Demonstration of an Operational Procedure for the Model-Based Testing of CTI Systems

Andreas Hagerer[1], Hardi Hungar[1], Tiziana Margaria[1], Oliver Niese[2], Bernhard Steffen[2], and Hans-Dieter Ide[3]

[1] METAFrame Technologies GmbH, Dortmund, Germany
{AHagerer,HHungar,TMargaria}@METAFrame.de
[2] Chair of Programming Systems, University of Dortmund, Germany
{Oliver.Niese,Steffen}@cs.uni-dortmund.de
[3] Siemens AG, Witten, Germany
Hans-Dieter.Ide@wit.siemens.de

Abstract. In this demonstration we illustrate how *a posteriori* modeling of complex, heterogeneous, and distributed systems is practically performed within an automated integrated testing environment (ITE) to give improved support to the testing process of steadily evolving systems. The conceptual background of the modeling technique, called *moderated regular extrapolation* is described in a companion paper [3].

1 Moderated Regular Extrapolation

Moderated **regular extrapolation** aims at providing *a posteriori* descriptions of complex, typically evolving systems or system aspects in a largely automatic way. These descriptions come in the form of extended finite automata tailored for automatically producing system tests, grading test suites and monitoring running systems. Regular extrapolation builds models from observations via techniques from machine learning and finite automata theory. These automatic steps are steered by application experts who observe the interaction between the model and the running system. This way, structural design decision are imposed on the model in response to the diagnostic information provided by the model generation tool in cases where the current version of the model and the system are in conflict.

Moderated regular extrapolation is particularly suited for *change management*, i.e. in cases where the considered system is steadily evolving, which requires continuous update of the systems specification as well.

We will illustrate our method using a regression testing scenario for system level Computer Telephony Integration (*CTI*)(cf. [3, Sec. 4.2], this volume): Here, previous versions of the system serve as reference for the validation of future releases. A new release is required to support any unchanged feature and to enhance it with new or modified features. The iterative process of moderated regular extrapolation (Sec. 2) supports this system evolution, by incrementally building a model comprising the current spectrum of functionality on the basis of concise diagnostic feedback highlighting locations and sources of system/model mismatches.

R.-D. Kutsche and H. Weber (Eds.): FASE 2002, LNCS 2306, pp. 336–339, 2002.

2 Regular Extrapolation in Practice

In this section we sketch the demonstration content, which will successively address the five steps of the model generation by regular extrapolation process by one simple example each.

Trace Collection. To build a model, the system is stimulated by means of test cases and the effects are traced and collected to form an initial model. The example of [3, Fig. 6](left, this volume) shows a simple test case as it is specified in the ITE by a test engineer. Here, three users pick up and hang up the handset of their telephones in arbitrary order. Test case executions are automatically protocoled in form of traces by the ITE's tracer (cf. [3, Fig. 6](right)). In a trace, both states and transitions are labeled with rich labels that describe portions of the system state and protocol messages respectively.

Abstraction. Here, we generalize observed traces to sequential behavioral patterns. The demo will illustrate the effect of abstracting from concrete components to *actors* playing specific roles. An observed trace (actor-set trace) coming from the execution of the test case where this abstraction has taken place is shown in [3, Fig. 6].

Folding. Folding a trace to a trace automaton allows a further powerful generalization of all possible interleaved combinations of actor-set traces. In the folding step, stable states that are considered equivalent are identified and can then be merged. For example, typically all observed devices are classified according to the status of display messages and LEDs. In this step *extrapolation* takes place: the behavior of the system observed so far is extrapolated to an automaton, which typically, due to cycle introduction, has infinite behavior.

The model shown in Fig. 2(left) has been generated via folding from a set of independent traces. It represents the behavior of two users picking-up and hanging-up handsets independently.

Refinement. With new observations, we can refine the model by adding further trace automata to a model. Again, each refinement step is based on the identification of behaviorally equivalent states. In Fig. 1 we show how the trace of [3, Fig. 6](right), is added to the previous model on the left and leads to the model of Fig. 2(left) with four stable system states. Here, a system state is extremely abstract: it is characterized by the number of phones currently picked up. As a comparison, the observations on the original executable test cases were fully instantiated (e.g. they referred to single concrete device names).

Validation. Temporal properties of the models, reflecting expert knowledge, can be checked at any stage by means of standard model checking algorithms. This establishes an independent control instance: vital application-specific frame conditions, like safety criteria guaranteeing that nothing bad happens, or liveness properties guaranteeing a certain progress can automatically checked on the model. In case of failure, diagnostic information in terms of error traces reveals

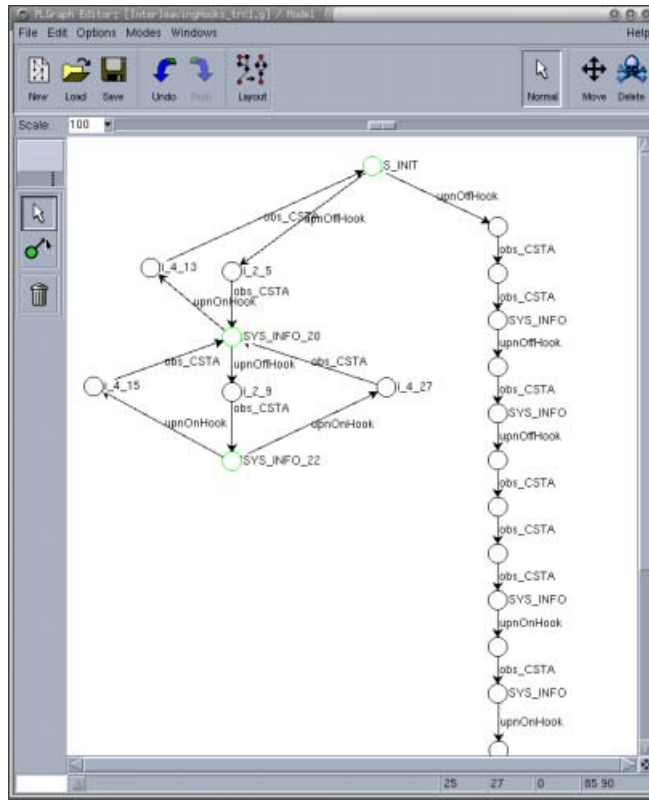

Fig. 1. Adding a new trace to the model

the source of trouble on the model level. An application expert then has to examine whether the revealed problem is just due to the inaccuracies of the model obtained so far, or whether there must be a problem in the underlying system as well.

Validation typically initiates the next iteration of the extrapolation process, which may now also involve technically more updating steps, like, e.g., model reduction, in cases where the model contained too many paths. Our system provides a number of automata theoretic operations and temporal synthesis procedures for the various updating steps. Moreover, it comprises algorithms for fighting the state explosion problem. This is very important, as already comparatively small sets of traces lead to quite big automata. E.g. Fig. 2(right) shows part of a model describing two very simple independent calls. For each call the model describes the correct interplay of the following actions: caller pick-ups handset, dials number, callee pick-ups handset, caller and callee hang-up their handsets. Already this simple scenario leads to a model with 369 states and 441 transitions. Our current research therefore focuses on the investigation of appropriate abstraction methods.

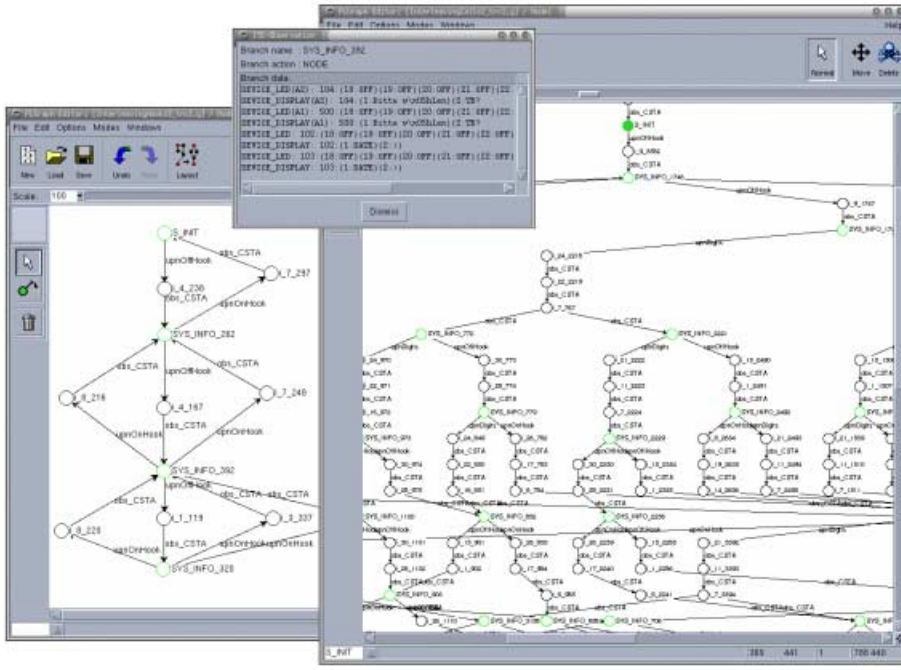

Fig. 2. The refined optimized model (left), example of a larger POTS model (right)

References

1. European Computer Manufactures Association (ECMA). Services for computer supported telecommunications applications (CSTA) phase II, 1994.
2. European Computer Manufactures Association (ECMA). Services for computer supported telecommunications applications (CSTA) phase III, 1998.
3. A. Hagerer, H. Hungar, O. Niese, and B. Steffen: Model Generation by Moderated Regular Extrapolation. In *Proc. of the 5th Int. Conf. on Fundamental Approaches to Software Engineering (FASE 2002)*, this Volume.
4. O. Niese, T. Margaria, A. Hagerer, M. Nagelmann, B. Steffen, G. Brune, and H. Ide. An automated testing environment for CTI systems using concepts for specification and verification of workflows. *Annual Review of Communication*, Int. Engineering Consortium Chicago (USA), Vol. 54, pp. 927-936, IEC, 2001.
5. O. Niese, B. Steffen, T. Margaria, A. Hagerer, G. Brune, and H. Ide. Library-based design and consistency checks of system-level industrial test cases. In H. Hußmann, editor, *Proc. FASE 2001*, LNCS 2029, pages 233–248. Springer Verlag, 2001.