Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 277–280

ISBN 978-83-60810-22-4

Formal Verification of Business Processes as Role Activity Diagrams

Amelia Bădică* and Costin Bădică†
*University of Craiova, Romania, Email: ameliabd@yahoo.com
†University of Craiova, Romania, Email: costin.badica@software.ucv.ro

***Abstract*—Business process modeling is performed during the requirements analysis and specification of business software systems. Checking qualitative aspects of business processes is required for quality assurance, as well as for compliance with non-functional requirements. We show how business process models represented as Role Activity Diagrams can be formally checked using process algebras and temporal logics.**

I. Introduction

MODERN organizations are process-centered and the truth of the statement "process precedes information" [1] is now widely recognized. This explains why most of the frameworks for organizational modeling and design emphasize the role of processes and process modeling from the high level description of the organization to the underlying IT support systems. Nevertheless, most often, the mapping of business process models to IT applications is defined in an ad-hoc way and the support for managing this mapping is poor. Even if the importance of a separate modeling stage was recognized and a preliminary business process modeling activity is carried out, the resulting model does not have a formal computational semantics and thus it is difficult to map it onto an IT language. On the other hand, the language spoken by IT specialists is too technical, so it is hard to link it up to higher level goals of the organization stated by business analysts.

Consider for example the requirements analysis for an IT system supporting a process-centered business organization. The set of resulting requirements usually contains functional, non-functional, as well as domain specific requirements. According to [2] (i) non-functional requirements may be very hard to verify, as the customers describe them using high-level and informal goals that usually apply to the system or process as a whole, rather than to a specific functionality, and (ii) domain requirements are usually very difficult to understand and specify by non-experts of the application domain. Recent works proposed methods to link non-functional requirements to business process models via goal analysis [3], although the support for automated verification is lacking.

Second, during the last decade, several formal approaches for business process modeling and design were proposed, relying on sound logical approaches. We classified them into (i) lightweight formalizations that use classic first-order logics [4], (ii) heavyweight formalizations based on temporal logics [5], and (iii) hybrid approaches that combine first-order and more advanced (e.g. dynamic or temporal) logics [6].

Third, software engineering provides sound formal modeling and verification techniques for the modeling and verification of software specifications. These methods are now well-supported by model checking technologies [5] and their practical application was improved by introduction of property verification patterns that can be expressed using temporal logics [7]. Recently, the application of formal methods was extended to business processes [8], [9]. A significant step ahead was achieved by introduction of property verification patterns for business process models [10], recently enhanced with visual notations [11], [12]. In particular, model checking can be applied for quality assurance of business processes [13], [9].

We conclude that business process modeling is necessary during the requirements analysis and specification of a business software system for a process-centered organization. Checking qualitative aspects of business processes is required for quality assurance, as well as for compliance with non-functional or domain specific requirements. However, although the software technology that could help to automate this verification exists, the main difficulty is the semantic gap between the languages "spoken" by the business analysts and the IT people. While business analysts are using high-level diagrammatic notations with an intuitive meaning and closer to the business world, IT people are using low-level computational languages that are closer to the computing world. We claim that the missing bit that hinders their clean interaction is the lack of, broadly understood, formal rigor of the current notations used by both business and IT communities.

We propose our contribution for bridging this gap by focusing on formal modeling and verification of business processes represented as Role Activity Diagrams (RAD) [1]. We are aware of the large variety of modeling languages that were proposed for the business and software engineering communities, including the most recent Business Process Modeling Notation (BPMN)[1]. Our chose RAD because: (i) RAD was used for modeling requirements of business processes [12], [14], [15]; (ii) RAD has formal semantics that we introduced in [8]; thus the extension of this work to formal verification is quite natural and straightforward; moreover RAD is at the core of the formalism for knowledge-based modeling of organizations proposed in [6]; (iii) RAD is intuitive and more appropriate for business analysts [1]; (iv) while the focus of other process

[1]http://www.bpmn.org/

978-83-60810-22-4/$25.00 © 2011 IEEE

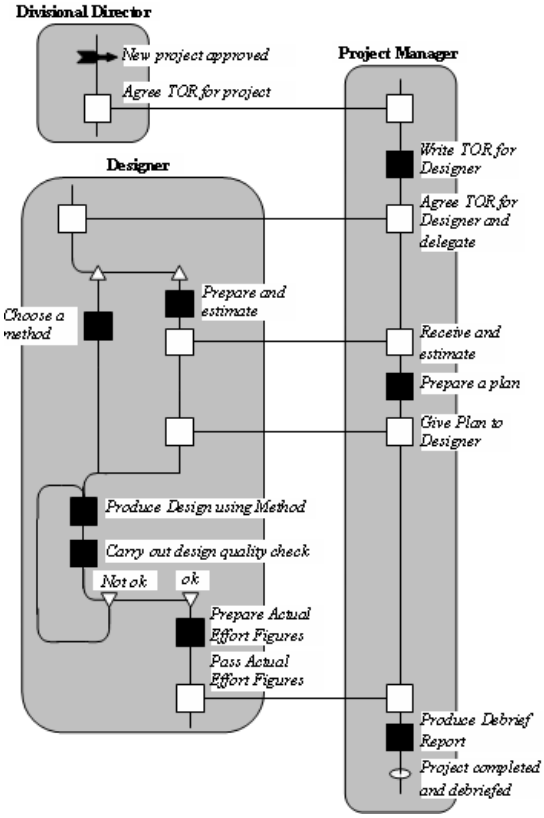

Fig. 1: Sample RAD model for a design project

notations, like UML activity diagrams and BPMN is set on the technical aspects including standardization, interoperability, and integration with software modeling languages, the focus of RAD is set on addressing the high-level needs of a process modeler from business rather than technical perspective.

II. An Example of Formal Verification

RAD is a visual notation for business process modeling [1]. We consider the Finite State Process algebra (FSP) model of a RAD process and show how it can be cheacked against qualitative properties represented using temporal logics. FSP is introduced in [16] and the mapping of RAD to FSP is presented in [8].

Sample Formal Model. Let us consider the RAD model of the design process shown in Figure 1. Its FSP model is shown below. Note that the meaning of the actions used in this FSP model is given in Table I.

$\|DD_0 = SS/\{npa/o\}.$
$\|DD_1 = L/\{npa/i, atp/o\}.$
$\|DD_2 = E/\{atp/i\}.$
$\|DivisionalDirector = (DD_0 \parallel DD_1 \parallel DD_2).$

$\|D_0 = SS/\{atdd/o\}.$
$\|D_1 = L/\{atdd/i, dp/o\}.$
$\|D_2 = L/\{dp/i, cm/o\}.$
$\|D_3 = L/\{dp/i, pe/o\}.$
$\|D_4 = L/\{cm/i, ds/o\}.$
$\|D_5 = L/\{pe/i, re/o\}.$
$\|D_6 = L/\{re/i, gpd/o\}.$
$\|D_7 = L/\{gpd/i, ds/o\}.$
$\|D_8 = L/\{ds/i, pdm/o\}.$
$\|D_9 = L/\{\{nok, pdm\}/i, cdqc/o\}.$
$\|D_{10} = L/\{cdqc/i, \{nok, ok\}/o\}.$
$\|D_{11} = L/\{ok/i, paef/o\}.$
$\|D_{12} = L/\{paef/i, psaef/o\}.$
$\|D_{13} = E/\{psaef/i\}.$
$\|Designer = (D_0 \parallel D_1 \parallel D_2 \parallel \cdots \parallel D_{13}).$

$\|PM_0 = SS/\{atp/o\}.$
$\|PM_1 = L/\{atp/i, wtd/o\}.$
$\|PM_2 = L/\{wtd/i, atdd/o\}.$
$\|PM_3 = L/\{atdd/i, re/o\}.$
$\|PM_4 = L/\{re/i, pp/o\}.$
$\|PM_5 = L/\{pp/i, gpd/o\}.$
$\|PM_6 = L/\{gpd/i, psaef/o\}.$
$\|PM_7 = L/\{psaef/i, pdr/o\}.$
$\|PM_8 = E/\{pdr/i\}.$
$\|ProjectManager = (PM_0 \parallel PM_1 \parallel PM_2 \parallel \cdots \parallel PM_8).$

$\|System = (DivizionalDirector \parallel Designer \parallel ProjectManager).$

Sample Verification. Formal modeling of business processes has the advantage that models can be systematically checked against user-defined properties. A property is defined by a statement that should be true for all the possible execution paths of the process. A property is used to describe a desirable feature of the system behavior. Formal definition of business process properties has the advantage that it enables their concise, rather than speculative analysis.

Properties of software systems are expressed as temporal logic formulas [5]. Temporal logics are used for declarative specification of properties of dynamic systems defined as labeled transition systems, including business processes. A property holds if the associated formula is true for all the possible executions of the system, as it is described by the system model. For system models captured using FSP it was shown that a convenient logic for property specification is *fluent linear temporal logic* (FLTL) [16].

In FLTL primitive properties are expressed using *fluents*. A fluent is a property whose truth is triggered by an initiating event and that holds until the signalling of a terminating event. In FSP it is natural to model initiating and terminating events by execution of specific actions. Every action a defines a singleton fluent $F(a)$ having a as the single initiating action and the rest of all actions as terminating actions. A singleton fluent $F(a)$ is usually written as a in FLTL formulas.

FLTL formulas are built over fluent propositions using the logical operators $\wedge, \vee, \rightarrow, \neg$ and temporal operators $\mathbf{X}$ (next), $\mathbf{U}$ (until), $\mathbf{W}$ (weak until), $\mathbf{F}$ (eventually) and $\mathbf{G}$ (always) [16]. A property P is specified using an FLTL formula Φ.

At the core of a verification task is the activity of property specification. This activity is recognized as very difficult, because on one side it requires special skills in formal specification using temporal logics, while on the other side it requires a good understanding of the target application domain. Nevertheless, some steps have been made in order to help the human modeler to produce specifications of properties by the introduction of specification patterns [7], [10], [11]. However,

TABLE I: Mapping of RAD elements onto FSP actions.

Action	RAD entity name	RAD entity type	Role
npa	*New project approved*	External event	*Divisional Director*
atp	*Agree TOR for project*	Interaction	*Divisional Director*, *Project Manager*
wtd	*Write TOR for Designer*	Activity	*Project Manager*
$atdd$	*Agree TOR for Designer and delegate*	Interaction	*Designer*, *Project Manager*
dp	n/a	Part splitting	*Designer*
cm	*Choose a method*	Activity	*Designer*
pe	*Prepare and estimate*	Activity	*Designer*
re	*Receive and estimate*	Interaction	*Designer*, *Project Manager*
ds	n/a	Parts synchronization point	*Designer*
gpd	*Give Plan to Designer*	Interaction	*Designer*, *Project Manager*
pdm	*Produce Design using Method*	Activity	*Designer*
$cdqc$	*Carry out design quality check*	Activity	*Designer*
$paef$	*Prepare Actual Effort Figures*	Activity	*Designer*
$psaef$	*Pass Actual Effort Figures*	Interaction	*Designer*, *Project Manager*
pp	*Prepare a plan*	Activity	*Designer*
gpd	*Give Plan to Designer*	Interaction	*Designer*, *Project Manager*
nok	*Not ok*	Case refinement	*Designer*
ok	*Ok*	Case refinement	*Designer*
pcd	Project completed and debriefed	State description	*Project Manager*

in our opinion even with the availability of visual specification patterns [12] we are still far from bridging the gap between requirements analysis and verification, as pointed out in the introduction of this paper. Nevertheless, we found very useful the use of patterns to specify simple properties for the process example considered in this paper.

In our case we considered two sample properties:

P1 Each project approved must be eventually completed and debriefed.
P2 Each project cannot be completed and debriefed without being approved by a design quality check.

Based on a rigorous analysis of the business domain, [10] proposed four classes of property specification patterns for business process models; tracing, consequence, combined occurrence, and precedence. Those two properties were formalized using consequence and precedence patterns by reformulating them as follows:

P1 The action "New project approved" leads to reaching the state "Project completed and debriefed".
P2 The state "Project completed and debriefed" requires the action "Carry out design quality check" to return a positive response.

Their formal description using FLTL is as follows:

$$\textbf{assert } P1 = \mathbf{G}\,(npa \rightarrow \mathbf{F}\,pcd)$$
$$\textbf{assert } P2 = ((\neg pcd\,\mathbf{U}\,ok) \vee \mathbf{G}\,\neg pcd)$$

We have checked the sample process model against these two properties with the help of Labelled Transition System Analyser (version 3.0)[2]. Both P1 and P2 were found as not violated by the process model. However, the analysis of P2 revealed a deadlock that can be explained as follows. A property check assumes three steps: (i) construction of a new process coresponding to the given property; (ii) computation of the parallel composition of the original process with the property process; (iii) performing a graph analysis of the resulting parallel composition. In our case, the resulting parallel composition has a deadlock because the property P2 does not explicitly check that action pcd actually occurs, while in the original process this action will always eventually occur.

This simple experiment revealed a number of difficulties with the application of formal specification to requirements verification of business process models.

- The formal business process model is very difficult to understand and maintain. Without a proper management of the links between the formal model and the initial RAD model, it is impossible to manage large and complex process models.
- The formalization of requirements is very difficult to realize, even with the availability of property specification patterns. Additional support, beyond patterns, is needed to better manage the requirements and their mapping to formal properties.
- The results of the verification process are very difficult to interpret. Special support to link them back to the RAD model, as well as to explain them to the human modeler is lacking.

III. Conclusions

We introduced a method for checking qualitative properties of business processes represented as RADs using the formal specification languages of process algebras and temporal logics. We presented our initial analysis of the problems encountered during the application of formal verification for checking requirements of business processes. In our opinion the core technologies already exist. However, the necessary links between them are lacking. We think that the main cause is the gap between the languages spoken by the business and computing communities. We plan to address this issue as medium term future work. In the short term we plan to enhance our results by considering more complex processes and properties.

[2]http://www.doc.ic.ac.uk/~jnm/book/ltsa/download.html

ACKNOWLEDGMENT

The work reported here was partly supported by (i) the research project "SCIPA: Servicii software semantice de Colaborare si Interoperabilitate pentru realizarea Proceselor Adaptive de business" with the National Authority for Scientific Research, Romania and partly by (ii) the research project "Agent-Based Service Negotiation in Computational Grids" between Systems Research Institute, Polish Academy of Sciences, Poland and Software Engineering Department, University of Craiova, Romania.

REFERENCES

[1] M. A. Ould, *Business Process Management: A Rigorous Approach*. British Computer Society, 2005.

[2] I. Sommerville, *Software Engineering, 9/E*. Addison-Wesley, 2011.

[3] F. Aburub, M. Odeh, and I. Beeson, "Modelling non-functional requirements of business processes," *Information and Software Technologies*, vol. 49, no. 11-12, pp. 1162–1171, 2007.

[4] Y.-H. Chen-Burger and D. Robertson, *Automating Business Modelling*. Springer, 2004.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. The MIT Press, 1999.

[6] M. Koubarakis and D. Plexousakis, "A formal framework for business process modelling and design," *Information Systems*, vol. 27, no. 5, pp. 299–319, 2002.

[7] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proc.*21^{st} *international conference on Software engineering (ICSE 1999)*. IEEE Computer Society Press, 1999, pp. 411–420.

[8] A. Bădică, C. Bădică, and V. Liţoiu, "Role activity diagrams as finite state processes," in *Proc.*2^{nd} *International Symposium on Parallel and Distributed Computing (ISPDC 2003)*. IEEE Computer Society, 2003, pp. 15–22.

[9] B. B. Anderson, J. V. Hansen, P. B. Lowry, and S. L. Summers, "Model checking for e-business control and assurance," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, pp. 445–450, 2005.

[10] W. Janssen, R. Mateescu, S. Mauw, P. Fennema, and P. v. d. Stappen, "Model checking for managers," in *Proceedings of the 5th and 6th International SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking*, ser. Lecture Notes in Computer Science, vol. 1680. Springer-Verlag, 1999, pp. 92–107.

[11] A. Forster, G. Engels, T. Schattkowsky, and R. Van Der Straeten, "Verification of business process quality constraints based on visual process patterns," in *Proceedings of the First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering*. IEEE Computer Society, 2007, pp. 197–208.

[12] A. Awad, M. Weidlich, and M. Weske, "Visually specifying compliance rules and explaining their violations for business processes," *Journal of Visual Languages and Computing*, vol. 22, no. 1, pp. 30–55, 2011.

[13] W. Wang, Z. Hidvégi, A. D. Bailey, and A. B. Whinston, "E-process design and assurance using model checking," *Computer*, vol. 33, no. 10, pp. 48–53, 2000.

[14] N. V. Patel, "Healthcare modelling through role activity diagrams for process-based information systems development," *Requirements Engineering*, vol. 5, no. 2, pp. 83–92, 2000.

[15] S. Bleistein, K. Cox, J. Verner, and K. Phalp, "Requirements engineering for e-business advantage," *Requirements Engineering*, vol. 11, no. 1, pp. 4–16, 2006.

[16] J. Magee and J. Kramer, *Concurrency. State Models and Java Programs, 2/E*. John Wiley & Sons, 2006.