VU Research Portal

Modeling Context with an Architecture Viewpoint

Bedjeti, Adriatik; Lago, Patricia; Lewis, Grace A.; De Boer, Remco D.; Hilliard, Rich

published in

Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017
2017

DOI (link to publisher)

<https://doi.ieeecomputersociety.org/10.1109/ICSA.2017.26>

document version

Publisher's PDF, also known as Version of record

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Bedjeti, A., Lago, P., Lewis, G. A., De Boer, R. D., & Hilliard, R. (2017). Modeling Context with an Architecture Viewpoint. In *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017* (pp. 117-120). [7930207] Institute of Electrical and Electronics Engineers, Inc..
<https://doi.org/https://doi.ieeecomputersociety.org/10.1109/ICSA.2017.26>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

Modeling Context with an Architecture Viewpoint

Adriatik Bedjeti, Patricia Lago, Grace A. Lewis
Vrije Universiteit Amsterdam
The Netherlands
adriatikbedjeti@gmail.com, {p.lago, g.a.lewis}@vu.nl

Remco C. De Boer
ArchiXL
The Netherlands
rdeboer@archixl.nl

Rich Hilliard
Consulting Software Systems Architect
r.hilliard@computer.org

Abstract—The context of a software system comprises the knowledge that architects need to have about the environment in which a system is expected to operate. Contextual knowledge, however, is often unknown or overlooked. This results in software architects designing systems based on assumptions that are largely unfounded and can potentially lead to system failures. To address this problem, this paper presents a *Context Description Viewpoint* that captures context in software architecture. The viewpoint is based on the results of a literature review that analyzed the state-of-the-art in context, its elements, and modeling techniques. We evaluated and revised the viewpoint by using two case studies based on real-world projects. The case studies showed that the viewpoint is expressive enough to capture context. For software architects it represents a reusable work product to design software systems and to help them identify, capture, and analyze contextual knowledge.

I. INTRODUCTION

The ISO/IEC/IEEE 42010 standard [1] defines software architecture as the *fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles of its design and evolution*. As such, software architectures need to be described and documented properly to be useful to architects, developers, and other stakeholders [2]. At the same time, modern software is increasingly becoming more complex in terms of the number and variety of software components that it depends on. Applications are expected to run in constantly changing environments in terms of their technical, usage, and socio-economic characteristics [3], and need to address ethical, social, legal, security, and economic concerns [4]. However, software systems are conceptualized and modeled based on functional and non-functional requirements, while the external execution environment, dependencies, and context are unknown, taken for granted, or assumed to be or behave in a certain way, which can lead to unfounded assumptions [5][6]. As a result, architectural decisions may not be suitable for the designed system and the actual context that it will operate in.

In spite of numerous studies on context, there is no generic or unifying definition of context. In pervasive computing it refers to the environment of the individual using the application [6][7], in software architecture to the environment of the system [5], and in requirements engineering also to socio-economic aspects [3][6]. The goal of this work is to provide architects with what they need for their job. In summary, we want to (1) identify the context elements that form the context

of a software system, and (2) provide a generic, comprehensive way of modeling the context of a software system.

II. SCOPING CONTEXT

To investigate the state-of-the-art in context identification and modeling, we conducted a systematic literature review following the guidelines in [8], with the goal of discovering models that define and scope context in terms of its elements, regardless of application domain. The results formed the basis for constructing our own context model in software architecture, shown in Figure 1. Full details of the literature review are available in [9]. The leaves of the figure are *context constituents*: pieces of information about the context in which the system operates that can aid design decision making in software architecture.

Fig. 1. Context categories

Platform Context is comprised of the hardware technology a user employs to access an application, the software it runs, and the network capabilities of such technology. Specifically, **Hardware Context** is comprised of the *Platform* entity which defines the device through which the user accesses and uses the application and can be of different types, such as desktop computers, laptops, mobile devices and wearables. *Sensors* can be considered a special hardware element [10] defined as *Platform Accessories*. **Software Context** comprises information about the software elements of the device, such as the operating system or the installed applications/apps [11][12]. Platform connectivity to a network such as the Internet, including connectivity properties (type, bandwidth), comprise the platform network defined as **Network Context**.

User Context relates to the end-user who is expected to use the software. **Identity Context** is the personal information that identifies and characterizes the user such as name, background, and preferences. **Activity Context** represents the

current action the user is engaged in, a particularly important element for applications that monitor and sense users so that they can adapt and/or react to it. **Location Context** represents the place the user is situated in when they use the application and is particularly important for location-based services.

In **Application Context** the *application* is defined as the software that delivers a service to users. Accessed via a device (platform entity), it is essentially the system that is to be designed. From the context perspective, the system-of-interest is to be viewed as a black-box in terms of its software functionality. **External Software Context** refers to the *software components* that are integrated into the system-of-interest but are externally developed, governed or maintained. **External Service Context** refers to *specifications* of external software products and services that the application uses or offers. **Infrastructure Context** consists of *hardware* such as servers and sensors; *software and services* such as the operating system (of the server) and cloud computing services; and *network connectivity* properties. **External System(s) Context** refers to the *software systems* that communicate with the application, either to consume its services or to provide services. **Physical Context** consists of *physical world objects* that the system needs to know about, such as temperature and traffic [11][13]. These objects are monitored by a *sensor* infrastructure and delivered via some communications protocol.

As part of **Organizational Context**, *Organizations* design and develop the application (via architects and engineers), commission it (e.g., clients), or have regulatory power over the system. As such, they can influence the system or its governing organization. Organizations can establish or be subject to **Regulations Context**, i.e. *regulations and guidelines* including laws and standards that affect the system to be developed. **Stakeholder Context** consists of stakeholders who are typically *anyone with an interest in the success of the system* [2], such as business/project managers, owners, architects, developers, users, organizations, and adversaries such as hackers. However, we place the user in its own context category because of its relevance for context modeling. We refer to all other stakeholders as people operating in a certain organization with an interest in the system and may have concerns, roles and goals [14][15]. Organizations have (or lack) *resources* at their disposal, such as financial, operational, staff, or technological competencies [14][16]), to develop the application under established conditions, which form the **Resources Context**.

III. CONTEXT DESCRIPTION VIEWPOINT (CDV)

We define the *Context Description Viewpoint (CDV)* as a standalone, precisely defined and reusable *work product* based on the ISO/IEC/IEEE 42010:2011 standard [1]. The CDV addresses the following concerns:

- **C1: System Scope:** Which constituents are in the span of control of the system (system scope) and which are not (context scope)? Where is the boundary between the system and its context, and what interactions between the system and its context cross this boundary?

- **C2: System Users:** Who are the users of the system; what are their types, roles, and characteristics; and how and where do they access and use the system?
- **C3: External Dependencies:** Which external services and/or applications are relevant for the system, including their properties and providers?
- **C4: Execution Environment:** What is the expected or desired technical execution environment that the system will be running on?
- **C5: Stakeholder Impact:** Which stakeholders, including organizations and their resources, influence the system and in what way? What influence does the system have on organizations and stakeholders?

The CDV targets the **Owners of the system (S1)** who are interested in how the system impacts the involved (external) parties, and are concerned primarily with C1, C3 and C5. The second stakeholders are the **Developers of the system (S2)**, including software architects, developers and testers, who are interested in the context elements likely to influence the system and its functionality. They are concerned with C1 – C4.

According to [1] we define three model kinds. Figure 2 specifies the metamodel for the Scope Model Kind (SMK). It frames concern C1 with the purpose of establishing what lies within the span of control of the system (*System Scope*) and what does not (*Context Scope*). The *Context Constituent* represents instances of the entities depicted in Figures 3 and 4, for which it serves as a super-type (generalization relation). Context constituents can be either part of the System Scope or Context Scope, represented via an association relationship. Two context constituents can be associated if they are within different scopes to denote interactions between the system and its context. UML conventions are used for the notation.

Fig. 2. Metamodel for Scope Model Kind

The User Model Kind (UMK) is specified with the metamodel in Figure 3. Its main purpose is to frame concerns C2 and C4. The main entities are *User*, *Platform* with its associated entities, and *Application* which represents the system-of-interest, meant to be viewed as a black box.

The Environment Model Kind (EMK) is specified according to the metamodel in Figure 4. Its main purpose is to frame concerns C3 – C5. The context constituents involved are: *Application* (External Software Component, External Service Specification, External System, Application Infrastructure, Physical World Object) and *Organization* (Resource, Agent, Regulation). The *Infrastructure* entity is meant to represent the infrastructure of the system-of-interest as a black box where storage and servers, for instance, are not relevant details.

Fig. 3. Metamodel for User Model Kind

Fig. 4. Metamodel for Environment Model Kind

IV. VALIDATION

To evaluate the CDV we designed and conducted two case studies on two real-world projects, *Smart Parking* and *School Enrollment System*. After the first case study the CDV was revised based on the findings and applied to the second. The Smart Parking view based on the SMK of the CDV is presented in Figure 5. It establishes the boundary between the system and its context scope by showing the relationships between the entities that interact across these different scopes.

The view of the Smart Parking system based on the UMK of the CDV is shown in Figure 6. It depicts how the user accesses and uses the application, in terms of technical environment and constraints for use. Figure 7 shows how the School Enrollment System uses the services provided by the external systems for transfer of student data.

We were able to map most of the context elements of the Smart Parking system to context constituents in the CDV, though some were not applicable such as user activity and location. One observation is that an employee plugging the electric vehicle into the charging point is indeed an activity, however, based on the definitions of context constituents in Section II, it is not: the detection of plugging an electric vehicle into the charging point is not a result of monitoring user activities, but rather a device response (of the charging point in this case). This is a small drawback of the CDV because the electric vehicle in this sense is not a platform either, as it does not help the user access the application, but it

Fig. 5. Scope Architecture View based on the Scope Model Kind of the CDV

Fig. 6. User Architecture View based on the User Model Kind of the CDV

Fig. 7. Environment Architecture View based on Environment Model Kind

is needed for the user to carry out his/her actions. Therefore we introduced a new *Constraint* context constituent (pink-shaded elements in Figure 6), which is a physical object or digital enabler that users need to have in order to use the functionality of the system.

Another observation that emerged from the case study is the relationship between some Organization entities and Infrastructure. The charging point (CP) providers provide the CPs to the company. That relationship is captured in the CDV as indicated by the red text in Figures 2 to 4. However,

there was no relationship between the CP Provider and the CPs themselves. To make this explicit, a revision was to include a new association relationship between *Organization* and *Application Infrastructure*, which portrays the relationship between organizations and hardware infrastructure.

Lastly, the Scope Model Kind (SMK) was modified to include a new relationship between Context Constituent entities (cf. Figure 2), and revised concern C1 (pertaining to system-context interactions). This relationship indicates that context constituents can interact with each other and is shown in the SMK if the context constituents are part of different scopes (system or context). The revised SMK for the Smart Parking system is shown in Figure 5. All changes to the CDV after the first case study are indicated in red in Figures 2, 3, and 4. No further changes were made after the second case study.

V. RELATED WORK

There is research on context viewpoints that provides guidelines for framing and constructing architecture views. Rozanki et al. [17] provide a catalog of viewpoints for software system architecture and documentation. While the concerns and stakeholders are similar to the ones defined in our CDV, it only models context as an informal context diagram in terms of system, external entities, and relations among them. The *4+1 view model of architecture* depicts some of the CDV concepts but across multiple views [18]. Musil et al. [19] also define a context viewpoint that adheres to the 42010 standard, but it is a domain-specific viewpoint for *Collective Intelligence Systems* (e.g., content sharing platforms such as wikis). Lastly, Harper et al. [3] aim at defining a context viewpoint in terms of who, what, where, when, how, and why, and propose modeling it in a context viewpoint as an extension to a pre-existing framework, but the context viewpoint itself is not developed and no model is provided. As a result, and to the best of our knowledge, the CDV represents a unique, generic, and reusable artifact for describing, modeling, and reasoning about context.

VI. CONCLUSIONS

This paper addresses the problem of context modeling in software architecture by providing guidelines for software architects in the form of a reusable architecture viewpoint to analyze and capture the context of a software system. Starting with a literature review on context and context modeling in many application domains, we identified four context categories and used them as input to create a list of *Context Constituents*. To model them, we defined a graphical model to represent the context of a software system through a library viewpoint called Context Description Viewpoint, a generic architecture viewpoint based on ISO/IEC/IEEE 42010:2011, which defines three model kinds centered around context scope, user context and execution context. The CDV was evaluated using two case studies and was able to appropriately capture the context of each of these systems. The observations from the case studies were used for its revision and improvement. As future work, we expect that with evaluation of the CDV on more software systems from different application

domains and different architectures than the ones used in the case studies, it can be extended to include new model kinds for additional concerns. Furthermore, more formal guidelines or models could be developed to help architects decide which context constituents are within which scope.

REFERENCES

- [1] International Organization Of Standardization, "ISO/IEC/IEEE 42010:2011 - Systems and software engineering - Architecture description," Tech. Rep. March.
- [2] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*.
- [3] K. E. Harper and J. Zheng, "Exploring Software Architecture Context," in *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*, 2015, pp. 123–126.
- [4] M. Shaw, J. Herbsleb, and I. Ozkaya, "Deciding what to design: closing a gap in software engineering education," *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pp. 28–58, 2005.
- [5] K. Sandkuhl, H. Koc, and J. Stirna, "Capability-as-a-service: Towards context aware business services," in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW*, 2014, pp. 324–332.
- [6] R. Ali, F. Dalpiaz, and P. Giorgini, "A goal-based framework for contextual requirements modeling and analysis," *Requirements Engineering*, no. 4, pp. 439–458.
- [7] C. Jongmyung and M. Hyun-Joo, "Software engineering issues in developing a context-aware exhibition guide system," in *Proc. 9th ACIS Int. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2008 and 2nd Int. Workshop on Advanced Internet Technology and Applications*, 2008, pp. 840–845.
- [8] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. TR/SE-0401, p. 28, 2004.
- [9] A. Bedjeti, "Context modeling in software architecture," Master's thesis, Vrije Universiteit Amsterdam, 2016. [Online]. Available: goo.gl/qwKUXm
- [10] M. Knappmeyer, S. L. Kiani, E. S. Reetz, N. Baker, and R. Tonjes, "Survey of context provisioning middleware," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1492–1519, 2013.
- [11] D. Preuveneers, J. V. D. Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. D. Bosschere, "Towards an Extensible Context Ontology for Ambient Intelligence," *Ambient Intelligence*, vol. 3295, pp. 148–159, 2004.
- [12] R. F. B. De Neto and M. G. C. Da Pimentel, "Toward a domain-independent semantic model for context-aware computing," in *Proceedings - Third Latin American Web Congress, LA-WEB 2005*, vol. 2005, 2005, pp. 61–70.
- [13] D. Ejigu, M. Scuturici, and L. Brunie, "An Ontology-Based Approach to Context Modeling and Reasoning in Pervasive Computing," *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07)*, 2007.
- [14] K. Pohl and P. Haumer, "Modelling Contextual Information about Scenarios," *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ'97*, pp. 187–204, 1997.
- [15] G. Antunes, J. Barateiro, C. Becker, J. Borbinha, and R. Vieira, "Modeling contextual concerns in enterprise architecture," in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, 2011, pp. 3–8.
- [16] P. Petrov, U. Buy, and R. L. Nord, "The need for a multilevel context-aware software architecture analysis and design method with enterprise and system architecture concerns as first class entities," in *Proceedings - 9th Working IEEE/IFIP Conference on Software Architecture, WICSA 2011*, 2011, pp. 147–156.
- [17] N. Rozanski and E. Woods, "Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives," *Viewpoints*, vol. 8, no. 2, p. 576, 2005.
- [18] P. B. Kruchten, "The 4+1 view model of architecture," *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.
- [19] J. Musil, A. Musil, D. Weyns, and S. Biffl, "An Architecture Framework for Collective Intelligence Systems," in *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture - WICSA'15*, 2015, pp. 21–30.