Sensoria

016004

Software Engineering for Service-Oriented Overlay Computers

www.sensoria-ist.eu

Manual for Using the UMC Model of the Finance Case Study

Author(s): Maurice H. ter Beek and Franco Mazzanti (ISTI–CNR)

Date of preparation: March 12, 2010
Revision: final
Dissemination level: PU

Contract start date: September 1, 2005 Duration: 48 months
Project coordinator: Martin Wirsing (LMU)
Partners: LMU, UNITN, ULEICES, UWARSAW, DTU, PISA, DSIUF,
UNIBO, ISTI, FFCUL, UEDIN, ATX, TILab, FAST, BUTE,
S&N, LSS-Imperial, LSS-UCL, MIP, ATXT, CIR

Integrated Project funded by the
European Community under the
"Information Society Technologies"
Programme (2002—2006)

Information Society Technologies

Executive Summary

We show how to use a UMC model of the credit request scenario of SENSORIA's Finance case study, described in detail in [1], and verify properties formulated in the service-oriented temporal logic SocL.

Contents

1 UMC

UMC [2, 3] is an on-the-fly model checker (its current prototype can be experimented via a web interface [5], which also includes a user guide [4]). UMC allows the efficient verification of SocL formulae over a set of communicating UML state machines. SocL [3] is an event- and state-based, branching-time, efficiently verifiable, parametric temporal logic that was specifically designed to capture peculiar aspects of services. UMC's web interface is depicted in Fig. 1.

Figure 1: UMC web interface.

1.1 Selecting a UMC model

By selecting "Model Definition ..." in the Commands Menu on the left, one obtains Fig. 2.

Figure 2: Selecting a model.

Subsequently clicking “Select one of the examples . . . ” brings one to Fig. 3 (it might be necessary to use the scrollbar to select the UMC model `0-0-CreditPortal.umc`).

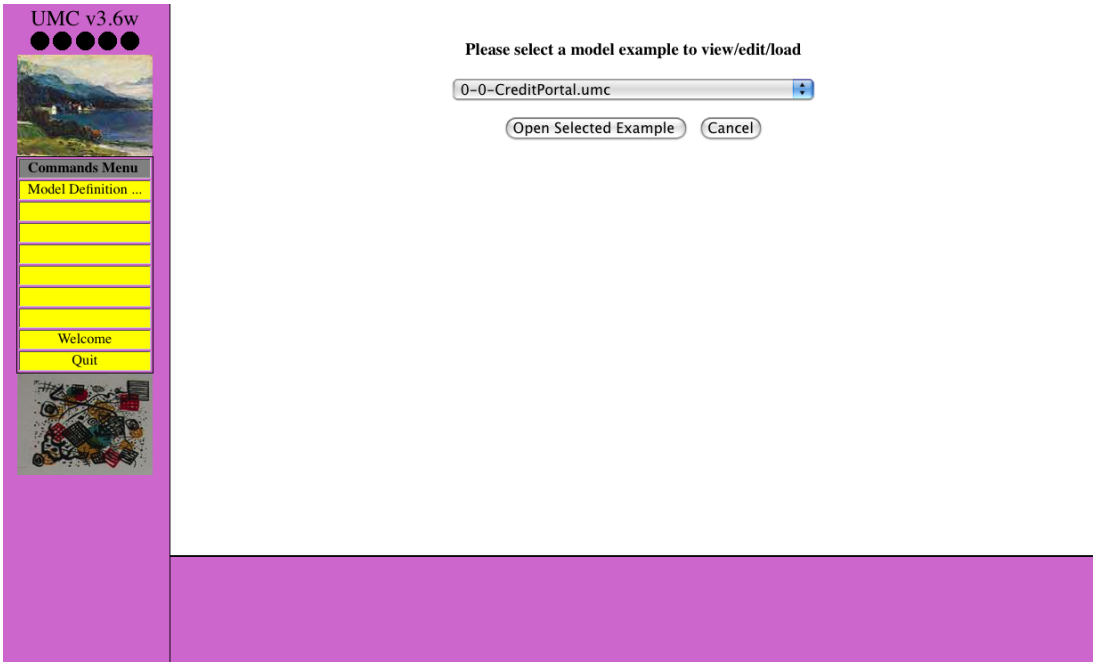

Figure 3: Selecting an example model.

Clicking “Open Selected Example” leads to Fig. 4: the UMC model of the Credit Portal scenario of the Finance case study.

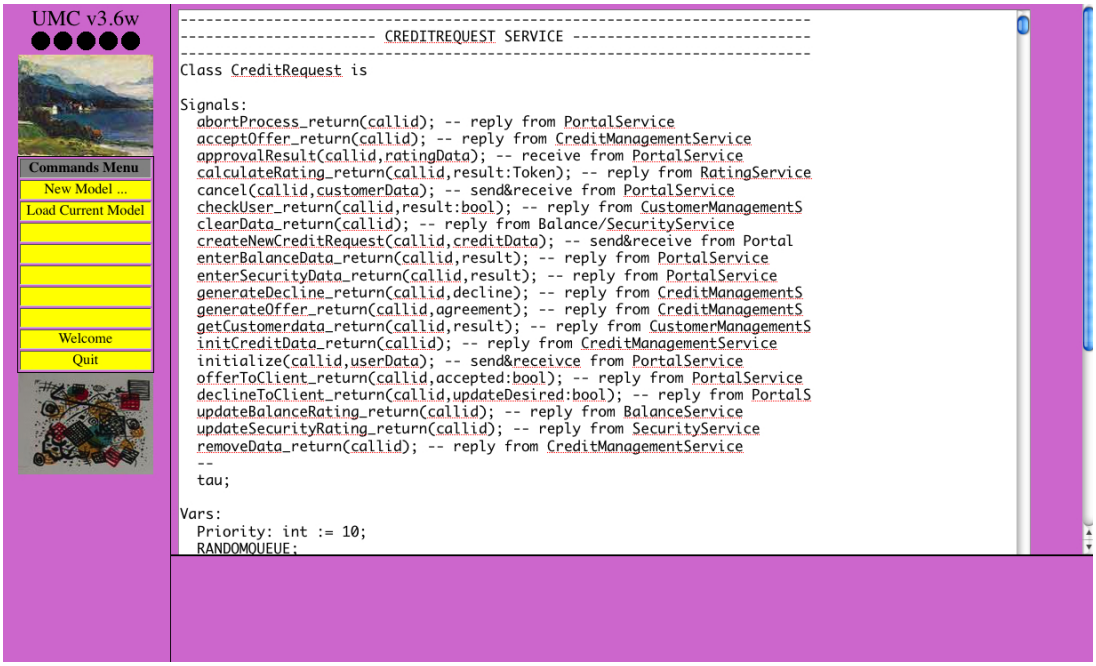

Figure 4: UMC model of Credit Portal.

Using the scrollbar on the right, one can inspect this UMC code.

1.2 Experimenting with a UMC model

To start experimenting the UMC model, one must select "Load Current Model" in the Commands Menu on the left, resulting in Fig. 5. This figure shows the model's classes and active objects, as well as its current (initial) configuration.

Figure 5: A loaded UMC model.

The latter can be inspected further by clicking "(show details ...)", which results in Fig. 6 (using the scrollbar on the right details of variables, active states, event queues and possible evolutions of all active objects can be consulted).

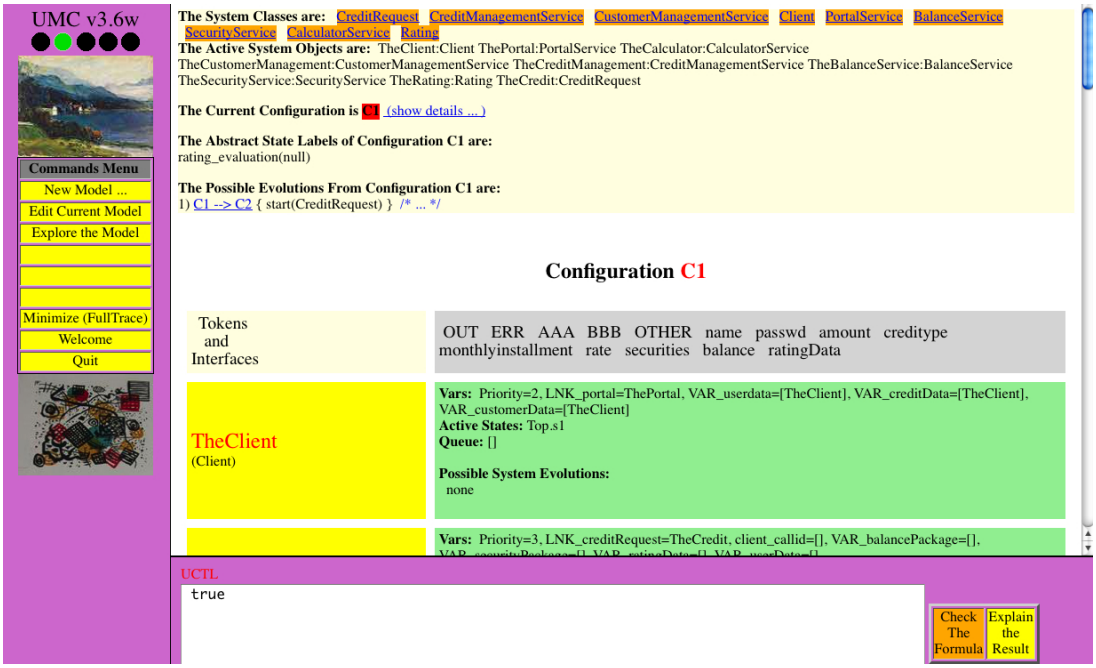

Figure 6: A model's details.

Another possibility provided in Fig. 5 is to perform an evolution step by clicking on “C1 −−> C2”, which results in Fig. 7 (which can be ‘repeated’ in the obvious way to perform more evolution steps).

Figure 7: An evolution step.

Yet another possibility provided in Fig. 5 is to create a minimized abstract evolution graph of the model by clicking on “Minimize (FullTrace)”.

2 Verification with UMC

By means of two examples, we show how to use UMC to verify SocL formulae over the model of the previous section (described in detail in [1]). More example properties that we have verified, inspired by the Patterns of service properties listed in [3], can be found in [1]. As a first example, let us verify that the CreditRequest service is *available*. To this aim, it suffices to verify the SocL formula

$$\texttt{AF}\,(\texttt{accepting\_requests(initialize)}),$$

which means that in every state the CreditRequest service may eventually accept (the initialization of) a credit request. This is done by inserting this formula in the field labelled UCTL (of which SocL is a specialized version) on the lower side, which by default contains the formula `true`, and subsequently pushing the button “Check The Formula” on the lower right side. This results in Fig. 8, i.e. the above formula is TRUE. Note that a UMC model needs to be loaded before verifying properties, so the UCTL field only appears from Fig. 5 onward.

2.1 Interpreting a Counterexample

As a second example, we now repeat the above operations for the SocL formula

$$\texttt{AF}\,(\texttt{accepting\_requests(cancel)}),$$

which results in Fig. 9, i.e. this formula is FALSE.

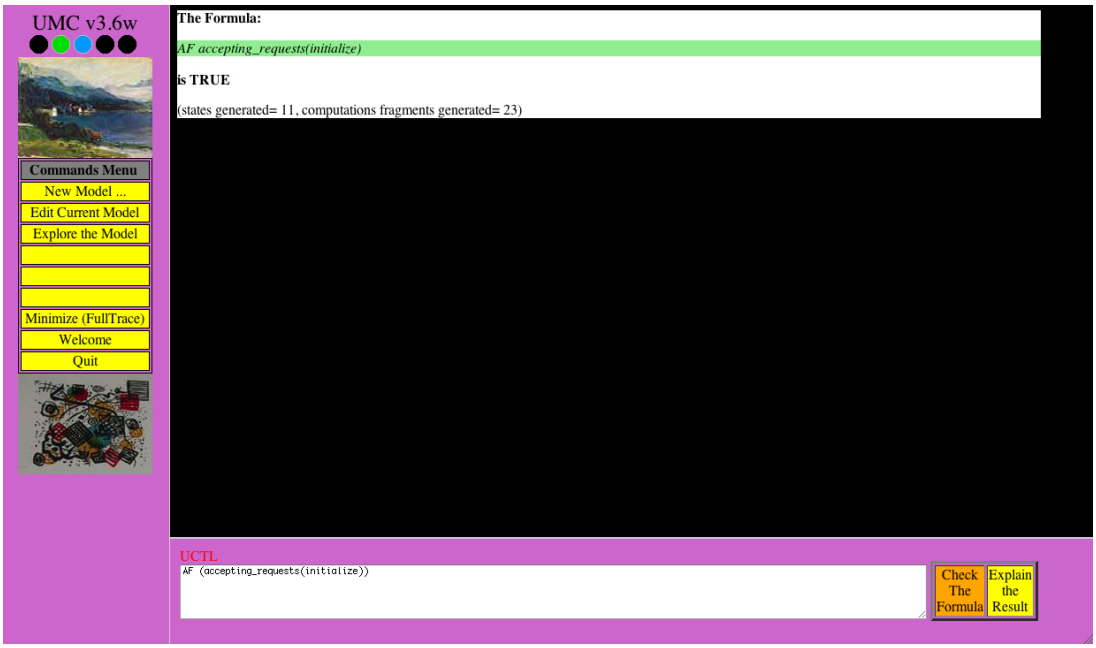

Figure 8: Result (true) of a verification.

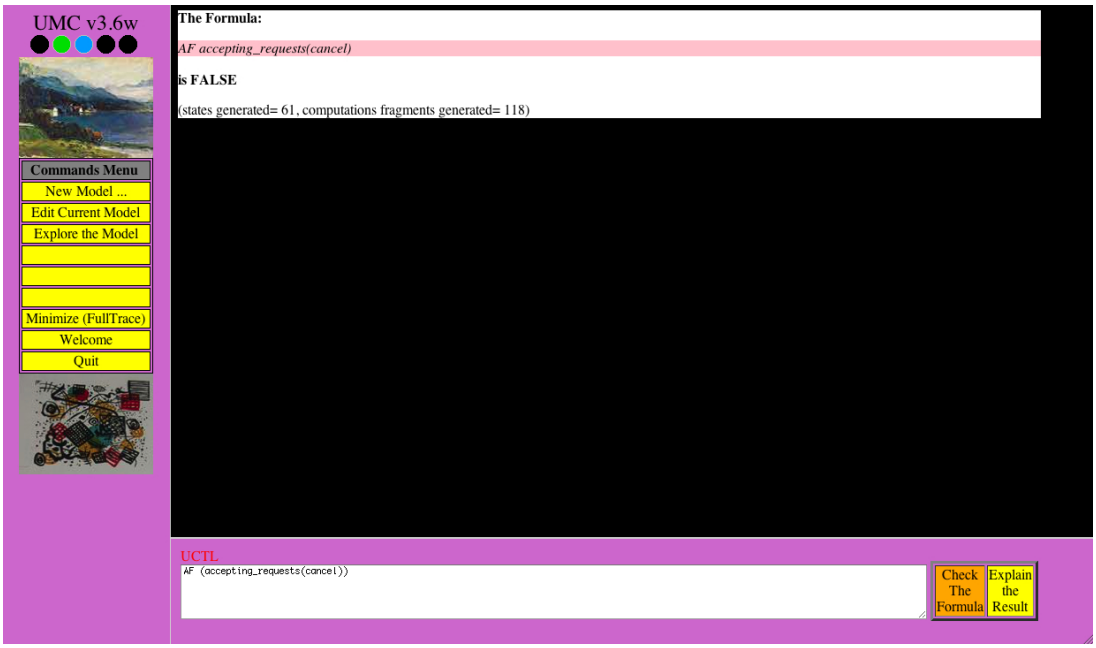

Figure 9: Result (false) of a verification.

Pushing the button "Explain the Result" on the lower right side of Fig. 9 results in Fig. 10, displaying the counterexample produced by UMC. The node names are hyperlinks which, when followed, allow one to observe all details of that configuration. Furthermore, while abstract transition labels are always fully displayed on the right-hand side of the transitions, their corresponding underlying ground events (which are useful for understanding what exactly is happening in the ground model's evolutions) are shown as dynamic tooltips that appear when the cursor is moved over the "/*...*/" regions. Note that the explanation returned by UMC has the form of a (partial) proof, in the sense that not only the *witnessing* model fragment but also the subformulae holding in the various substates, are put in evidence; moreover, only what are considered the *useful* parts of the explanation are shown.

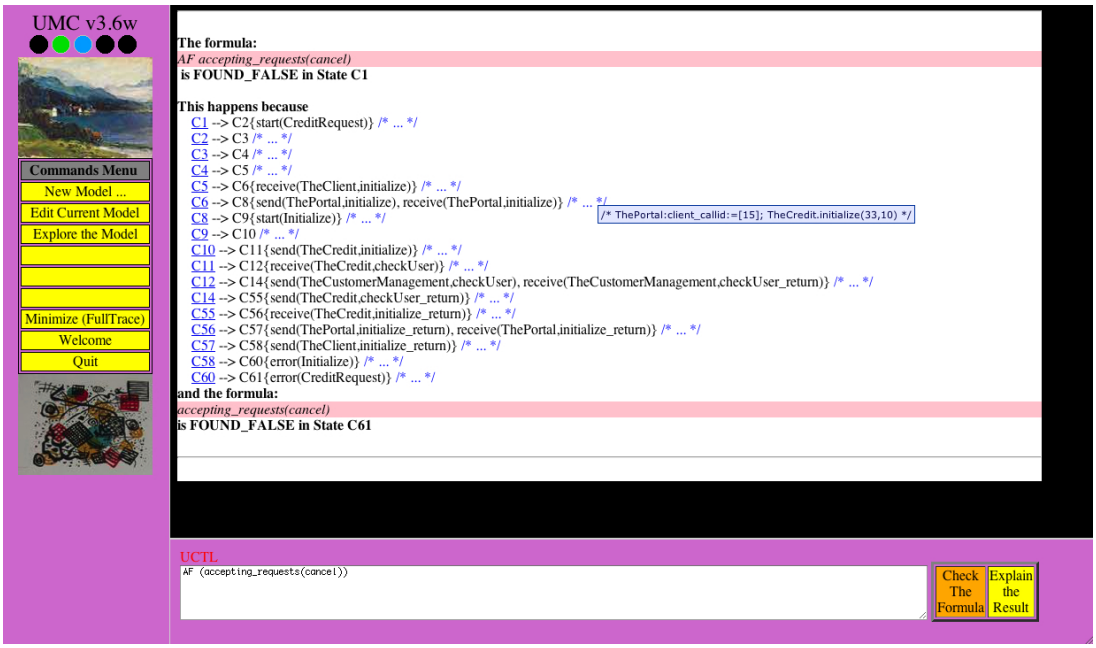

Figure 10: Counterexample of a formula.

References

[1] M.H. ter Beek and F. Mazzanti, Modelling and Analysing the Finance Case Study in UMC. Technical Report 2010-TR-007, ISTI–CNR, 2010.

[2] M.H. ter Beek, A. Fantechi, S. Gnesi and F. Mazzanti, A state/event-based model-checking approach for the analysis of abstract system properties. In *Science of Computer Programming*, 2010.

[3] S. Gnesi and F. Mazzanti, An Abstract, on the Fly Framework for the Verification of Service Oriented Systems. In [6], 2010.

[4] F. Mazzanti, Designing UML models with UMC. Technical Report 2009-TR-43, ISTI–CNR, 2009.

[5] UMC: `http://fmt.isti.cnr.it/umc/`.

[6] M. Wirsing and M. Hölzl (Eds.), *Rigorous Software Engineering for Service-Oriented Systems—Results of the SENSORIA project on Software Engineering for Service-Oriented Computing*, Springer, 2010.