Transition from Process- to Product-Level Perspective for Business Software

Nuno Ferreira, Nuno Santos, Pedro Soares, Ricardo Machado, Dragan Gašević

Transition from Process- to Product-level Perspective for Business Software

Nuno Ferreira[1], Nuno Santos[2], Pedro Soares[2], Ricardo J. Machado[3], and Dragan Gašević[4]

[1]I2S Informática, Sistemas e Serviços S.A., Porto, Portugal
nuno.ferreira@i2s.pt
[2]CCG - Centro de Computação Gráfica, Campus de Azurém, Guimarães, Portugal
{nuno.santos, psoares}@ccg.pt
[3]Centro ALGORITMI, Escola de Engenharia, Universidade do Minho, Guimarães, Portugal
rmac@dsi.uminho.pt
[4]School of Computing and Information Systems, Athabasca University, Canada
dgasevic@acm.org

Abstract. When there are insufficient inputs for a product-level approach to requirements elicitation, a process-level perspective is an alternative way for achieving the intended base requirements. We define a V+V process approach that supports the creation of the intended requirements, beginning in a process-level perspective and evolving to a product-level perspective trough successive models derivation with the purpose of creating context for the implementation teams. The requirements are expressed through models, namely logical architectural models and stereotyped sequence diagrams. Those models alongside with the entire approach are validated using the architecture validation method ARID.

Keywords: Enterprise logical architecture; Information System Requirement Analysis; Design; Model Derivation

1 Introduction

A typical business software development project is coordinated so that the resulting product properly aligns with the business model intended by the leading stakeholders. The business model normally allows for eliciting the requirements by providing the product's required needs. In situations where organizations focused on software development are not capable of properly eliciting requirements for the software product, due to insufficient stakeholder inputs or some uncertainty in defining a proper business model, a process-level requirements elicitation is an alternative approach. The process-level requirements assure that organization's business needs are fulfilled. However, it is absolutely necessary to assure that product-level (IT-related) requirements are perfectly aligned with process-level requirements, and hence, are aligned with the organization's business requirements.

One of the possible representations of an information system is its logical architecture [1], resulting from a process of transforming business-level and technological-level decisions and requirements into a representation (model). It is necessary to

promote an alignment between the logical architecture and other supporting models, like organizational configurations, products, processes, or behaviors. A logical architecture can be considered a view of a system composed of a set of problem-specific abstractions supporting functional requirements [2].

In order to properly support technological requirements that comply with the organization's business requirements, we present in this paper an approach composed by two V-Models [3], the V+V process. The requirements are expressed through logical architectural models and stereotyped sequence diagrams [4] in both a process- and a product-level perspective. The first execution of the V-Model acts in the analysis phase and regards a process-level perspective. The second execution of the V-Model regards a product-level perspective and enables the transition from analysis to design trough the execution of the product-level 4SRS method [5]. Our approach assures a proper compliance between the process- and the product-level requirements through a set of transition steps between the two perspectives.

This paper is structured as follows: section 2 presents the V+V process; section 3 describes the method assessment through ARID; in section 4 we present an overview of the process- to product-level transition; in section 5 we compare our approach with other related works; and in section 6 we present the conclusions.

2 A V+V Process Approach for Information System's Design

At a macro-process level, the development of information systems can be regarded as a cascaded lifecycle, if we consider typical and simplified phases: analysis, design and implementation. We encompass our first V-Model (at process-level) within the analysis phase and the second V-Model (at product-level) in the transition between the analysis and the design. One of the outputs of any of our V-Models is the logical architectural model for the intended system. This diagram is considered a design artifact but the design itself is not restricted to that artifact. We have to execute a V+V process to gather enough information in the form of models (logical architectural model, *B-type* sequence diagrams and others) to deliver, to the implementation teams, the correct specifications for product realization.

Regarding the first V-Model, we refer that it is executed at a process-level perspective. How the term *process* is applied in this approach can lead to inappropriate interpretations. Since the term *process* has different meanings depending on the context, in our process-level approach we acknowledge that: (1) real-world activities of a business software production process are the context for the problem under analysis; (2) in relation to a software model context [6], a software process is composed of a set of activities related to software development, maintenance, project management and quality assurance. For scope definition of our work, and according to the previously exposed acknowledgments, we characterize our process-level perspective by: (1) being related to real-world activities (including business); (2) when related to software, those activities encompass the typical software development lifecycle. Our process-level approach is characterized by using refinement (as one kind of functional decomposition) and integration of system models. Activities and their interface in a process can be structured or arranged in a process architecture [7].

Our V-Model approach (inspired in the "Vee" process model [3]) suggests a roadmap for product design based on business needs elicited in an early analysis phase. The approach requires the identification of business needs and then, by successive artifact derivation, it is possible to transit from a business-level perspective to an IT-level perspective and at the same time, aligns the requirements with the derived IT artifacts. Additionally, inside the analysis phase, this approach assures the transition from the business needs to the requirements elicitation.

In this section, we present our approach, based on successive and specific artifacts generation. In the first V-Model (at the process-level), we use *Organizational Configurations* (OC) [8], *A-type* and *B-type* sequence diagrams [4], (business) *Use Case* models (UCs) and a process-level logical architectural model. The generated artifacts and the alignment between the business needs and the context for product design can be inscribed into this first V-Model.

The presented approach encompasses two V-Models, hereafter referred as the V+V process and depicted in Fig. 1. The first V deals with the process-level perspective and its vertex is supported by the process-level 4SRS method detailed in [9]. The process-level 4SRS method execution results in the creation of a validated architectural model which allows creating context for the product-level requirements elicitation and in the uncovering of hidden requirements for the intended product design. The purpose of the first execution of the V-Model regards eliciting requirements from a high-level business level to create context for product design, that can be considered a business elicitation method (like the Business Modeling discipline of RUP).

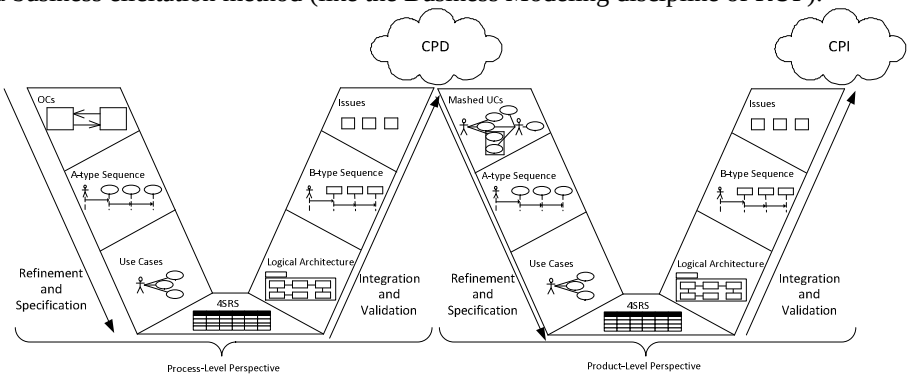

Fig. 1. The V+V process approach

The second execution of the V-Model is done at a product-level perspective and its vertex is supported by the product-level 4SRS method detailed in [5]. The product-level V-Model gathers information from the context for product design (*CPD*) in order to create a new model referred as *Mashed UCs*. Using the information present in the *Mashed UCs* model, we create *A-type* sequence diagrams, detailed in [4]. These diagrams are input for the creation of (software) *Use Case Models* that have associated textual descriptions of the requirements for the intended system. Using the 4SRS method in the vertex, we derive those requirements into a *Logical Architectural model*. Using a process identical to the one used in the process-level V-Model, we create *B-type* sequence diagrams and assess the *Logical Architectural Model*.

The V-Model representation provides a balanced process representation and, simultaneously, ensures that each step is verified before moving into the next. As seen in Fig. 1, the artifacts are generated based on the rationale and in the information existing in previously defined artifacts, i.e., *A-type* diagrams are based on OCs, (business) use case model is based on *A-type* sequence diagrams, the logical architecture is based on the (business) use case model, and *B-type* sequence diagrams comply with the logical architecture. The V-Model also assures validation of artifacts based on previously modeled artifacts (e.g., besides the logical architecture, *B-type* sequence diagrams are validated by *A-type* sequence diagrams). The aim of this manuscript is not to detail the inner execution of the V-Model, nor is it to detail the rules that enable the transition from the process- to the product-level, but rather to present the overall V+V process within the macro-process of information systems development.

In both V-Models, the assessment is made using an adaption of ARID (presented in the next section) and by using *B-type* sequence diagrams to check if the architectural elements present in the *Logical Architectural Model* produced by the models are contained in the scenarios depicted by the *B-type* sequence diagrams.

The first V produces a process-level logical architecture (that can be considered the information system logical architecture); the second V produces a product-level logical architecture (that can be considered the business software logical architecture). Also, for each of the V-Models, in the descending side of the V (left side), models created in succession represent the refinement of requirements and the creation of system specifications. In the ascending side (right side of the V), models represent the integration of the discovered logical parts and their involvement in a cross-side oriented validating effort contributing for the inner-validation for macro-process evolution.

3 V-Model Process Assessment with ARID

In both V-Models execution, the assessments that result from comparing *A-* and *B-type* sequence diagrams produce *Issues* documents. These documents are one of the outputs of the Active Reviews for Intermediate Designs (ARID) method [10, 11] used to assess each V-Model execution. The ARID method is a combination of Architecture Tradeoff Analysis Method (ATAM) [11] with Active Design Review (ADR) [11]. By its turn, ATAM can be seen as an improved version of Software Architecture Analysis Method (SAAM) [11]. These methods are able to conduct reviews regarding architectural decisions, namely on the quality attributes requirements and their alignment and satisfaction degree of specific quality goals. The ADR method targets architectures under development, performing evaluations on parts of the global architecture. Those features made ARID our method of choice regarding the evaluation of the in-progress logical architecture and in the assistance to determine the need of further refinements, improvements, or revisions before assuming that the architecture is ready to be delivered to the teams responsible for implementation. This delivery is called context for product implementation (*CPI)*.

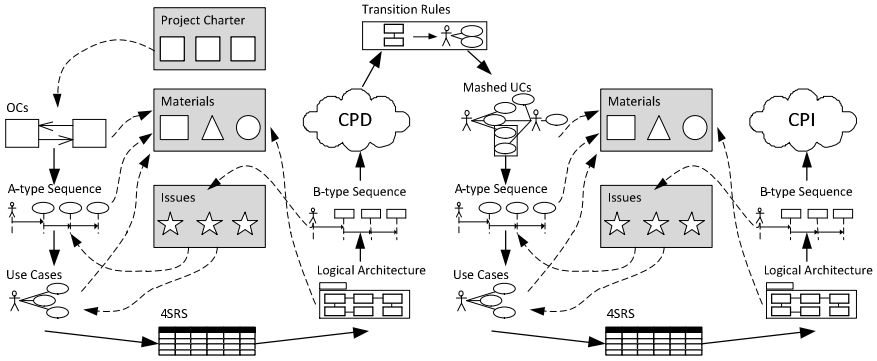

Fig. 2. Assessment of the V+V execution using ARID

In Fig. 2, we present the simplified interactions between the ARID-related models in the V+V process. In this figure, we can see the macro-process associated with both V-Models, the transition from one to the other (later detailed) and the ARID models that support the assessment of the V+V execution.

The *Project Charter* regards information that is necessary for the ongoing project and relates to project management terminology and content [12]. This document encompasses information regarding the project requirements in terms of human and material resources, skills, training, context for the project, stakeholder identification, amongst others. It explicitly contains principles and policies of the intended practice with people from different perspectives in the project (analysis, design, implementation, etc.). It also allows having a common agreement to refer to, if necessary, during the project execution.

The *Materials* document contains the necessary information for creating a presentation of the project. It regards collected seed scenarios based on *OCs (or Mashed UCs), A-type* sequence diagrams and (business or software) *Use Case Models*. Parts of the *Logical Architectural* model are also incorporated in the presentation that will be presented to the stakeholders (including software engineers responsible for implementation). The purpose of this presentation is to enlighten the team about the logical architecture and propose the seed scenarios to discussion and create the *B-type* sequence diagrams based on presented information.

The *Issues* document supports information regarding the evaluation of the presented logical architecture. If the logical architecture is positively assessed, we can assume that we reached consensus to proceed into the macro-process. If not, using the *Issues* document it is possible to promote a new iteration of the corresponding V-Model execution to adjust the previously resulting logical architecture to make the necessary corrections to comply with the seed scenarios. Main causes for this adjustment are: (1) bad decisions that were made in the corresponding 4SRS method execution; (2) *B-type* sequence diagrams not complying with all the *A-type* sequence diagrams; (3) created *B-type* sequence diagrams not comprising the entire *logical architecture*; (4) the need to explicitly placing a design decision in the logical architectural model, usually done by using a common architectural pattern and injecting the necessary information in the use case textual descriptions that are input for the 4SRS.

The adjustment of the logical architectural model (by iterating the same V-Model) suggests the construction of a new use case model or, in the case of a new scenario, the construction of new *A-type* sequence diagrams. The new use case model captures user requirements of the revised system under design. At the same time, through the application of the 4SRS method, it is possible to derive the corresponding logical architectural model.

Our application of common architectural patterns include business, analysis, architectural and design patterns as defined in [13]. By applying them as early as possible in the development (in early analysis and design), it is possible to incorporate business requirements into the logical architectural model and at the same time assure that the resulting model is aligned with the organization needs and also complies with the established non-functional requirements. The design patterns are used when there is a need to detail or refine parts of the logical architecture and, by itself, to promote a new iteration of the V-Model.

In the second V, after being positively assessed by the ARID method, the business software logical architectural model is considered a final design artifact that must be divided into products (applications) for latter implementation by the software teams.

4 Process- to Product-level Transition

As stated before, a process-level V-Model can be executed for business requirements elicitation purposes, followed by a product-level V-Model for defining the software functional requirements. The V+V process is useful for both stakeholders, organizations and technicians, but it is necessary to assure that they properly reflect the same system. In order to assure an aligned transition between the process- and product-level perspectives in the V+V process we propose the execution of a set of transition steps whose execution is required to create the *Mashed UC* model referred in Fig. 1 and in Fig. 2. The detail of the transition rules is subject of future publications.

Like in [2, 14], we propose the usage of the 4SRS by recursive executions with the purpose of deriving a new logical architecture. The transition steps are structured as follows: (1) Architecture Partitioning, where the Process-level Architectural Elements (AEpc's) under analysis are classified by their computation execution context with the purpose of defining software boundaries to be transformed into Product-level (software) Use Cases (UCpt's.); (2) Use Case Transformation, where AEpc's are transformed into software use cases and actors that represent the system under analysis through a set of transition patterns that must be applied as rules; (3) Original Actors Inclusion, where the original actors that were related to the use cases from which the architectural elements of the process-level perspective are derived (in the first V execution) must be included in the representation; (4) where the model is analyzed for redundancies; and (5) Gap Filling; where the necessary information of any requirement that is intended to be part of the design and that is not yet represented, is added, in the form of use cases.

By defining these transition steps, we assure that product-level (software) use cases (UCpt) are aligned with the architectural elements from the process-level logical architectural model (AEpc); i.e., software use case diagrams are reflecting the needs of

the information system logical architecture. The application of these transition rules to all the partitions of an information system logical architecture gives origin to a set of *Mashed UC* models.

5 Comparison with Related Work

An important view considered in our approach regards the architecture. What is architecture? In the literature there is a plethora of definitions but most agree that an architecture concerns both structure and behavior, with a level of abstraction that only regards significant decisions and may be in conformance with an architectural style, is influenced by its stakeholders and the environment where it is intended to be instantiated and also encompasses decisions based on some rationale or method.

It is acknowledged in software engineering that a complete system architecture cannot be represented using a single perspective [15, 16]. Using multiple viewpoints, like logical diagrams, sequence diagrams or other artifacts, contributes to a better representation of the system and, as a consequence, to a better understanding of the system. Our stereotyped usage of sequence diagrams adds more representativeness value to the specific model than, for instance, the presented in Krutchen's 4+1 perspective [16]. This kind of representation also enables testing sequences of system actions that are meaningful at the software architecture level [17]. Additionally, the use of this kind of stereotyped sequence diagrams at the first stage of analysis phase (user requirements modeling and validation) provides a friendlier perspective to most stakeholders, easing them to establish a direct correspondence between what they initially stated as functional requirements and what the model already describes.

6 Conclusions and Outlook

We presented an approach to create context for business software implementation teams in contexts where requirements cannot be properly elicited. Our approach is based on successive models construction and recursive derivation of logical architectures, and makes use of model derivation for creating use cases, based on high-level representations of desired system interactions. The approach assures that validation tasks are performed continuously along the modeling process. It allows for validating: (1) the final software solution according to the initial expressed business requirements; (2) the *B-type* sequence diagrams according to *A-type* sequence diagrams; (3) the logical architectures by traversing it with *B-type* sequence diagrams. These validation tasks, specific to the V-Model, are subject of a future publication.

It is a common fact that domain-specific needs, namely business needs, are a fast changing concern that must be tackled. Process-level architectures must be in a way that potentially changing domain-specific needs are local in the architecture representation. Our proposed V+V process encompasses the derivation of a logical architecture representation that is aligned with domain-specific needs and any change made to those domain-specific needs is reflected in the logical architectural model through successive derivation of the supporting models (OCs, *A*- and *B-type* sequence diagrams, and use cases). Additionally, traceability between those models is built-in by construction, and intrinsically integrated in our V+V process.

Acknowledgments
This work has been supported by project ISOFIN (QREN 2010/013837).

References

1. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. Information Systems (2002)
2. Azevedo, S., Machado, R.J., Muthig, D., Ribeiro, H.: Refinement of Software Product Line Architectures through Recursive Modeling Techniques In: Meersman, R., Herrero, P., Dillon, T. (eds.) On the Move to Meaningful Internet Systems: OTM 2009 Workshops, vol. 5872, pp. 411-422. Springer Berlin / Heidelberg (2009)
3. Haskins, C., Forsberg, K.: Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities; INCOSE-TP-2003-002-03.2. 1. INCOSE (2011)
4. Machado, R., Lassen, K., Oliveira, S., Couto, M., Pinto, P.: Requirements Validation: Execution of UML Models with CPN Tools. International Journal on Software Tools for Technology Transfer (STTT) 9, 353-369 (2007)
5. Machado, R.J., Fernandes, J.M., Monteiro, P., Rodrigues, H.: Transformation of UML Models for Service-Oriented Software Architectures. Proceedings of the 12th IEEE ECBS 2005, pp. 173-182. IEEE Computer Society (2005)
6. Conradi, R., Jaccheri, M.: Process Modelling Languages. Software Process: Principles, Methodology, and Technology, vol. 1500, pp. 27-52. Springer US (1999)
7. Browning, T.R., Eppinger, S.D.: Modeling impacts of process architecture on cost and schedule risk in product development. IEEE Trans on Eng Management 49, 428-442 (2002)
8. Evan, W.M.: Toward a theory of inter-organizational relations. Management Science 217-230 (1965)
9. Ferreira, N., Santos, N., Machado, R.J., Gasevic, D.: Derivation of Process-Oriented Logical Architectures: An Elicitation Approach for Cloud Design. PROFES 2012, vol. LNCS 7343, pp. 44–58. Springer-Verlag, Berlin Heidelberg, Germany Madrid, Spain (2012)
10. Clements, P.C.: Active Reviews for Intermediate Designs., Technical Note CMU/SEI-2000-TN-009. (2000)
11. Clements, P., Kazman, R., Klein, M.: Evaluating software architectures: methods and case studies. Addison-Wesley (2002)
12. Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (2008)
13. Azevedo, S., Machado, R.J., Bragança, A., Ribeiro, H.: Systematic Use of Software Development Patterns through a Multilevel and Multistage Classification. Model-Driven Domain Analysis and Software Development: Architectures and Functions 304 (2010)
14. Machado, R.J., Fernandes, J., Monteiro, P., Rodrigues, H.: Refinement of Software Architectures by Recursive Model Transformations. In: Münch, J., Vierimaa, M. (eds.) Product-Focused Software Process Improvement, vol. 4034, pp. 422-428. Springer Berlin / Heidelberg (2006)
15. Sungwon, K., Yoonseok, C.: Designing logical architectures of software systems. SNPD/SAWN 2005. , pp. 330-337 (2005)
16. Kruchten, P.: The 4+1 View Model of Architecture. IEEE Softw. 12, 42-50 (1995)
17. Bertolino, A., Inverardi, P., Muccini, H.: An explorative journey from architectural tests definition down to code tests execution. Proceedings of the 23rd International Conference on Software Engineering, pp. 211-220. IEEE CS, Toronto, Ontario, Canada (2001)