Adaptation Mechanisms for Role-Based Software Systems

Martin Weißbach(✉)

Chair of Computer Networks, Institute for Systems Architecture,
Technische Universität Dresden, Dresden, Germany
martin.weissbach1@tu-dresden.de

1 Introduction and Related Work

Software Systems have become incredibly large and complex, hence, difficult to develop, maintain and evolve. Furthermore, those software systems do not operate in a stable environment. Certain properties of the environment, e.g. available bandwidth, through-put or workload of the hosting machine of the software system, vary over time and might influence the system's performance adversely under certain conditions. This issue is addressed by self-adaptive software systems, which are systems that can change their own behavior in response to changes in its operational environment [2].

Moreover, in a distributed application, local changes might require changes of remote parts of the application as well. Consequently, the adaptation runtime has to provide mechanisms to ensure that such related operations can be executed synchronously, i.e. either all operations are executed successfully or changes are reverted to prevent the application from being in an inconsistent state.

Recently, role-oriented programming has come into focus to allow behavioral adaptations on the level of programming languages. Roles are used on the design and implementation level to cover context-dependent behavior of software entities to increase the expressivity of static and dynamic parts of an application. In [1,3] approaches were presented to incorporate self-adaptive software systems and role-oriented programming, but the execution of the planned changes was not closer investigated.

This thesis further investigates the execution of adaptations of role-bases software systems, especially of distributed role-based applications.

2 Discussion

The general concept of roles as adaptable entities can be applied at multiple layers of a software system. Coarse-grained structural adaptations, e.g. exchanging components, would be possible as well as fine-grained modifications of the component's behavior, if implemented using roles.

Our research will mainly cover two parts: First and foremost, we are concerned with the behavioral modification of software systems at runtime that roles allow. We develop a set of adaptation operations that operate on roles

I. Ciuciu et al. (Eds.): OTM 2015 Workshops, LNCS 9416, pp. 3–4, 2015.
DOI: 10.1007/978-3-319-26138-6_1

rather than on components or runtime objects directly and will therefore be applicable to both layers what makes the adaptation more transparent to the system controlling the adaptation. The controlling system must further determine a safe point in time in the programs execution to alter the system without any loss of data. Hence, we will discuss a lifecycle for roles at runtime that supports the adaptation operations and helps to prevent unwanted behavior during the adaptation as well as data loss. Roles are usually bound to players, simply passivating a role when a player's behavior is supposed to be modified is not sufficient, e.g. when roles are exchanged it must be ensured that state information are preserved ant the new role is activated after it has been bound to the player. Second, when the application's context changes, multiple roles might have to be exchanged in a coordinated and synchronized manner, e.g. if two roles on remote nodes collaborate, it might be necessary to exchange both roles if one of them has to be exchanged due to context changes. Therefore, we are investigating mechanisms how the controlling system can ensure the safe transition of the application from an outdated source state to the desired target state. Crucial at this point is especially the decentralized execution of such operations in a distributed software system.

As possible evaluation criteria, the performance and reliability of the adaptation execution at runtime can be considered, especially in distributed and concurrent applications where invalid role configurations are supposed to be prevented during adaptation. Closely coupled to this issue is the interrupt time that is required to exchange roles at runtime. Naturally, that time frame is supposed to be minimal. Moreover, a formal proof that the algorithms and protocols that drive the execution of adaptation operations do not run into deadlocks and behave as specified would be also desirable.

Acknowledgments. This work is funded by the German Research Foundation (DFG) within the Research Training Group "Role-based Software Infrastructures for continuous-context-sensitive Systems" (GRK 1907).

References

1. Monpratarnchai, S., Tetsuo, T.: Applying adaptive role-based model to self-adaptive system constructing problems: a case study. In: 2011 8th IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASe), pp. 69–78. IEEE (2011)
2. Oreizy, P., Gorlick, M.M., Taylor, R.N., Heimhigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D.S., Wolf, A.L.: An architecture-based approach to self-adaptive software. IEEE Intelligent Systems **14**(3), 54–62 (1999)
3. Tamai, T., Monpratarnchai, S.: A Context-Role Based Modeling Framework for Engineering Adaptive Software Systems. APSEC **1**, 103–110 (2014)