Using partial model synthesis to support model integration in large-scale software development

Marsha Chechik and Rick Salay

Department of Computer Science, University of Toronto, Toronto, Canada
{chechik, rsalay}@cs.toronto.edu

Abstract. Global software development projects typically require the integration of models from multiple stakeholders. However, the need for integration creates dependencies between modelers that can hamper progress when the models are developed according to different schedules. For example, when a required model is not available or only partially available, the model user is limited in how to proceed with integration efforts. The use of interface contracts improves the situation but interfaces cannot be used for some kinds of analysis such as simulation. To address this, in this paper we propose the use of model synthesis to automatically create "model stubs" that approximate the behaviour of an unknown model according to its interface specification. Such model stubs can be used temporarily in place of the required model to facilitate integration and permit testing and simulation.

1 Introduction

Global software engineering projects are typically highly distributed and involve many stakeholders producing heterogeneous models reflecting different concerns. In such a context, the GEMOC initiative seeks to find ways to support collaboration and interoperability at the integration points between models. Thus, a key GEMOC challenge is management of dependencies between models so that the project can move forward efficiently without sacrificing correctness at these points of integration. A common strategy to address this challenge is to clearly define interface specifications early in the development lifecycle. Interface specifications can range from a set of operation signatures to detailed interaction scenarios, properties, invariants, etc. expressed in appropriate specification languages. However, while such specifications are useful for guiding model development and doing static analysis, they are not well suited to dynamic techniques such as testing, simulation and model checking.

In this paper, we propose to address this gap by automatically synthesizing *partial* models from interface specifications that act as model "stubs" and can be used as a temporary stand-in for the model to allow dynamic analysis. A partial model represents the set of possible models that is admissible by the specification. Thus, the stub approximates the actual model "up to" the specification - i.e., the more detailed the specification is, the better the approximation will be.

Our objective in this paper is to present a simple, but non-trivial, example of how partial models can be created and used. For this, we use existing work on synthesizing a type of behavioural partial model called Modal Transition Systems (MTS) from heterogeneous interface specifications [6].

The paper is structured as follows. In Section 2, we introduce an example development project. In Section 3, we show the result of applying the MTS synthesis method of [6] to this example and discuss its uses. Section 4 describes dynamic analysis with MTSs. We conclude in Section 5 with a discussion of a research agenda for developing the approach to accommodate a wider variety of models in order to address the broader vision of GEMOC.

2 Example

As an example (presented in [6]), consider development of a complex communication system that allows users to interact with each other through a communication console in a variety of ways, including via web-based email (webmail). Assume the webmail component is being developed by a third party provider and they have been given an interface specification as shown in Fig. 1.

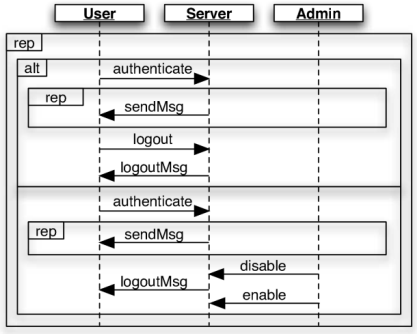
The specification consists of particular scenarios expressed as sequence diagrams (in this case, just one) and a set of system properties that must hold expressed using Fluent Linear Temporal Logic (FLTL). The sequence diagram shows a repetition of two possible behaviours – either the user authenticates and receives messages from the server until she logs out or until an administrator forces her to log out using a disable message. The FLTL properties enforce certain natural requirements of the webmail component. For example, p_1: at all times, if the user is logged in then she must be registered; p_2: at all times, if a message is sent to the user, the user must be logged in; and, p_3: at all times, if the user logs out, the next message from the server must be logout.

The model of the communication console component must integrate with the model of the webmail component and the interface specification is useful for understanding the kinds of interactions that can happen between these components. However, it is not clear how the interface specification can be used to help debug the communication console model using testing or simulation. Uchitel et. al [6] describe a method for synthesizing a Modal Transition System (MTS) from an interface specification such as given in Fig. 1 that can address this issue. We describe it in the next section.

3 Partial Model Synthesis

A Labelled Transition System (LTS) is a common formalism for expressing system behaviour and we expect that when the webmail component is known, we can model it with an LTS. LTSs are easily amenable to simulation [2].

Since the webmail component is currently unknown, we would like to characterize the set of possible LTSs that it can become. A *Modal Transition System* [4] is a partial modeling variant of an LTS that can represent a set of LTSs since



(a) Webmail scenario specification *sc*.

$Registered = \langle enable, disable \rangle$ initially *TRUE*
$LoggedIn = \langle authenticate, \{logout, disable\} \rangle$ initially *FALSE*

$$\begin{array}{rrcl} \text{(Legal access)} & p_1 & = & \Box(LoggedIn \Rightarrow Registered) \\ \text{(Private access)} & p_2 & = & \Box(sendMsg \Rightarrow LoggedIn) \\ \text{(Logouts are ack'd)} & p_3 & = & \Box(logout \Rightarrow \mathbf{X}\ logoutMsg) \end{array}$$

(b) Webmail system properties.

Fig. 1. Interface specification for webmail component consisting of (a) sequence diagram and (b) system properties expressed using FLTL.

it allows for uncertainty on the transitions. Specifically, any transition can be marked as *may* if the modeler is uncertain as to whether the transition exists (otherwise, it is *must* by default). For example, Fig. 2 shows an MTS that represents the set of LTSs that satisfy property p_3. The *may* transitions are denoted using a question mark symbol for the appropriate action ("?").

MTSs can also be used to represent scenarios, and Fig. 4 shows the encoding of scenario *sc* from Fig. 1. Furthermore, MTSs can be refined to reduce the possible LTSs they admit. For example, a *may* transition can be refined by turning it into a *must* or removing it. Fig. 3 shows a refinement of the MTS in Fig. 2 that represents the conjunction of all three properties.

[6] shows how to synthesize the above MTSs automatically from FLTL properties and sequence diagrams. In addition, it shows how to merge two MTSs to produce their *minimal common refinement*, allowing synthesis from heterogeneous specifications such as those in Fig. 1. Fig. 5 shows the merge of the MTSs in Fig. 4 and Fig. 5 to produce the MTS encoding the entire interface specification given in Fig. 1.

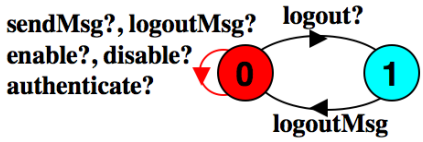


Fig. 2. MTS representing property p_3 from Fig. 1.

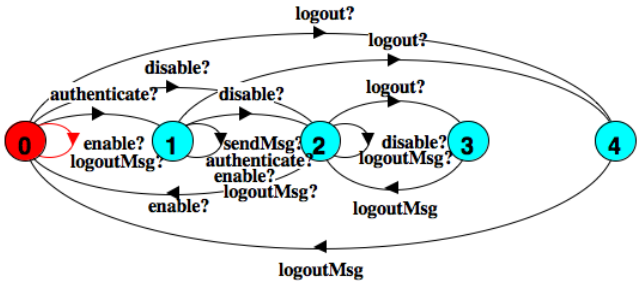


Fig. 3. MTS representing conjunction $p_1 \wedge p_2 \wedge p_3$ from Fig. 1.

4 Dynamic Analysis with a Partial Model

The partial model resulting from the synthesis process is well suited to dynamic analysis techniques. Since a partial model actually represents a set of "concrete" models, one could ask whether all, some or none of the concrete models can exhibit a particular behavior. If only some do, then this provides an impetus to refine the interface specification to eliminate the unacceptable concrete models from the set of possibilities.

For example, the MTS produced by the synthesis illustrated above can be simulated (and model checked) using the Eclipse-based tool MTSA [1]. One way to use simulation here is to do "sequence checking" - i.e., do all LTSs represented by an MTS accept a given sequence of actions? MTSA can be used to animate the execution of the MTS to show the path of transitions taken for the sequence of actions being checked. If this path goes through only the *must* transitions then the sequence exists in all possible LTSs; otherwise, some LTSs do not accept it. MTSA also provides functions to use this information to potentially refine the MTS.

5 Summary and Discussion

In this paper, we have illustrated the use of synthesis of partial models to create approximations of unknown models based on interface specifications. Having such approximations allows us an early application of engineering techniques

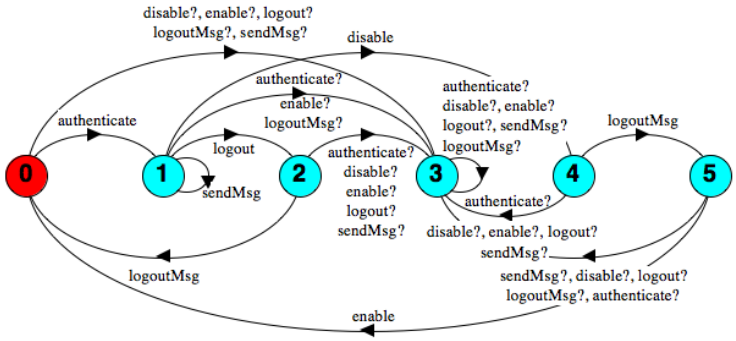


Fig. 4. MTS representing scenario in sequence diagram *sc* from Fig. 1.

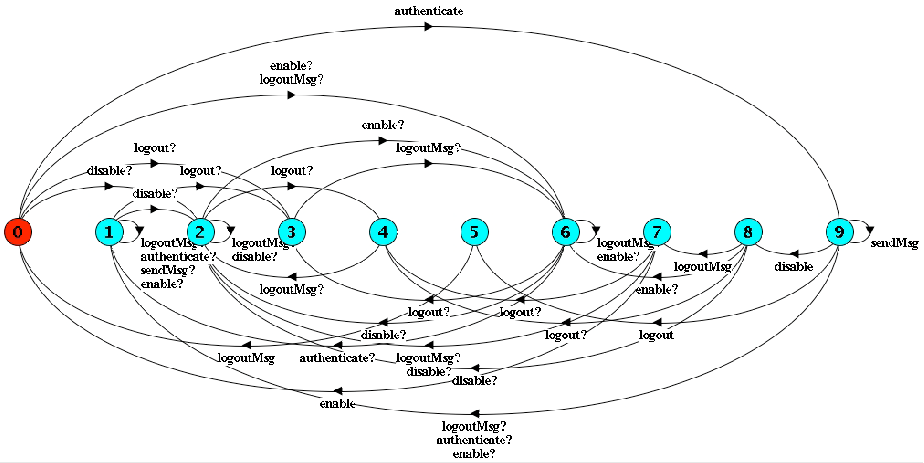


Fig. 5. MTS representing entire specification in Fig. 1.

such as testing and simulation, before the details about models are known. This can help prevent delays in large-scale heterogeneous and distributed software development processes.

In our illustration of synthesis, we assumed that the interface specification is expressed using FLTL properties and sequence diagrams. Although this may be adequate in some situations, the broad scope of the GEMOC program suggests that we must consider extensions to this assumption. In particular, an interface specification should allow for:

– different models of computation: for example, components based on continuous time could allow interface properties expressed in terms of differential equations;

– different types of models: our focus in this example has been on behavioural models, but model integration may include other facets such as structural models, functional models, goal models, etc.;
– domain-specific property languages: for example, the webmail component may include a specification of its user interface layout using a domain-specific language for web layout (e.g., see [3]).

Part of the solution is to use language-independent partial modeling techniques (e.g., see [5]). Our longer term objective is to develop a model specification language that can address these factors and to develop the corresponding partial model synthesis approaches.

Acknowledgements

We would like to thank Robert France and Benoit Combemale for the many insightful discussions that has led to this paper. The work on MTS synthesis and simulation has been done in collaboration with Sebastián Uchitel, Nicolás D'Ippolito, Dario Fischbein and Greg Brunet.

References

1. N. D'Ippolito, D. Fischbein, M. Chechik, and S. Uchitel. MTSA: The Modal Transition System Analyser. In *Proc. of ASE'08*, pages 475–476, 2008.
2. Magee J. Behavioral Analysis of Software Architectures Using LTSA. In *Proc. of ICSE'99*, pages 634–637, 1999.
3. N. Koch, A. Knapp, G. Zhang, and H. Baumeister. UML-Based Web Engineering. In *Web Engineering: Modelling and Implementing Web Applications*, pages 157–191. Springer, 2008.
4. K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Proc. of LICS'88*, pages 203–210, 1988.
5. R. Salay, M. Famelis, and M. Chechik. "Language Independent Refinement using Partial Modeling". In *Proc. of FASE'12*, 2012.
6. S. Uchitel, G. Brunet, and M. Chechik. Synthesis of Partial Behaviour Models from Properties and Scenarios. *IEEE Transactions on Software Engineering*, 3(35):384–406, 2009.