Validation of Reactive Software from Scenario-Based Models *

Óscar R. Ribeiro João M. Fernandes
Dep. Informática / CCTC, Universidade do Minho, Braga, Portugal
{orribeiro, jmf}@di.uminho.pt

Abstract

This thesis proposal suggests a model-based approach to obtain, from a set of behavioural scenarios of a given reactive software system, a graphical animation for reproducing that set of scenarios for validation purposes.

The approach assumes that the requirements of the system are described by a use case diagram, being the behaviour of each use case detailed by a collection of scenario descriptions. These use cases and scenarios are transformed into a Coloured Petri Net (CPN) model, which is next complemented with animation-specific elements.

By executing the CPN model, it is possible to animate the scenarios in a user-friendly way and thus ensuring an effective involvement of the users in the system's validation.

The CPN model is enforced to be (1) parametric, allowing an easy modification of the initial conditions of the scenarios, (2) environment-descriptive, meaning that it includes the state of the relevant elements of the environment, and (3) animation-separated, implying that the elements related to animation are clearly separated from the other ones.

We validate our approach based on its application to two examples of reactive systems.

1 Introduction

A reactive system "is a system that, when switched on, is able to create desired effects in its environment by enabling, enforcing or preventing events in the environment" [18]. In contrast, a transformational system computes the output from the input and then terminates. When developing a reactive system, which typically has an intensive behaviour and a rich set of interactions with its environment, requirements validation, before any design and implementation decisions are taken, is an important task.

Validation consists on checking if a model or a system satisfies the user expectations. During the development of software systems, validation is a crucial activity to allow developers to be confident that they are building the right system. One of the key issues to have a successful validation is to adopt a process where users can actively discuss the requirements of the system under development.

To model software systems the Unified Modelling Language (UML) is the standard notation used nowadays in industry. In this work, we adopt two UML diagrams: Use Case Diagrams and Sequence Diagrams. Use cases specify the set of functionalities presented by a system, and permit, due their simplicity, the dialogue between clients and developers. A sequence diagram is used to capture a behavioural scenario of a given system, which can be seen as a sequence of steps describing interactions between the actors and that system. In this work we consider that each use case is described by a set of sequence diagrams.

The main goal of this work is to introduce novel methods into the software development process to create a graphical user-intuitive animation of the problem domain from a set of scenario descriptions. In particular, we aim to study how to translate models of behavioural scenarios, into a state-based model that represents the global behaviour of the system under development. The resulting model, which in this work we propose to be written in the CPN modelling language [9], is used to coordinate an animation of the problem domain, in order to facilitate the validation activity.

CPNs constitute a graphical modelling language appropriate to describe the behaviour of systems with characteristics like concurrency, resource sharing, and synchronization. The CPN Tools [10, 19] is a well established tool supporting the CPN modelling language and allowing the execution of animations in accordance with the CPN model.

In order to validate our approach, it is applied to two examples of reactive systems, namely an elevator controller and an automatic gas pump station.

This paper is structured as follows. Section 2 presents a review of the literature about the main topics of this work. In section 3, the research objectives are detailed and the adopted approach is described. Section 4 introduces the current state of the work. Section 5 shows the work plan we are following. The conclusions are presented in Section 6.

*This work has been supported by the grant with reference SFRH/BD/19718/2004 from "Fundação para a Ciência e Tecnologia".

2 State-of-the-Art

The focus of our work is the transformation of models of behavioural scenarios into state-based models. In particular we are interested in the translation of sequence diagrams into CPNs. Next we describe several approaches which were already proposed to combine the usage of these two types of models.

Krüger et al. suggest the usage of Message Sequence Charts (MSCs) for scenario-based specifications of component behaviour, especially during the requirements capture phase of the software process [13]. They discuss how to schematically derive statecharts from MSCs, in order to have a seamless development process.

Harel and Marelly propose the usage of scenario-based programming, through UML use cases and play-in scenarios [7]. Harel's play-in scenarios make it possible to go from a high-level user-friendly requirements capture method, via a rich language for describing message sequencing, to a full model of the system, and from there to the final implementation.

Whittle and Schumann propose an algorithm to automatically generate UML statecharts from a set of UML sequence diagrams [17]. This work also presents the usage of the algorithm for a real application. Their main conclusion is that it is possible to generate code mostly in an automatic way from scenario-based specifications.

Hinchey et al. propose an approach, called *requirements to design to code*, where designers write requirements as scenarios in constrained (domain-specific) natural language [8]. Other notations are however also possible, including UML use cases. Based on the requirements, an equivalent formal model, using CSP, is derived, which is then used as a basis for code generation.

Uchitel and Kramer present an MSC language with semantics in terms of labeled transition systems and parallel composition [15]. The language integrates other languages based on the usage of high-level MSCs and on the identification of component states. With their language, scenario specifications can be broken up into manageable parts using high-level MCSs. These authors also present an algorithm that translates scenarios into a specification in the form of Finite Sequential Processes, which can be used for model checking and animation purposes.

There are also works on the synthesis of Petri nets from scenario-based specification. Juhás et al. present a polynomial algorithm to decide if a scenario, specified as a Labelled Partial Order, is executable in a given Place/Transition Petri net [12]. The algorithm preserves the given amount of concurrency and does not add causality. In case the scenario is indeed executable in the Petri net, the algorithm computes a process net that respects the concurrency expressed by the scenario. Although quite useful, this technique is not yet available for high-level Petri nets (such as Object-oriented Petri nets, CPNs, or Reference nets), and to validate the scenario the user must simulate the obtained process net, where the concepts of the problem domain are not clearly represent as in the created animations used in our approach.

Amorim et al. introduce an informal methodology to map Live Sequence Charts (LSCs) into CPNs for allowing properties of the system to be verified and analysed [2]. They do not consider the validation of gathered behavioural scenarios, but only its verification, namely to detect some inconsistencies between them.

Eichner et al. present a formal semantics by means of Petri nets for the majority of the concepts of sequence diagrams [4]. This semantics allows the concurrent behaviour of the diagrams to be modelled and subsequently analysed. Moreover, the usage of high-level Petri nets with data representation in its tokens permits an efficient structure for data types and control elements. In their approach they use places to represent the messages, instead of transitions as we do.

Elkoutbi and Keller suggest the usage of use case diagrams diagrams and scenarios to obtain one hierarchical CPN to model the behaviour of an interactive system [5]. The hierarchy of the CPN mimics the one of the use case diagram diagram. The usage of the colours in the nets preserves the independence of several scenarios after their integration in to the CPN. This permits modeling of concurrency between use cases, scenarios and copies of the same scenario. However, their approach only tackles the controller perspective, and does not include the environment parts.

Dano et al. suggest use cases to be collected and described by tables, to facilitate the communication between the analyst and the domain expert [3]. Later, through some mapping rules, Petri Nets are built from the tables to formalise the requirements. The approach is used for producing object-oriented requirements specifications, based on structural models and focuses on deriving intra-object behavioural models. Again, the environment part is not modelled.

3 Research Objectives and Approach

In this section we present the details of the research objectives of this project and also a description of the approach to be followed.

The research approach taken in this project begins with the study of the state of the art, in order to clearly describe the research problem. After that, we verify the hypothesis in the thesis finding solutions for the problem. To validate the obtained solutions we explore two different case studies already described in the literature. During the exploration

of the case studies we consider the possibility to introduce improvements to the solution. We report the application of these solutions to the case studies, and the results will be submitted to conferences in the area.

The focus of our work is in the translation of models of behavioural scenarios into a state-based model. We consider that the requirements document includes a set of use case diagrams and its associated descriptions. If these artefacts are not explicitly available we assume that the document has sufficient information to develop them. The behaviour of each use case is detailed by a collection of scenario descriptions, which can be represented by sequence diagrams. Recently the version 2.0 of UML has been launched to substitute the previous versions. Sequence diagrams in UML 2.0 have many new high-level flow operators. Due to their simplicity we believe that sequence diagrams of UML1.x are more adequate to capture the system's behavioural scenarios at a first stage of analysis. Then, in the beginning we do not use all the features available in UML 2.0 (namely the high-level operators) and later in the development process, these diagrams can be aggregated into UML 2.0 sequence diagrams with all its features.

Most of the works on the translation of scenario-based models into state-based models concentrate only on the modelling of the controller part, and do not take into account the environment [5, 3], which is an important part, especially when considering reactive systems. The obtained model usually is not parametric, i. e., it does not permit the simulation of scenarios obtained by changing some initial conditions of the original scenario.

We want to explore the capabilities of the CPN modelling language in order to have a state-based and executable model with the following additional characteristics: (1) It allows an easy modification of the initial conditions of the scenario (parametric), (2) it includes the state of the relevant elements of the environment (environment-descriptive), and (3) the elements related to animation are clearly separated from the other elements in the model (animation-separated).

Our approach is based on the translation of models of behavioural scenarios into a CPN model. We believe that the CPN modelling language is adequate to reach our goals because it was already used on the modelling and validation of problems from a wide range of areas [9], and the edition, simulation and analysis of CPNs are supported by the CPN Tools, a well proven tool. The CPN modelling language is a formal language supporting concurrency. To animate a CPN model, the BRITNeY Suite Animation Tool [16] permits the creation of an animation on top of CPN Tools, using the animation plug-in based on the SceneBeans framework [1].

Fig. 1 sketches the general software process proposed to be followed in our approach, where rectangles represent artefacts and arrows represent activities. The idea is that from the requirements document, it is possible to obtain a

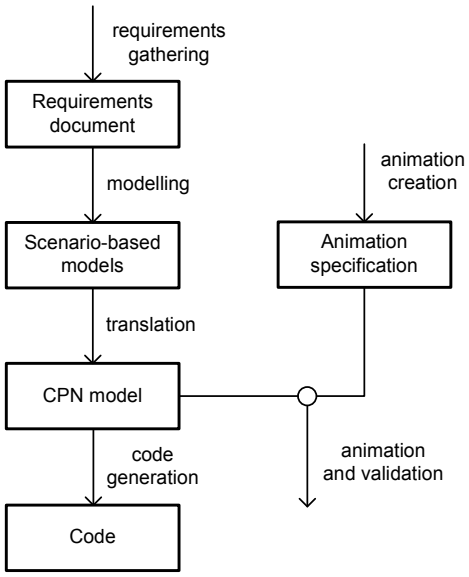

Figure 1. Process for the approach.

set of scenario-based models, which can be subsequently translated into a CPN model. This CPN model, when used in conjunction with an animation specification, drives an animation that permits the users to perceive how the system behaves and to validate that behaviour with respect to the requirements.

The animation creation is an activity consisting on defining how each element in the problem domain is represented in the animation, namely how it is graphically depicted, the movements associated to it, the messages to be sent, and the commands it receives.

The CPN model can be used as a formal support in the next steps of the development process. In particular we can generate implementation code from it.

Our work concentrates on two activities of the approach: translation and animation creation (Fig. 1). It aims to discover how the translation from a set of scenario-based models to a CPN model can be automated, and if it is possible to clearly separate the CPN model and the animation specification.

In order to increase the flexibility level of the animation's usage we aim to permit that the user can change the initial conditions of the scenario being animated, obtaining a variation of the scenario. The obtained CPN model must permits also the concurrent execution of various scenarios.

The approach will be validated through the exploration of two examples of reactive systems. During the exploration of these systems a conceptual tool will be developed to support others applications in different examples and to be used as a formalization of the translation.

4 Current Work and Preliminary Results

The research project described in this proposal started by defining the necessary rules to transform behavioural scenarios into a CPN model [14]. For these first experiments we had considered a small control application from the industry. In order to establish these rules, we apply them to two case studies, namely an elevator system and a gas pump station. In [6], we present an analysis of the elevator system, based on the rules previously defined, where we improve the rules of translation in order permit the concurrent execution of different scenarios.

By now, we are defining ways to clarify the separation, in the obtained CPN model, between the definitions related to the definitions about the system being analysed, and the animation specific elements introduced in CPN model. With this animation-separated CPN-model, after the validation of the system, it is possible to reuse the CPN model related to the control part as a formal basis for the next development steps. For example, the CPN model can be used for generation of implementation code for the system being developed. We also plan to define mechanisms to have a parametric CPN model that permits the change of the initial conditions of a given scenario, thus obtaining a different version of the scenario being considered. We will explore another reactive system, an automated gas pump station, which is a system that permits customers to buy fuel in a self-served way. We plan to create a requirements document modelling all the behaviours as a scenario, and produce an animation of the system's problem domain. In this example we have more types of external actors, than in the elevator system, in this way we believe that the exploration of this example will complement the definition of translation, namely in what concerns to the treatment given to the external actors in the animation.

5 Work Plan and Implications

In this section we describe in a chronological way the activities of our work plan, which have been done, and some activities that we plan to do in the next steps of this research project.

In the first semester, we have studied in the main areas to be covered in this work: reactive systems, validation of software systems, system modelling with CPNs, UML, and scenario-based behavioural models. From this study we have produced a synthesis about the state-of-the-art in the area. As a practical exercise, we have constructed some animations of the behaviours described in the requirements document, using the CPN Tools together with the BRITNeY Suite Tool, applying informal rules to translate a behavioural requirement to the target CPN model.

In the second semester, we have studied some works about the translation of scenario-based models into state-based models, and we have started the definition of some rules to translate sequence UML 2.0 diagrams into a CPN model. In a first stage, we considered a subset of the high-level operators present in the UML 2.0 sequence diagrams.

We have explored a small control application, defining the sequence diagrams representing its behavioural scenarios, and studying the applicability of the defined rules to the system being considered. Using the rules for translation we created a CPN model, and on top of it we created an animation of the system. We produced a technical report where these experiences are detailed.

In the third semester, we submitted first results of applying the ideas into an article at a workshop organized by people responsible for the development of CPN modelling language, and its supporting tools. We received feedback about the way we were using the CPN modelling language in our approach. Then, some improvements were done over the rules of translation, and they were extended to the rest of high-level operators present in the language of UML 2.0 sequence diagrams.

We explored an elevator controller system, which is responsible for controlling two cars in a building with six floors. By exploring this system we discovered the necessity to put some mechanisms in the target CPN model, in order to permit the representation of the variations of a use case, and to permit the concurrent execution of various scenarios. We produced from this work a technical report where we present the behavioural models of scenarios related to the elevator system, and all the details of the experience of applying these ideas to the elevator system. We also published also a paper about the problem of combining scenarios and state machines [6].

In the fourth semester, we developed an animation to the elevator system and some applications to facilitate the management of the animation's definitions to be used as the input to the SceneBeans Suite Tool. We started the development of a conceptual tool to support the application of the translation rules. To do this, we stated a modelling language to capture behavioural scenarios based essentially on the sequence diagrams of UML, but probably it will be necessary to include some annotations to facilitate the automatic generation of the CPN model. We plan to use the VDM specification language [11] to formalize this translation.

In the fifth semester, we plan to write an article about the usage of animations in the requirements engineering area, in order to validate the usefulness of the properties that we are introducing in the obtained CPN model. Additionally we continue to improve the tool that supports the translation.

For the sixth semester, we plan to mature the results obtained from exploration of the considered examples, and from the development of the supporting tools for the translation. We also plan to do the final adjustments to finish the final thesis document to be submitted for evaluation.

6 Conclusions

In this proposal we describe a Ph.D. thesis research project that aims to present novel methods to be applied on the validation of reactive systems, namely through its animation with a language using problem-domain concepts.

The focus of our work is on the translation of behavioural scenario models into a state-based model, which is intended to represent the global behaviour of the system according to the requirements. In particular, our approach considers the translation of use cases described by a set of behavioural scenarios into a CPN model that is parametric, environment-descriptive and animation-separated. The CPN is used to drive an animation of the problem domain. We plan to study what parts of the translation process can be automatized, and to create tool-support for that process.

The validation of the approach will be obtained by applying its ideas, concepts, and methods to two examples.

References

[1] SceneBeans. www-dse.doc.ic.ac.uk/software/SceneBeans/.

[2] L. Amorim, P. Maciel, M. Nogueira, R. Barreto, and E. Tavares. A Methodology for Mapping Live Sequence Chart to Coloured Petri Net. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 4, pages 2999–3004, Oct 2005.

[3] B. Dano, H. Briand, and F. Barbier. An Approach Based on the Concept of Use Case to Produce Dynamic Object-Oriented Specifications. In *3rd IEEE Int. Symp. on Requirements Engineering (RE '97)*, pages 54–64. IEEE CS Press, 1997.

[4] C. Eichner, H. Fleischhack, R. Meyer, U. Schrimpf, and C. Stehno. Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets. In *12th Int. SDL Forum*, volume 3530 of *LNCS*, pages 133–48. Springer, Jan 2005.

[5] M. Elkoutbi and R. Keller. Modeling Interactive Systems with Hierarchical Colored Petri Nets. In *Advanced Simulation Technologies Conference 1998*, pages 432–37, 1998.

[6] J. M. Fernandes, S. Tjell, J. Jørgensen, and O. R. Ribeiro. Designing Tool Support for Translating Use Cases and UML 2.0 Sequence Diagrams into a Coloured Petri Net. In *6th Int. Workshop on Scenarios and State Machines (SCESM 2007), at ICSE 2007*. IEEE CS Press, 2007.

[7] D. Harel and R. Marelly. *Come, Let's Play: Scenario-based Programming Using LSCs and the Play-Engine*. Springer, 2003.

[8] M. G. Hinchey, J. L. Rash, and C. A. Rouff. A Formal Approach to Requirements-Based Programming. In *Proceedings of the 12th IEEE Int. Conf. and Workshops on the Engineering of Computer-Based Systems (ECBS 2005)*, pages 339–45. IEEE CS Press, 2005.

[9] K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Volume 1-3*. Monographs in Theoretical Computer Science. EATCS Series. Springer, 1992-97.

[10] K. Jensen, L. M. Kristensen, and L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *Int. Journal on Software Tools for Technology Transfer (STTT)*, 9(3-4):213–54, June 2007.

[11] C. B. Jones. *Systematic software development using VDM*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1990.

[12] G. Juhás, R. Lorenz, and J. Desel. Can I Execute My Scenario in Your Net? In *Proceedings of 26th Int. Conf. Applications and Theory of Petri Nets*, volume 3536 of *LNCS*, pages 289–309, Miami, USA, June 2005. Springer.

[13] I. Krüger, R. Grosu, P. Scholz, and M. Broy. From MSCs to Statecharts. In F. J. Rammig, editor, *Distributed and Parallel Embedded Systems*, pages 61 – 71. Kluwer Academic Publishers, 1999.

[14] O. R. Ribeiro and J. M. Fernandes. Some Rules to Transform Sequence Diagrams into Coloured Petri Nets. In *7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN 2006)*, pages 237–56, 2006.

[15] S. Uchitel, J. Kramer, and J. Magee. Synthesis of Behaviour Models from Scenarios. *IEEE Transactions on Software Engineering*, 29(2):99–115, Feb. 2003.

[16] M. Westergaard and K. B. Lassen. The BRITNeY Suite Animation Tool. In *27th Int. Conf. on Applications and Theory of Petri Nets*, pages 431–40, 2006.

[17] J. Whittle, R. Kwan, and J. Saboo. From scenarios to code: An air traffic control case study. *Software and Systems Modeling*, 4(1):71 – 93, Feb 2005.

[18] R. J. Wieringa. *Design Methods for Reactive Systems: Yourdon, Statemate, and the UML*. Morgan Kaufmann, 2003.

[19] CPN Tools. www.daimi.au.dk/CPNtools.