Automated Evaluation of Coordination Approaches[†]

Tibor Bosse Mark Hoogendoorn Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{tbosse, mhoogen, treur}@cs.vu.nl

1. Introduction

Coordinating processes in a complex software (or agent) system is a nontrivial issue. By a component-based approach to software systems, a divide and conquer strategy can be used to address the various aspects involved. This may lead to a possibly large number of components, which each can be analysed and designed independently. However, a designer may still be left with the problem how all these fragments can be combined into a coherent system. To solve such a problem, many different coordination approaches have been proposed, each having its advantages and drawbacks. Important questions when choosing such a coordination approach are the suitability, correct functioning, and efficiency of the approach for the particular component-based system. This paper presents a methodology to enable a comparison of such factors for the different coordination approaches in a series of test examples.

2. Comparison Methodology

To explore possibilities to address the coordination problem, an evaluation methodology, supported by a software environment, has been created which follows the following steps: (a) a number of *coordination approaches* are selected, (b) a number of *test examples* representing specific software component configurations are chosen, (c) based on each of these coordination approaches a *simulation model* is formally specified, (d) related to the test examples, relevant *requirements* are formally specified in the form of relevant dynamic properties, (e) *simulations* are performed where selected coordination approaches are applied to the chosen test examples, resulting in a number of

[†] The full version of this paper appeared in: *Proceedings of the Eighth International Conference on Coordination Models and Languages, Coordination'06*. Lecture Notes in Computer Science, vol. 4038. Springer Verlag, 2006, pp. 44-62.

simulation traces, and (f) the simulation traces are *evaluated* (automatically) for the specified requirements.

To enable a formal specification of the simulation model, and an evaluation of the resulting traces, the Temporal Trace Language (TTL) [1] is used. TTL first of all allows the specification of executable properties for each of the coordination approaches. After such properties have been specified and test examples are given as input, a simulation engine is used to execute the properties. The execution results in a formal trace with sequences of events that occurred during the simulation of the coordination approach for a particular test example. With such a formal trace as input, and properties on a non-executable level specified in TTL that ought to be fulfilled by the coordination approach (e.g. successfulness, efficiency), a verification tool is used to automatically verify whether these properties are indeed satisfied for the given trace.

3. Results

Based on the approach presented above, the following well-known coordination approaches have been compared: (1) Behaviour networks introduced by Pattie Maes [2]; (2) the pandemonium model [4], and (3) voting [3]. Since the approach also requires test examples to be specified, a choice has been made to use relatively simple workflow patterns. These patterns can be seen as building blocks for more complex patterns occurring in real-life component-based systems. In total, seven such test examples have been used. All approaches turned out effective in finding the solution in all cases. However, none of the approaches is always efficient for all test examples. The behaviour networks and pandemonium approaches perform equally well; they succeed for the “simple” cases and sometimes fail to be efficient for two complicated cases. Surprisingly, the voting approach always finds an efficient solution for one of the complicated cases but fails in a rather trivial case. Finally, the overall methodology turned out to be very useful in comparing the different coordination approaches.

References

[1] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A, and Treur, J. A Temporal Trace Language for the Formal Analysis of Dynamic Properties. Technical Report, Vrije Universiteit Amsterdam, Dept. of Art. Int., 2006.

[2] Maes, P. How to do the right thing. Connection Science 1(3): pp. 291-323.

[3] Ordeshook, P. Game theory and political theory: An Introduction. Cambridge: Cambridge University Press, 1986.

[4] Selfridge, O. G. Pandemonium: a paradigm for learning in mechanization of thought processes. In Proceedings of a Symposium Held at the National Physical Laboratory, pages 513-526, London, November 1958.