A Framework for Defining and Injecting Aspect-Oriented Adapters

Kardelen Hatun (ACM Member: 1203340)
University of Twente
Enschede, The Netherlands
hatunk@ewi.utwente.nl

1. MOTIVATION

In a component-based system, component binding is crucial. Such systems often use third party components, which may not provide an interface that is recognized by the base system. This requires an explicit and tailored implementation of the binding, which involves mapping of the concepts, i.e structures that represent the same information, that are common to the system and the component.

This binding is a potential cross-cutting concern. When there is a one-to-one mapping between the concepts, i.e. the concept is represented with a class or a field in both the component and the system, then the binding is merely a translation of types. However, in case of a one-to-many (when one structure representing a concept is scattered among multiple structures in the component) or many-to-one (the reverse of the former situation) then the binding concern becomes cross-cutting. This concern can be modularized by using an aspect-oriented way of defining and instantiating adapters. However current AO-languages do not offer a declarative way for describing the binding concern; an imperative programming language will lead to less readable and less maintainable implementation, which is fragile against software evolution.

Our goal is to design a framework, which introduces declarative AO-language elements that are used to define and inject adapters, and a runtime library that is responsible for adapter injection.

2. RELATED WORK

Existing aspect-oriented languages offer limited mechanisms for implementing adapters between interfaces. AspectJ [3] inter-type declarations can be used to make system classes to implement appropriate interfaces, however this approach is type-invasive. CaesarJ [1] offers a more declarative approach with *wrappers*, but their instantiation requires pointcut declarations or they should be explicitly instantiated in the base system.

In [2] an aspect-oriented framrwork called GluonJ is proposed in order to makes the relationship between aspects and aspect targets explicit, allowing better maintenance of glue code. However this approach does not specifically target component binding. In [4] introduces coordination aspects as a part of the Dynamic Aspect-Oriented Platform, which can be used as adapters between components. JasCo [5] also presents a similar approach for binding components in special structures called connectors. Both of these works present strong and influential approaches for using aspect-orientation in component binding.

3. APPROACH

Our approach is a framework which consists of a new language mechanism and a run-time library which offers a way to non-intrusively define and inject adapters. We make the declaration of adapters explicit in the aspect syntax. In an adapter declaration, the user indicates the types which they would like to adapt from and to. Then in the adapter body, an implementation is provided which defines the mapping from one type to the other. The adapters are registered by the runtime environment according to the types they adapt from and to.

Our injection component has a link to the adapter registry. This allows a flexible injection mechanism in the following way. The developer indicates that he wants to inject an object of type A to a field of type B. If the types of the object and field are incompatible, then the injection framework communicates with the adapter registry to find an *A-to-B adapter*. If such an adapter exists, the object is automatically wrapped with that adapter and the adapter instance is injected to the field. By encapsulating this logic in a framework, we allow the developer to be oblivious to adapters. During injection it is also possible to point to a specific adapter, in that case framework performs type checks to ensure the given adapter is usable and gives context-relevant warning and error messages.

4. RESULTS

Our approach aims to modularize the binding concern in a *maintainable* way. We will evaluate our approach on a realistic component-based application, by re-attaching a component with our adapter-injection framework. We will investigate how scattering and tangling is reduced with our approach. We will perform a comparative study by evolving the component and quantifying the amount of code changes that need to be made with the old binding and our modu-

larized binding. This study will help us evaluate the maintainability of our approach in the face of software evolution.

5. REFERENCES

[1] I. Aracic, V. Gasiunas, M. Mezini, and K. Ostermann. An overview of caesarj. *Transactions on Aspect-Oriented Software Development I*, pages 135–173, 2006.

[2] S. Chiba and R. Ishikawa. Aspect-oriented programming beyond dependency injection. In *Proceedings of the 19th European conference on Object-Oriented Programming*, ECOOP'05, pages 121–143, Berlin, Heidelberg, 2005. Springer-Verlag.

[3] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold. An overview of aspectj. *ECOOP 2001, Object-Oriented Programming*, pages 327–354, 2001.

[4] M. Pinto, L. Fuentes, M. E. Fayad, and J. M. Troya. Separation of coordination in a dynamic aspect oriented framework. In *Proceedings of the 1st international conference on Aspect-oriented software development*, AOSD '02, pages 134–140, New York, NY, USA, 2002. ACM.

[5] D. Suvée, W. Vanderperren, and V. Jonckers. Jasco: an aspect-oriented approach tailored for component based software development. In *Proceedings of the 2nd international conference on Aspect-oriented software development*, AOSD '03, pages 21–29, New York, NY, USA, 2003. ACM.