Distributed Adaptation Reasoning for a Mobility and Adaptation Enabling Middleware

Nearchos Paspallis and George A. Papadopoulos

Department of Computer Science, University of Cyprus,
75 Kallipoleos Street, P.O. Box 20537, CY-1678, Nicosia, Cyprus
{nearchos, george}@cs.ucy.ac.cy

Abstract. The prospect of adaptive, mobile applications provides both opportunity and challenge to the application developers. Adaptive, mobile applications are designed to constantly adapt to the contextual conditions with the aim of optimizing the quality of their offered service. In this respect the MADAM project provides software engineers with reusable models, tools and runtime support for enabling adaptive behavior in their mobile applications. This paper presents an extension to the MADAM middleware architecture which enables distributed compositions. To this end, a new adaptation reasoning approach is described, which improves on the original one in two ways: it allows decentralized reasoning for selecting the most suitable adaptation and it supports distributed application composition. Moreover, the proposed approach is argued to provide additional benefits such as robustness, agility and scalability.

Adaptive, mobile and pervasive computing applications are designed to constantly adapt to their contextual conditions in an autonomous manner. The aim of the adaptations is to optimize the quality of the service offered to the end users. This study builds on the results established by existing solutions [1, 2], and extends them to introduce mechanisms for enabling distributed adaptation reasoning while at the same time maintaining attributes such as robustness, agility and scalability.

Modern approaches define adaptive, component-based applications as collections of software components which can be configured according to a number of different compositions. Furthermore, technologies such as reflection and component orientation enable reasoning and possibly altering of their behavior. An important question though is *how* can the underlying middleware automatically reason about the context and *select* an optimal composition to adapt to. This question becomes further challenging as additional attributes such as robustness, agility and scalability are aimed.

The proposed approach depends on composition plans which are defined at design time and which can be used to dynamically construct different variants of the application. Individual variants are designed so that they offer certain advantages, such as for example better resource utilization for a particular context. Naturally, each variant is designed with the aim of maximizing the utility of the application for at least some points in the context space. While multiple options are possible for the realization of the adaptation reasoning, utility functions offer the important advantage of supporting adaptation reasoning of components which become available after deployment.

R. Meersman, Z. Tari, and P. Herrero et al. (Eds.): OTM Workshops 2006, LNCS 4277, pp. 17 – 18, 2006.

Utility functions can be used to compute the *utility*, i.e. a quantifiable measure of the quality of the service as it is experienced by the application users. In this respect, the overall objective of the middleware can be defined as the continuous evaluation, and selection of a composition which maximizes the utility. Any knowledgeable decision requires that the reasoning process is aware of the contextual information of all the parts involved. In distributed environments, this implies that the contextual information of all participating hosts must be communicated to the host which performs the adaptation reasoning.

The continuous computation of the utility values can be quite costly though, especially in frequently changing environments such as in mobile and pervasive computing settings. In this respect, two custom-tailored adaptation reasoning approaches are introduced: *proactive* and *reactive* adaptation reasoning. These two approaches differ in the timing of the adaptation reasoning and its required steps. Proactive reasoning requires that all context data is communicated as soon as it becomes available. Contrary to this, reactive adaptation reasoning defers the communication of such context data until they are actually needed. Additional (hybrid) approaches are also possible.

Both options provide individual benefits which make them better choices, depending on the particular requirements of the application. For example, the proactive approach is more likely to achieve faster and more accurate decisions as more context data is available to the decision making process at any moment. In contrast, the reactive approach is better in terms of resource consumption as the context data are communicated only when needed. The latter benefit becomes more important when the context changes more often than the rate at which the application is needed to adapt.

Besides the benefits of the two individual adaptation reasoning approaches, it is argued that an implementation middleware can also benefit by using hybrid approaches, or by dynamically switching from one approach to the other on demand. Furthermore, it is argued that the use of utility functions in this approach not only enables *proactive* and *reactive* adaptation reasoning, but also enables the construction of distributed protocols which satisfy the *robustness*, *agility*, and *scalability* requirements. Robustness is achieved by means of supporting applications to re-compute and failover to alternative, possibly centralized, compositions when a failure (e.g. network outage) prevents the originally selected adaptation. Additionally, the ability to re-compute and implement only a part of a composition greatly improves the agility of an application, especially in the case of the proactive approach. Finally, scalability is achieved as a means of the protocol support for distributed, decentralized computation of the utility functions and construction of compositions.

References

1. Floch, J., et al., Using Architecture Models for Runtime Adaptability, Software, IEEE, 2006. Volume 23 (Number 2): p. 62-70.
2. Paspallis, N. and G.A. Papadopoulos, An Approach for Developing Adaptive, Mobile Applications with Separation of Concerns, to appear in the 30th Annual International Computer Software and Applications Conference (COMPSAC), Chicago, IL, USA, September 16-21, 2006: IEEE.