.

In strategic management the notion of dynamic capability [4] can be used to describe the ability on an organisation to evolve and thrive in changing environments. The definition of the concept of capability has fluctuated somewhat in the area of strategic management; however there is a tendency to associate it with the organisation of resources and their allocation [5]. In this paper we use this interpretation of capability, that is, we view resources and their use as an integral part of capabilities. We define a capability as the ability and capacity that enable an enterprise to achieve a business goal in a certain context [3].

In the area of enterprise modelling and enterprise architecture, the concept of capability has been used as a mean to analyse organisations [6, 7]. It has also been used to describe an organizations ability to use enterprise architecture. This is most notable in the open groups TOGAF framework [8], where capability frequently refers the readiness of an organization to use enterprise architecture. Regarding strategic management and enterprise modelling, the contribution of this paper is to show how the notions of capability and context can be realised, rather than adding to the existing methods and conceptual foundation of capability analysis and architecture.

Examining the research in contextual systems, it can be said that a focus has been on both to provide extensive and formal descriptions of the context, as well as on development of context aware systems. Generally a context can be defined as information that can be used to characterize a situation of an entity [9], we elaborate this definition in Sect. 3. The relationships of an entity (such as an organisation, or an organisational capability) to its environment are a part of the context. Ontologies have been proposed by several authors as a mean to formally describe contextual information. For example, in [10, 11] the use of ontologies for context descriptions is central. There is also a class of systems referred to as context-aware systems that make context information an integral part of the IT systems. For example, [12] propose a system that in run-time replaces (software) services with adapted versions. Similarly, there have been proposals on how to design systems architectures for context aware systems, see for example [13, 14]. What these context aware systems have in common is that they replaces existing systems, or requires substantial modification of existing systems. In contrast to this, the approach presented in this paper provides a light-weight approach to extend existing systems with context awareness.

3 Capability and Context Analysis

This section provides a brief overview of how capability and context analysis was performed in the symbiosis case. This follows the CDD approach, as defined in [3].

For the purpose of improving the industrial symbiosis case an identification of the capabilities within the business was done. Capability identification is the process of finding the capabilities that the business relies on to function. Identification can be done

in several ways. For example, an identification based on recursive capability refinement is described in our earlier paper [6]. Moreover the identification can be based on concept models, process models or goal models [15].

A small excerpt of the capability model is shown in Fig. 1. Central to the business was in this case the capability to enable web industrial symbiosis (Capability 1 in Fig. 1). This capability relies on that a relevance rating can be determined for each symbiosis (Capability 1.1), that the resources can be described properly (Capability 1.2) and that the symbiosis is compliant with regulations (Capability 1.3). One example of the need for compliance with regulations is that the transport of hazardous goods is regulated. Capability 1 is dependent on the other capabilities, this are shown by using rhombs in Fig. 1.

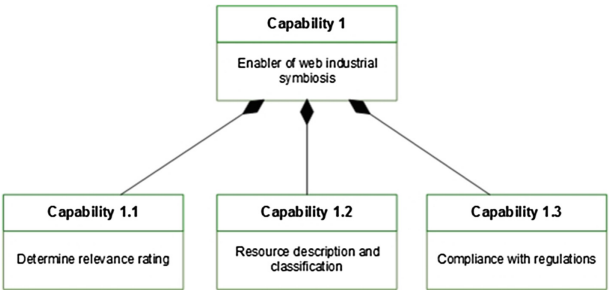

Fig. 1. Excerpt from the capability model for the symbiosis case

Given the identified capabilities their operational context could be analysed. We here start with the definition that context is any information that can be used to characterize the situation [9], in which the capability can be provided. Procedures for defining the context can be found in [15]. From an analysis perspective the identification of the context for a capability are based on questioning the assumptions under which the capability is working. More simply put, the context can be identified by asking questions such as "What external conditions affects the performance of the capability?". "External" in this case means that the organisation should not be able to control the capability context. Typical contexts that are clearly out of control for the organisation are legislation, weather and even to some extent the behaviour of customers or external (IT) systems. Moreover it is of interest to focus on external conditions that affect the capability as a whole, rather than a single instance of a capability. For example, the context in form of legislation may affect all ongoing work and future outcomes of a capability, while a change in how a single customer is sending their orders may only affect a limited part of ongoing work. Based on this discussion we specialize the definition in [9] so that the context of a capability is external information that affects the ability or capacity. This more specialized definition allows an analyst to focus on issues that may go undetected if only examining the regular flow of information that is used in a capability.

Figure 2 contains a small part of the context model for the capabilities in the symbiosis case. In the figure we apply the CDD context model syntax to describe the

context. A *context element* denotes the context, while a *measurable property* denotes a specific way to measure the context. A *context set* is used to group potential ranges of context element values, these ranges are the expected values under which the capability can work. In Fig. 2, it is shown that the resource description capability is affected by the context in form of matching health (a context element), measured by the amount of relative matches in the system (a measurable property).

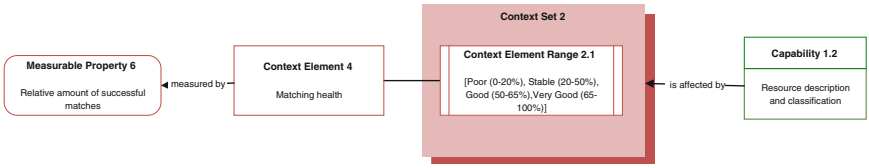

Fig. 2. Excerpt from the CDD model depicting context information

The context model shown in Fig. 2 can be extended by also introducing means to cope with changing contexts. For example, *variation aspects* could be described that each can define how to handle certain context values.

The CDD approach is supported by a set of tools, covering both design time and run-time needs:

- *Capability Design Tool* (CDT): This is the design time environment for creating capability and context models. For example, diagrams such as those shown in Figs. 1 and 2 can be drawn using the CDT.
- *Capability Navigation Application* (CNA). This is a run-time tool used to monitor the context of a capability. Essentially the current values of each context element are shown. The CNA can also signal if the capability needs to be adjusted to accommodate changes in the context.
- *Capability Context Platform* (CCP). This run-time tool work as a message broker that enable the monitoring of complex measurable properties from several context providers. The CCP continuously sends context information to the CNA.

The CDD approach also uses the concept of Capability Delivery Application (CDA) to denote the operational IT support system that a capability have. For example, the CDA can be in the form of an ERP system, or a more specialised system that are a part of a capability. In our case the symbiosis platform is the CDA.

Note that even though these tools are defined in the context of CDD, the concepts of context provider (CCP), context display/navigation (CNA) and operational IT system (CDA) are generic. Thus the approach is generic, even though it is described in terms of CDD.

4 Selecting Design Time Integration

The objective of this step is to decide how to integrate the design time tools of CDD and other tools that is currently in use. The design time tool of CDD is the CDT.

At design time, the components of the CDT environment and existing tools can be combined in several different ways. Likely, the CDT contains models, such as goal models, that may be on a higher level of abstraction compared to the tool currently in use. However, there may also be an overlap in the tools models, meaning that the concepts used during design are the same. In general, tools with this kind of overlap can be integrated in several ways. Depending on the overlap of the tools, and the desired design time functionality we have identified the following three options for performing design-time integration:

a. *Side-by-side*. Consider using the CDT as a side-by-side analysis tool that complements the existing tools. Essentially this means that the CDT and the existing tools are used in parallel, and no integration is performed. On a general level this option is useful if there is no overlap in the concepts used by the tools. This option is suitable when the CDD tools are just used for analysis purposes, and there is no need or desire to directly implement the CDD models in an IT system. The effect of choosing this option is that CDD is used as a side-by-side tool without any integration with the existing tools.
b. *CDT as a starting point* for using the existing tools. Using this option means that the models in CDT are used as input to create the initial solutions in the existing tools. A CDT model can be transformed into a model in an existing Model-driven Development (MDD) tool, or be used to generate code or configurations that can be further edited in existing tools. For example, the context models (Fig. 2) could be the base for creating an information model in a MDD tool, or an initial database schema in a DBMS. This option is useful if there is some form of overlap of the models, since some conceptual resemblance is needed to generate code or models from CDT. The effect of using this option is that initial models, code or configurations will be generated. However, the connection between CDD tools and the existing tools is one-way.
c. *Two-way integration* between CDD and existing tools. This is similar to the option of using the CDD as a starting point. However, here there is also a two-way integration, meaning that when changes are performed in the existing tool they should automatically be propagated to the CDD as well. The effect of using this option is that designers can work in an iterative fashion, performing updates in both the CDD and existing tools.

Table 1 can help in picking the desirable design time integration option. Note that the actual choice can be a combination of the above options, since each type of model in the CDT may use its own form of integration. An application of the options is described below.

4.1 Design Time Options in the Symbiosis Case

In the symbiosis case CLMS use an advanced model driven tool named zAppDev to develop the symbiosis platform. The model driven tool zAppDev contains several types of models, of which some have an overlap with the models used in CDT. For example,

Table 1. Design time integration options.

Option	When to use	Integration effort	Effect
Side-by-side	No or little conceptual overlap between CDD and existing tools	-	CDD is used side-by-side with the existing tools
CDD as starting point	Clear overlap between CDD and existing tools. CDD models commonly more abstract	A model transformation from CDD model to a models, code or configurations in the existing tool is needed	The use of the existing tools is guided by an initial input from the CDD design. The existing tool is used to refine the models
Two way integration	Clear overlap between CDD and existing tools. CDD models and existing tools on the same level of abstraction	Both existing tools and CDD tools need to be extended with synchronisation	The designer can use both CDD and existing tools, models are synchronized both ways

the zAppdev tool employs information models referred to as domain models, and also has the ability to express dynamic behaviour using IDEF0 process models. We here describe two cases of how the CDT and zAppDev tools can be integrated at design time using the *CDD as starting point for existing tools* option.

CDD Context model to zAppDev domain model. Comparing CDD and zAppDev a conceptual overlap can be found the CDD context models and zAppDevs domain models. A domain model in zAppDev depicts the information that the system is processing. By model transformations built into zAppDev the domain models becomes a relational database schema. The context models in CDD contain the concepts used for describing the context of a capability. If desired, the context model could be used as the basis for generating a domain model. By importing the CDD model into zAppDev, a solution engineer can get a draft of the context concepts needed in the domain model.

CDD variation aspect to zAppDev IDEF0 controls. A variation aspect in CDD describes a set of context elements that affect the execution of a capability. In essence, there can be variations of the execution of the capability that are affected by the variation aspect. If a capability is implemented as a process, the variation may be in the form of execution of different control flows in the process. zAppDev does not directly have the concept of variation aspect. However, the IDEF0 diagram in zAppDev has the notion of control flow. A control flow in this case works as a variable that is enabling/disabling a function in the IDEF0. Thus, the control flow could be generated from the CDD variation aspects. This would enable a change in context to enable a function as defined In the IDEF0.

5 Selecting Run-Time Integration

The purpose of this step is to decide how to integrate the run time tools of CDD and the IT system currently supporting the capability under study.

At run time the CDD tools, namely CNA and CCP, can provide a valuable addition to an existing IT system, or a system that is being designed. Using the CDD nomenclature such a system is denoted CDA - Capability Delivery Application. The CDA can be implemented using a model driven tool, or using a traditional development environment. While CDD can be used as a design-time aid, there are additional benefits using it in run time to extend the existing systems. We here distinguish between two options of run time use; as a *monitoring tool* and as an *adjustment tool*. The two options and sub-options are discussed below.

5.1 Monitoring Tool Extension

Using the CPP and CNA as monitoring tools entails making use of the CNA to monitor the context of capabilities. That is, existing system can be extended by allowing the users to monitor the context using the CNA. For this purpose two variants can be selected:

a. *Side-by-side monitoring*. In this option, the CDD is treated as an add-on to existing systems, and no run-time integration is performed. CDD is simply used to monitor how the context changes. For this option to be useful, the context must be defined so that it is not a part of the CDA. Otherwise an integration is needed to extract the context information from the CDA. Typically, side-by-side monitoring can be used for context elements that are affecting several instances of a capability, while context elements that are bound to a single instance are commonly an integral part of a CDA.
b. *Integrated monitoring*. This option can be used when the context information is held within the CDA. Typically the context information is local, that is describes the context of a single capability instance as it is executing. For this option, there is a need to implement a *data provider* in the CDA that sends information to the CCP. Since the CNA are designed for handling global context, there is also a need to perform data aggregation.

Table 2 provides an overview of the monitoring options, and the effect and integration effort they comes with.

5.2 Adjustment Tool Extension

The CDD environment has support for adjusting capabilities to match changes in the context. This is performed by the CNA, that based on current context values can propagate adjustments to the CDA. In an ideal case the adjustments can be designed and implemented, and then used at run time when the context changes. However, this fully automatic handling is not realistic in many cases. For example, some adjustments

Table 2. Summary of run time *monitoring* options

Option	When to use	Integration effort	Effect
Side-by-side monitoring	Context information is not a part of the CDA. Typically the context element is global	-	The CNA is used as a stand-alone tool for monitoring
Integrated monitoring	The CDA hold context information. Typically the context element is local	CNA and CDA need to be integrated, the CDA need to provide context data	CNA can be used to show context information from the CDA system

may need to be manually designed and applied when the context changes. An example of this kind of adjustment is changes needed to comply with new regulations - changed regulations may have a complex impact on the system, which may be impossible to foresee. For each context element there is thus a need to discern if a fully automatic adjustment can be made, or if a manual adjustment is a better option. To guide the adjustment selection, we below describe three options ranging from fully automatic to manual.

a. *Fully automatic*. This option allows the run-time discovery of context changes, and the triggering of adjustment running capabilities to cope with these changes. This requires that (1) the CNA has access to the context element, (2) that the CDA is able to receive information about needed changes and (3) that the CNA is able to notify the CDA when changes should occur. The effect of implementing this option is that the system can perform changes.
b. *Semi-automatic*. In this option, the system can detect changes in the context and suggest adaptations that are manually implemented. For example, this is a useful option if the changes needed are in the form of manual routines. This option requires that (1) the CNA has access to context elements, (2) that manual adaptations are defined. One options for defining manual adaptations is in the form of patterns. For example, in a help-desk there may be two different manual routines for answering questions under a high load compared a low load situation (in this example the current load is considered a part of the context). The effect of using this option is that the system can suggest changes.
c. *Manual*. Some context changes may be difficult to adapt to automatically, or even suggest how to address them. An example of this mentioned earlier is the adaptation to new legislation. However, in this case the context can be monitored with the help of CDD tools, and when a change occurs it can be handled manually. For example, a change in legislation may be monitored in order to start a manual analysis of the impact of the change when it occurs. This option requires no run-time integration with the CDA. The effect of using this option is that the system can discover changes.

Table 3 gives and overview of the effects, integration effort and recommendation of when to use each of the run time adaptations options.

Table 3. Summary of run time adaptation options

Option	When to use	Integration effort	Effect
Fully automatic	It is possible to implement fully automated adaptations that can be triggered at run-time based on context information	The CDA and CNA need to be integrated, the CNA need to be able to receive adaptation advice	The system (CDA + CNA) *detects* the need for, and perform changes
Semi-automatic	It is possible to design a set of adaptations, such as process variants, but these cannot be activated automatically	The adaptation need to be defined, for example as a pattern, but not implemented. CNA and CDA do not need to be integrated	The system suggests changes
Manual	The need for changes can be detected, but the change in itself is too complex to implement	The CNA only needs to monitor the context. No integration with the CDA needed	The system discovers change needs

5.3 Run Time Options in the Symbiosis Case

The i-Symbiosis platform is built using the zAppDev MDD tool and it can be integrated with the CDD runtime tools CCP and CNA. As an example, consider the context as shown in Fig. 2. The measurable property "Relative amount of successful matches" is a part of the CDA constructed by using zAppDev. By using the guidelines in Table 3, we select the monitoring option *integrated monitoring*. The reason for this selection is that the information is held within the CDA. This choice has the implication (see Table 3) that the CDA needs to relay the changes of the measurable property to the CCP, which in turn sends it to the CNA. This is done by implementing a CCP data provider. The CNA is then used to monitor the context element (this means that the CNA will display values from the context element range such as "Poor", "Stable" etc., see Fig. 2).

6 Summarize Tool Support Needed

In this step an overview of the integration need is created, to enable the planning of the integration implementation. Essentially the options selected in step 2 and 3 can be summarized in one large list. This list will include the types of integration options that are needed for a case. If there is a need to perform one single type of integration several times this may be the impetus to implement a generic and reusable solution. For example, there might be several context elements that have been selected for the option *integration monitoring at runtime*. In this case it is beneficial to implement a generic integration that allows the existing development tool to create the needed integrations

for all these elements. On the other hand, if only one context element need *integration monitoring at runtime* it may be more cost-effective to perform the integration without having to change the existing development tool or implement generic integrations.

For a tool vendor the run time and design time options selected will have an impact on the type of integration that the tool need to support.

Table 4 summarizes the integration needs per option. Note that some options do not affect the existing tool at all. For a tool to have a full support of CDD most, or all, of the integration options should be supported.

Table 4. Summary of integration needs

Type	Option	Development tool support needed
Design time	Side-by-side	-
	CDD as starting point for existing tools	A model transformation from CDD model to a model in the tool is needed
	Two way integration	Both existing tools and CDD tools need to be extended with synchronisation
Run-time monitoring	Side-by-side monitoring	-
	Integrated monitoring	The existing tool needs to be able to create a CCP data provider so that the CDA can provide context data
Run-time adaptation	Fully automatic	The existing tool needs to be able to implement an interface such that the CDA can receive adaptation advice from the CDA The existing tool needs to enable the definition of adaptations, and map them into the created interface
	Semi-automatic	–
	Manual	–

7 Conclusions

In this paper we have presented an approach for the combined use of existing systems with tools for capability context design and run-time monitoring. The approach consist of four steps, the first being the analysis of capabilities and their context. Secondly the design-time options are selected such that a potential system analyst and/or developer can work with both context design and system design using existing development tools. The third step includes selecting run-time options, enabling the potential interconnection of context measurements with adaptation in existing systems. The last step involves making an overview of the potential changes that need to be made to existing development tools.

The approach has been used to analyse, and implement IT system support for the case of industrial symbiosis. The implementation shows that it is a viable approach. However possible future work entails applying the approach on more cases.

The approach has been described using the nomenclature of CDD, an approach for capability analysis and design. However, the approach is generic enough to be applied to other methods and tools as well.

Acknowledgments. This work has been performed as part of the EU-FP7 funded project no: 611351 CaaS - Capability as a Service in Digital Enterprises.

References

1. Audia, P.G., Locke, E.A., Smith, K.G.: The paradox of success: an archival and a laboratory study of strategic persistence following radical environmental change. Acad. Manag. J. **43**(5), 837–853 (2000)
2. Wiggins, R.R., Ruefli, T.W.: Schumpeter's ghost: is hypercompetition making the best of times shorter? Strateg. Manag. J. **26**(10), 887–911 (2005)
3. Bērziša, S., et al.: Capability driven development: an approach to designing digital enterprises. Bus. Inf. Syst. Eng. **57**(1), 15–25 (2015)
4. Teece, D.J.: Explicating dynamic capabilities: the nature and microfoundations of (sustainable) enterprise performance. Strateg. Manag. J. **28**(13), 1319–1350 (2007)
5. Schreyögg, G., Kliesch-Eberl, M.: How dynamic can organizational capabilities be? towards a dual-process model of capability dynamization. Strateg. Manag. J. **28**(9), 913–933 (2007)
6. Henkel, M., Bider, I., Perjons, E.: Capability-based business model transformation. In: Iliadis, L., Papazoglou, M., Pohl, K. (eds.) CAiSE Workshops 2014. LNBIP, vol. 178, pp. 88–99. Springer, Heidelberg (2014)
7. Stirna, J., Grabis, J., Henkel, M., Zdravkovic, J.: Capability driven development – an approach to support evolving organizations. In: Sandkuhl, K., Seigerroth, U., Stirna, J. (eds.) PoEM 2012. LNBIP, vol. 134, pp. 117–131. Springer, Heidelberg (2012)
8. Open Group Standard, TOGAF - Enterprise Architecture Methodology, Version 9.1 (2011). http://www.opengroup.org/togaf/. Accessed 07 Mar 2016
9. Dey, A.: Understanding and Using Context. J. Pers. Ubiquit. Comput. Springer **5**(1), 4–7 (2001)
10. Hervás, R., Bravo, J., Fontecha, J.A.: Context model based on ontological languages: a proposal for information visualization. J. Univ. Comput. Sci. **16**(12), 1539–1555 (2010)
11. Moore, P., Hu, B., Wan, J.: Smart-context: a context ontology for pervasive mobile computing. Comput. J. **53**(2), 191–207 (2010)
12. Chaari, T., Ejigu, D., Laforest, F., Scuturici, V.M.: A comprehensive approach to model and use context for adapting applications in pervasive environments. J. Syst. Softw. **80**(12), 1973–1992 (2007)
13. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. Int. J. Ad Hoc Ubiquit. Comput. **2**(4), 263–277 (2007)
14. Vale, S., Hammoudi, S.: COMODE: a framework for the development of context-aware applications in the context of MDE. In: Proceedings of the 2009 4th International Conference on Internet and Web Applications and Services (ICIW 2009), pp. 261–266 (2009)
15. Sandkuhl, K., Koç, H., Stirna, J.: Context-aware business services: technological support for business and IT-alignment. In: Abramowicz, W., Kokkinaki, A. (eds.) BIS 2014 Workshops. LNBIP, vol. 183, pp. 190–201. Springer, Heidelberg (2014)