Automated Component-Based Configuration: Promises and Fallacies

Sander van Splunter[a] Niek J.E. Wijngaards[a]
Frances M.T. Brazier[a] Debbie Richards[b]

[a] IIDS Group, Vrije Universiteit Amsterdam, 1081 HV Amsterdam

[b] Macquarie University, Sydney 2109, Australia

The full version of this paper appeared in Proceedings of the Adaptive Agents and Multi-Agent Systems workshop at the AISB 2004 Symposium, Leeds, UK, pp. 130-145, 2004.

Re-use of software components is standard practice in software design and development in which humans play an important role. In many dynamic environments, however, (semi-)automated configuration of systems is required. Three such domains are examined: Agent Factories, Web service configuration, and general software composition. The differences and similarities, and the progress that is being made, are discussed.

Internet applications require flexibility to accommodate changes in their environment. As manual adaptation is not pragmatic when multitudes of agents and Web services are in use, automation becomes a necessity. Unfortunately, the current state of the art, even in software engineering, does not include fully automated component-based adaptation. Current research focuses on component-based configuration of agents, Web services, and software composition: a pre-condition for adaptation.

For each of these three domains, a number of specific research approaches are discussed and compared. Agent Factories provide facilities for (semi-) automated creation and optionally adaptation of software agents (e.g., Brazier & Wijngaards, Colier & O'Hare, and Cossentino et al). Web service configuration combines well-described Web services into more complex services (e.g., Casati et al. (Eflow), and Cardoso & Sheth). Component-based software engineering often includes semi-automated composition (e.g., De Bruin & Van Vliet (Quasar)). See our full paper for more detail.

The comparison shows differences with respect to the following:
Component definitions. Web service approaches employ standards, are used by a large community, and build on existing (Web-)protocols for inter-component communication. Agent Factories and the more general component-based software engineering methods are gray-box approaches, offering hooks for composition.

Component annotation. Only the Web service community has emerging computer-interpretable annotation standards. Though the human-interpretable general software engineering annotations, e.g. UML, are more widely applied and accepted. The Web service annotation standards currently do not extend software engineering standards. Within the agent community these standards are more generally accepted, e.g. AUML.
Component availability. Annotated Web services are becoming widely available with a large supportive community. The creation of semantic descriptions is part of the Web service development process. The more general component-based software engineering methods have also resulted in a critical mass of components. Agents are not often developed for reusability, leading to a scarce availability of agent components. Software engineering methods and Agent Factories, in general, lack publicly available semantic descriptions of their components.
Configuration processes. Fully automated configuration is not widely available. The Agent Factories have made the most progress in this area. Web services focus on limited configuration, and software engineering focuses explicitly on semi-automated configuration, though research on automated configuration and adaptation is appearing (e.g. self-managing/healing systems).

In conclusion, much needed interdisciplinary research may be most fruitful when (1) combining configuration- with annotation-expertise, (2) generalising and standardising reusable, configurable (non-black box) components, (3) using sound software engineering practices.

Acknowledgements

The authors thank Marta Sabou for her contributions to Web service configuration. The authors are grateful to the support provided by Stichting NLnet, http://www.nlnet.nl/.