Tool Support for a Hybrid Development Methodology of Service-Based Interactive Applications

Christian Liebing, Marius Feldmann, Jan Mosig, Philipp Katz, and Alexander Schill

Technische Universität Dresden, Department of Computer Science, Institute for Systems Architecture, Computer Networks Group
01062 Dresden, Germany
{christian.liebing,marius.feldmann,jan.mosig,philipp.katz, alexander.schill}@tu-dresden.de

Abstract. In recent years promising approaches that provide the graphical development of service-based interactive applications were presented, but they still lack of simple platform-independent modeling and variability. We address this issue by exploiting the concept of service annotations to establish a hybrid development methodology that relies on a novel concept named temporal annotations to specify the applications navigation flow. To facilitate the development methodology, we present a graphical authoring tool.

1 Motivation

Developing service-based interactive applications manually is a time-consuming, cost-intensive and potentially error-prone task. To facilitate their development, two model driven approaches have emerged in recent years, which differ heavily in the underlying methodology and complexity of the development process.

On one hand, the Servface approach [1] follows the paradigm of service composition at the presentation layer and enables the end-user development of service-based interactive applications for a variety of platfoms by composing Web services (WS) based on their dynamically generated frontends enhanced by UI-related annotations, which can be attached to an annotable WS element.

However, the approach has several shortcomings when it is applied to more complex development scenarios. Firstly, by focusing on end-user development, the approach enables modeling of rudimentary form-based applications only. Secondly, the development methodology demands the user to select a target platform at the beginning of the development process, whereby the resulting model cannot be transformed to applications running on various target platforms.

On the other hand, there is the expressive, task-driven development approach [2] of Paterno. This approach uses the notation of ConcurTaskTrees (CTT) [3] as a technology-independent task model to specify the temporal relations between user and system tasks. A platform-independent description of the abstract User

Interface (UI) forms the starting point of the approach. This representation is transformed into various platform-dependent concrete UIs and finally mapped to an implementation specific representation. However, the practical usability of this approach is reduced due to the variety of models and the need for a manual binding of abstract system tasks and concrete WS operations.

A common shortcoming of both mentioned approaches is their lacking support for variability and the associated re-development of distincive applications.

Due to the sketched drawbacks, we present a hybrid light-weight but powerful approach that supports platform-independent, annotation-based and graphical modeling. It relies on one model, facilitates the WS binding and delivers comparable results, but does not cover all application scenarios. Subsequently, our demonstration solely shows an authoring tool, which supports the hybrid development methodology of service-based interactive applications, and does not discuss any underlying concepts in detail.

2 Temporal Annotations Plugin for Eclipse (TAPE)

Service-based interactive applications are characterized by two aspects: firstly, their functionality is entirely encapsulated behind well-defined service interfaces and secondly, graphical user interfaces enable human interactions with one or more services. Due to the application modeling on basis of WS and annotations, a suitable methodology must provide a rapid and simple applications development, customization and modification and furthermore different versions with little effort. Moreover, the expressiveness in regards of the resulting applications should fulfill state-of-the-art requirements for interactive applications.

To meet the requirements of developing service-based interactive applications and to eliminate the shortcomings of the existing approaches, we developed a hybrid approach that does not use annotations solely for describing UI-related information but also for specifying the navigation and data flow of an application. To achieve the consistent use of annotations, we extended the existing Servface annotation model [4] through the use of application-specific annotations. Initially, we analyzed the temporal relations of Paternos CTTs, whether they are potential candidates to specify the applications navigation flow and then added the specified temporal annotations to the Servface model. Due to our extensions, there is no need for a self-contained model, which stores all information.

To facilitate the development process, we developed a graphical authoring tool[1] by using the Eclipse platform and the frameworks GEF[2] and EMF[3]. We decided to develop a plugin, which has its own perspective due to the well-known look and feel that minimizes the learning curve for application developers. However, the plugin can be easily transformed into a RCP- or RAP-based application.

The internal model that is invisible from users perspective provides the distinction between UI-related and application-specific annotations to simplify the

[1] http://www1.inf.tu-dresden.de/ s6334199/tape/

[2] Graphical Editing Framework - http://www.eclipse.org/gef/

[3] Eclipse Modeling Framework - http://www.eclipse.org/emf/

independent specification. As a result, the authoring tool produces a description that may contain UI-related and applications specific rather temporal annotations and provides their transformation together with the appropriate WS descriptions into executable interactive applications for various target platforms.

The UI of the authoring tool is structured into three important areas (depicted in Fig. 1): 1. The *Service overview* shows the available services including the associated operations, 2. The *Editor view* represents the working view and consists of an annotation tool bar to specify graphically temporal relations between WS operations, and 3. The *Project explorer* provides a project overview.

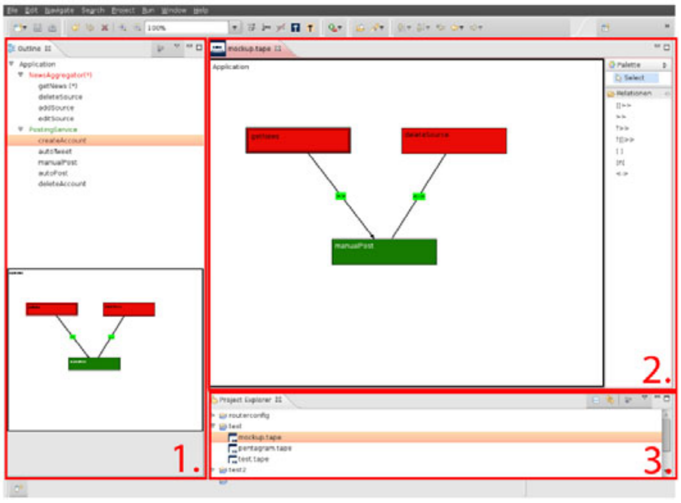

Fig. 1. Temporal Annotations Plugin for Eclipse (TAPE)

Our demonstration presents the graphical development process including the following steps to be accomplished by the user:

1. **Create a new TAPE project:** Initially, a WS has to be selected. During the modeling of the application's navigation and data flow further WS may be imported from the repository, which manages the URIs of available service descriptions. Subsequently, the initial WS operation has to be selected to specify the starting point of the application.
2. **Define the temporal relationships:** The application development starts by moving the initial operation into the *Editor*. It requires at least two operations to specify a binary annotation by using the annotation tool bar and clicking first on the left and then on the right operation. A feature for constraint checking can be activated to avoid incorrect model instances due to the fact that not all temporal annotations can be combined.
3. **Publish an application:** After modeling the whole application, the annotations can be published in the repository, which provides accessibility and

availability, and facilitates the exchange of created model instances. All annotation instances correspond to the extended Servface annotation model and may contain UI-related as well as application-specific annotations.

4. **Adapt an existing application:** To adapt an application to changing requirements, the published annotation file needs to be imported. Due to the fact that the tool does not store the layout information during the modeling process, we developed an algorithm that realizes the automatic and clearly arranged presentation of the WS operations and their temporal relationships.
5. **Generate an executable application:** By using transformation components, the application models can be transformed into executable applications for a variety of different platforms and devices.

Finally, we evaluated the scalability and usability of the authoring tool. The conducted user study with 10 participants has shown the development methodology is easily understood by developers and facilitates the development due to a clear layout and solely possesses small shortcomings with respect to the import and export of annotations.

3 Conclusion

In summary, the authoring tool demonstrates the feasibility of a hybrid lightweight development approach to create service-based interactive applications. The applied model relies solely on annotations to specify UI-related information and the navigation flow. We have shown that the hybrid approach provides the development of distinctive applications, which may differ in details with one single application model for a variety of platforms. As a next step, we plan to publish our extended Servface annotation model as well as the hybrid development methodology supporting platform-independency and variability. Future work includes the extension of the tool to cover all relevant aspects during development of service-based interactive applications and the development of code generators for mapping the modeled applications to different plattforms. Based on this we intend to focus further on the results of the generation process.

References

1. Feldmann, M., Nestler, T., Muthmann, K., Jugel, U., Hubsch, G., Schill, A.: Overview of an End-user enabled Model-driven Development Approach for Interactive Applications based on Annotated Services. In: Proceedings of the 4th Workshop on Emerging Web Services Technology, pp. 19–28. ACM, New York (2009)
2. Paterno, F., Santoro, C., Spano, L.D.: Support for Authoring Service Front-Ends. In: Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 85–90. ACM, New York (2009)
3. Paterno, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proceedings of the 6th International Conference on Human-Computer Interaction, pp. 362–369. Chapman and Hall, Australia (1997)
4. Servface Consortium: Models for Service Annotations, User Interfaces and Service-based Interactive Applications, Deliverable 2.9 (2010)