Reconfiguration of Service Failures in DAMASCo using Dynamic Software Product Lines

Javier Cubo, Nadia Gamez, Ernesto Pimentel, Lidia Fuentes

Dpto de Lenguajes y Ciencias de la Computación, Universidad de Málaga
{cubo, nadia, ernesto, lff}@lcc.uma.es

Abstract. Building service-based applications requires providing the ability to handle, maintain or upgrade the services that compose these applications. As same services may be used by a wide variability of applications, the management of the heterogeneity at runtime is required. This is crucial to reconfigure applications in case of service failures. The DAMASCo framework reduces the complexity of modeling services focusing on the discovery, composition and adaptation of context-aware services. But currently, it does not support the dynamic reconfiguration of service-based applications. In this work, we follow a Dynamic Software Product Line approach to extend DAMASCo for providing reconfiguration to support specific situations of fails at runtime. We propose a novel approach of grouping services in families facilitating the selection and usage of similar services in case of fails. We apply our approach to an intelligent transportation system case study where DAMASCo composes and reconfigures the necessary services to provide a dynamic route for a driver’s request.

Keywords: Service Reconfiguration; Heterogeneity; DSPL.

Using Dynamic Software Product Lines for Reconfiguration

Service-based applications need infrastructures with reconfiguration capabilities, to meet the changing requirements of businesses, environments and users. These infrastructures must handle, maintain or upgrade the services that compose these applications. In general, Service-Oriented Architectures (SOA) promotes the reusability and interoperability of entities. In this sense, DAMASCo is a framework based on SOA, which reduces the complexity of modeling services focusing on the reuse of services accessed via their public interfaces. DAMASCo discovers the most appropriate services to satisfy a user’s request considering context information and semantic matchmaking, and, if necessary, adapts them during the composition with the purpose of performing the required functionalities. These services could serve to several applications running at the same time. Once services are composed and are being executed, failures may occur at runtime. This requires proper reactions to dynamically reconfigure the applications using the services involved in the failures. These applications are very heterogeneous and then the requirements for reconfiguration can be variable.

Software Product Line (SPL) is a very well-known approach that promotes the reuse of the core asset of a family of products, by managing the heterogeneity. At first, following a SPL approach, we extended DAMASCo to deal with the service variability during the service composition. We used Feature Models (FMs) to model the service variability at design and composition time, introducing the concept of service family. A service family includes services with similar functionalities, but slightly different at design level (e.g., services whose interfaces have distinct internal activities to implement the same functionalities).

When services fail, the management of the heterogeneity only at composition time is not enough. Then, it also needs to be managed at runtime to allow the reconfiguration of heterogeneous applications. In order to address this need, we propose to use Dynamic Software Product Lines (DSPLs), which produce software products capable of being adapted to environmental changes at runtime. In this work, the changes we consider are contingent upon whether a service is operating or not (because a failure), and the products to be adapted are the service-based applications using the services involved in the failures. Therefore, in this paper we present the extension of the DAMASCo framework, which currently does not support the service...
failures at runtime, with a dynamic reconfiguration process driven by Feature Models based on DSPLs. Therefore, we follow a DSPL-based approach to reconfigure dynamically failed services, as shown in Figure 1.

The advantages of our proposal, Dynamic DAMASCo, are the following:

- Services are grouped in families represented by a feature model. When a service fails, we will search for another service from the same family, avoiding a new discovery.
- Application reconfiguration requirements are variable, due to the heterogeneity of the applications. Then, although several applications are using the same service, when this fails, reconfiguration actions may be different for each application to satisfy its specific requirements. We deal with this aspect by using the feature models that allows finding different services for distinct requirements.
- Compose and upgrade on-the-fly services selecting concrete internal activities of other services of the family, modeled in the same feature model. This enables that when a service fails, it does not need to be replaced for a new one, instead, we can replace just some activities.

Then, the main contributions compared to previous efforts are focused not only on identifying when and how reconfiguring services, but also injecting code to replace failed services; and exploiting the service family and feature models to handle the heterogeneity of the application.

Service Reconfiguration with Dynamic DAMASCo

Following our approach, depicted in Figure 1, after a service failure occurs, the first step of this process is the Runtime Discovery, which finds, for every application affected for the failure, a new service to replace the failed service. The new service is a member of the service family of the failed service, which will be selected taking into account the different application requirements and contexts. Then, the regular discovery process does not need to be performed again. In order to perform this selection, Dynamic DAMASCo uses Hydra to generate, from the feature model representing the service family, the service configuration that satisfies the needs of every application. Then, the use of feature models also helps to manage the heterogeneity of the application requirements. The new service configuration can be different for each request where the failed service is involved. Our approach also selects the service with the maximum number of equivalent internal activities with respect to the failed service. The rationale behind of this is to be able to replace on-the-fly just some activities instead of replacing the whole service.

Then, once we have the two feature model configurations, the one that represents the service previously being executed, and the one representing the new service, we need to determine the way to achieve the New Service Configuration. In this step, detailed in Section IV.B, we obtain, at model level, the exact changes to be performed to reconfigure each application that was using the failed service. These specific modifications are calculated from the difference between the two configurations.

Finally, the modifications specified at model level have to be carried out at implementation level. This Reconfiguration Execution is performed adding and/or removing service internal activities or full services. This injection of code is done using runtime monitoring and binding processes, detailed in Section IV.C.