Clint: A Composition Language Interpreter (Tool Paper)

Javier Cámara, Gwen Salaün, and Carlos Canal

Department of Computer Science, Universidad de Málaga, Spain
{jcamara,salaun,canal}@lcc.uma.es

1 Introduction

Composition of components or services is a crucial issue when building new applications by reusing existing pieces of software. This task turns out to be tedious when behavioural descriptions, acknowledged as one of the essential parts of component interfaces, are taken into account. Furthermore, mismatches may exist between component interfaces, and adaptation [2] is necessary to help the designer to solve them.

In this tool paper, we present Clint, a composition language interpreter. Clint accepts as inputs (i) behavioural interfaces described using Labelled Transition Systems (LTSs), and (ii) a composition specification. The latter is described using vectors that make explicit component interactions, and an LTS labelled with vectors to define an order on the application of some interactions [5] (important to solve some specific mismatch cases). Clint implements step-by-step simulation for the aforementioned inputs, as well as techniques to validate the composition specification, avoiding undesirable situations such as unexplored parts of the composition or deadlock states.

In the remainder of this paper, Section 2 introduces behavioural interfaces and composition specifications. In Section 3, the main functionalities of Clint are presented. Section 4 draws up some concluding remarks.

2 Behavioural Interfaces, Composition and Adaptation

Behavioural interfaces of components are described using par-LTSs (A, S, I, F, T) [4], referred to as LTSs in the rest of the paper for short, where A is a set of labels called alphabet representing message emissions and receptions (respectively written using `!` and `?`), S is a set of states (either basic or concurrent), $I \in S$ is the initial state, $F \subseteq S$ are final states, and $T \subseteq S \times A \times S$ are transitions. Par-LTSs extend basic LTSs by taking into account concurrent behaviours. This is achieved by introducing *concurrent states*. These states are identified by two or more labels, one for each concurrent branch.

A *composition specification* describes composition constraints and adaptation requirements. *Vectors* relate messages used in different components to implement interactions. A vector for a set of components C_i with $i \in \{1, .., n\}$, is a tuple

J. Fiadeiro and P. Inverardi (Eds.): FASE 2008, LNCS 4961, pp. 423–427, 2008.

$\langle l_1, \ldots, l_n \rangle$ with $l_i \in A_i \cup \{\varepsilon\}$, where each A_i is the alphabet of component C_i and ε means that some component does not participate in the interaction, *e.g.,* $\langle c_1 : comm!, c_2 : \varepsilon, c_3 : comm? \rangle$ for components $\{c_1, c_2, c_3\}$. Constraints on the application ordering of vectors can be given with an LTS whose alphabet is a set of vectors. A composition specification, for a set of components C_i with $i \in \{1, \ldots, n\}$, is a couple (V, L) where V is a set of vectors for components C_i, and L is an LTS whose alphabet is V.

Clint relies on adaptation techniques and algorithms presented in [4] to generate execution traces from behavioural interfaces and a composition specification. However, the composition and adaptation engine is implemented as an external module. Therefore, it makes Clint generic at this level, and allows to take into account other adaptation techniques as those proposed in [3,5,8,9].

3 Overview of Clint

Clint implements both a composition and adaptation engine, and a graphical user interface to load, visualise and modify the different inputs. The graphical interface is also used to animate and validate the composition of an arbitrary number of components. The tool has been implemented in Python, using the wxWidgets toolkit technology for the development of the user interface. The prototype accepts component interfaces described using an XML-based format or a textual format. The composition specification is written using another XML-based notation specific to the tool.

Once the inputs are loaded, Clint uses the Object Graphics Library (OGL) to visualise component interfaces and the composition specification. These inputs can be edited and modified making use of the graphical interface. As regards animation and validation techniques, first, interactive simulation of the composition can be performed in two different modes:

- Safe mode (default). The interface offers the user only *safe* vectors for selection (*i.e.,* there exists at least one correct termination state of the system after its application).
- Unsafe mode. The interface offers all applicable vectors to the user. This allows the possibility of applying vectors leading to incorrect compositions, but it may be interesting in order to observe and understand potential flaws of the composition process.

Additionally, the tool is able to perform validation on the composition specification *wrt.* component LTSs. The basic idea is to generate automatically many random execution traces using the composition engine (unsafe mode). From such a set of traces, states and transitions are labelled with the number of times they are traversed. These traces are also used to identify unreachable states or transitions which are never traversed in the composition specification. Clint colours the composition specification, highlighting such unreachable states and transitions in the graphical representation. Moreover, Clint can extract traces leading

to deadlock situations. Indeed, when a non-final state in the composition specification is reached where no further vectors can be applied, the tool identifies the sequence of vectors which has lead to the current situation as a deadlock trace.

Figure 1 gives an overview of Clint. The *simulation toolbar* on the top left side of the interface, is used to control the evolution and mode of the simulation. Just below, the *applicable vectors* panel displays the set of currently applicable vectors. In this panel, the user can successively select vectors to be applied in every step of the composition, visualising at the same time the effects of the application of a specific vector on the graphical representation of component interfaces and the composition specification on the right. We can also observe the *simulation state* panel, which displays the current states of the components and the composition specification. Finally, the *simulation trace* panel displays the trace of the current simulation, including vectors applied, as well as the actions performed by components and the composition engine.

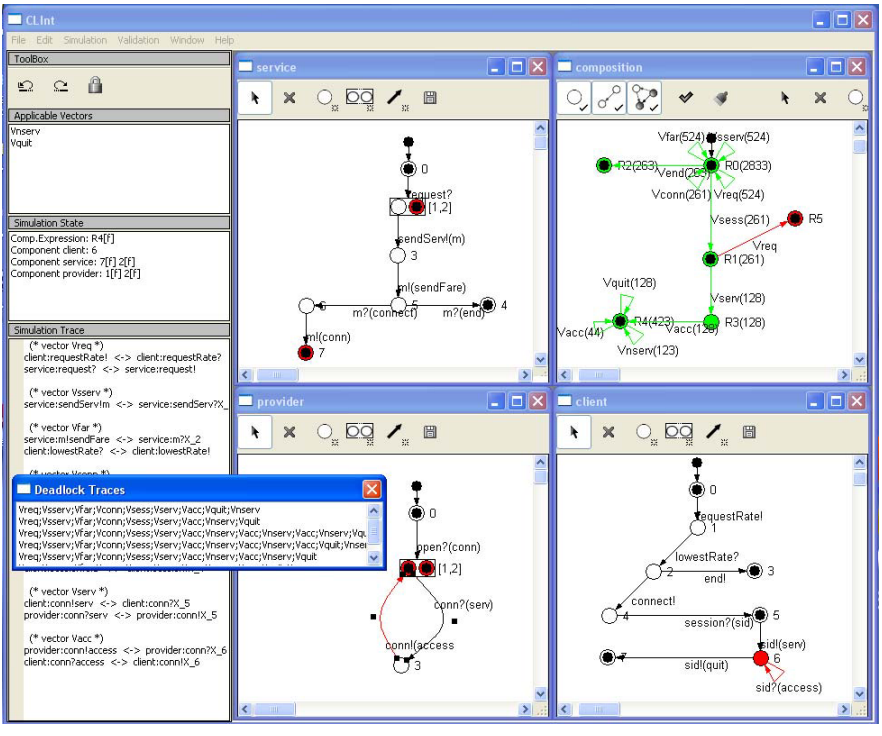

Fig. 1. Validation of a rate finder service using Clint

Clint has been validated on more than 200 examples, either small ones to experiment boundary cases, or real-world examples such as travel agency, push-out advertisements, multi-device service, rate finder service, on-line computer material store, video-on-demand, music player, disk manager, library management

systems, a SQL server and several other client/server systems. For a comprehensive description of several real-world examples, such as a rate finder service or a push-out advertisement service, see the Clint webpage [1].

4 Concluding Remarks

In this paper, we have presented Clint, a graphical and user-friendly tool to specify and validate compositions of component behavioural descriptions. Our tool is being applied to the WF/.NET 3.0 platform in order to support the reuse and composition of such components [7]. Compared to other tools such as LTSA [6], Clint focuses on the specification and validation of the composition, whereas LTSA focuses on the component description, and relies on basic synchronization techniques. As regards future work, our main perspective aims at extending Clint to support the incremental construction of the composition specification. Indeed, the writing of this specification by a designer is a difficult and error-prone task. On the other hand, approaches dedicated to the automatic generation of compositions are not mature enough. An assisted approach seems to be a good trade-off between complete automation and manual writing of the composition specification. Clint will be extended to accept a partial specification of the composition, point out composition issues, and propose possible solutions or further message correspondences to complete this specification. We also plan to extend validation techniques by connecting Clint to model-checking tools, such as SPIN or CADP.

Acknowledgements. This work has been partially supported by the project TIN2004-07943-C04-01 funded by the Spanish Ministry of Education and Science (MEC), and project P06-TIC-02250 funded by the Andalusian local Government. We thank C. Joubert and E. Pimentel for their fruitful comments.

References

1. Clint - version 01 / September 2007 (Available from J. Cámara's Webpage)
2. Becker, S., Brogi, A., Gorton, I., Overhage, S., Romanovsky, A., Tivoli, M.: Architecting Systems with Trustworthy Components. In: Reussner, R., Stafford, J.A., Szyperski, C.A. (eds.) Architecting Systems with Trustworthy Components. LNCS, vol. 3938, Springer, Heidelberg (2006)
3. Bracciali, A., Brogi, A., Canal, C.: A Formal Approach to Component Adaptation. Journal of Systems and Software 74(1), 45–54 (2005)
4. Cámara, J., Salaün, G., Canal, C.: Run-time Composition and Adaptation of Mismatching Behavioural Transactions. In: Proc. of SEFM 2007, IEEE Computer Society Press, Los Alamitos (2007)
5. Canal, C., Poizat, P., Salaün, G.: Synchronizing Behavioural Mismatch in Software Composition. In: Gorrieri, R., Wehrheim, H. (eds.) FMOODS 2006. LNCS, vol. 4037, Springer, Heidelberg (2006)
6. Magee, J., Kramer, J.: Concurrency: State Models & Java Programs. Wiley, Chichester (1999)

7. Cubo, J., Salaün, G., Canal, C., Pimentel, E., Poizat, P.: A Model-Based Approach to the Verification and Adaptation of WF/.NET Components. In: Proc. of FACS 2007. ENTCS, Elsevier, Amsterdam (to appear, 2007)
8. Mateescu, R., Poizat, P., Salaün, G.: Behavioral Adaptation of Component Compositions based on Process Algebra Encodings. In: Proc. of ASE 2007, IEEE Computer Society Press, Los Alamitos (2007)
9. Motahari Nezhad, H.R., Benatallah, B., Martens, A., Curbera, F., Casati, F.: Semi-Automated Adaptation of Service Interactions. In: Proc. of WWW 2007, ACM Press, New York (2007)