Generating consistent universal controllers for Web-Service-enabled appliances

Marius Feldmann, Thomas Springer, Alexander Schill
Technische Universität Dresden
Fakultät Informatik
Professur Rechnernetze
Germany
{marius.feldmann, thomas.springer, alexander.schill}@tu-dresden.de

ABSTRACT
Today, an increasing number of home and office appliances that we interact with contain embedded Web Servers. Making available Web Services for remote access to their functionality is just a small step. In this paper, we present a multi-platform, model-based generation approach enabling efficient and low-cost development of interactive applications for accessing Web-Service-enabled appliances. The results are not stand-alone, monolithic elements of software but are capable of being integrated into a consistent host application ad-hoc during runtime. Thus, a heterogeneous and dynamic infrastructure of appliances from different manufacturers can be accessed via single control applications. The approach has been proven to be applicable for generating interactive applications for various target platforms including mobile devices.

Keywords
Web Services, Model-based User Interface Generation, Dynamic Service Infrastructures

INTRODUCTION AND MOTIVATION
Web Services are software components offering their functionalities via a well-defined interface. This interface is described by a functional interface description language such as the Web Service Description Language (WSDL). Nowadays, Web Services are an accepted and widely used means for offering remote functionalities. Various tools are available for managing the whole lifecycle of Web Services starting from creating, via testing, to deploying and managing them. However, the field of developing user interfaces for Web Services has not been covered in a satisfactory manner yet. Though some approaches have been specified during the last years enabling the development of user interfaces for static Web Service infrastructures, no approach exists that makes it possible to extend an interactive application fully automatically depending on the associated set of Web Services the application interacts with. The necessity for such an interactive application becomes obvious in the case of the selected use case: Within an automated home, a set of appliances can be controlled via a universal control application available on different platforms. In the sketched scenario in Figure 1, three appliances (Light Control System, Music Control System and a Router) can be accessed via their offered Web Services by using either a universal control application for a mobile device or a Web-based universal control application. If a new device (DVB-T device) is added to the device infrastructure, a user interface has to be made available dynamically within the two universal control applications. Developing these user interfaces for various target platforms usually tends to constitute a time consuming and expensive task.

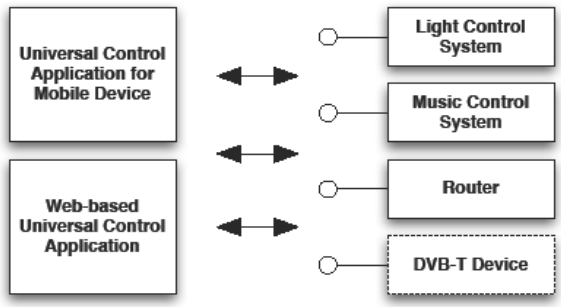

Figure 1. Sketch of the home appliance scenario

In order to solve these problems, this paper presents an approach to generate the user interfaces of the Web-Service-enabled appliances fully automatically from functional interface descriptions. Due to the fact that without further input, the generation result would be of low quality, so called Service annotations [8] are applied as platform independent modeling approach for defining user interfaces.

The results of the generation chain named **interactive components** can be embedded in the universal control applications dynamically during runtime. Without stopping this application, they appear in their navigation menu and can directly be used to remote control the appropriate device. Due to the possibility to parameterize the generation of interactive components by using layout and style descriptions, they can be visually adapted to different appearances of the embedding universal control applications. Thus, a consistent visual representation is achieved.

The remainder of this paper is structured as follows. In the second section an overview of the state of the art in the addressed domain is provided. The third section defines the concept of interactive components and points out their central characteristics. In the fourth section the assumed development approach is described. The fifth section

M. Feldmann, T. Springer, A. Schill: Generating consistent universal controllers for Web-Service-enabled appliances. Proc. of 1st International Workshop on Model-based Interactive Ubiquitous System 2011, Pisa, Italy, June 13, 2011, http://ceur-ws.org/Vol-787

introduces the steps for generating interactive components from the results of the development approach. The sixth section describes the technical realization of the mentioned appliance control scenario and an end user study applied using the prototypical implementation. The paper is concluded by a summary and outlook on future work.

STATE OF THE ART

Creating user interfaces for home appliances is a topic covered in the HCI community already for years (e.g. [3]). As well the specific goal to create consistent user interfaces for controlling different devices has been addressed [4]. However, these approaches do not focus in any way on Web-Service-enabled appliances. Thus, the possibility to generate user interfaces from functional interface descriptions is not discussed. The only approach exploring Web Services in the field of universal appliance control applications exploits a task-driven development approach to create consistent user interfaces [5]. Though the efficiency of this approach may be increased by using Service annotations [6], it still takes various manual steps to develop a user interface for different target platforms. Furthermore, the resulting user interfaces cannot be embedded automatically into a host application during runtime. Extending the focused research and development field to ad-hoc UI generation approaches for Web-Service-based interactive applications, we discovered a set of ten approaches. They usually take either a functional interface description or additionally Service annotations as input and create stand-alone markup-based UIs as output. An example for this category of approaches is described in [7]. The results have a low quality and do not address the generation of consistent UIs enabling the interaction with various underlying Web Services as it is demanded within the home appliance scenario.

To summarize our efforts, we realized an intensive state-of-the-art analysis searching for approaches that make it possible to generate automatically user interfaces for Service-based interactive applications and that create results which can be included ad-hoc into a running host application. To the best of our knowledge, no comparable approach like the one presented in this paper exists.

INTERACTIVE COMPONENTS

Interactive components are a domain-specific solution for user interfaces of Service-based interactive applications. Due to their characteristics, they solve the identified problems of dynamic integration of ad-hoc generated user interfaces into a hosting interactive application.

Interactive components contain all information necessary for registration purposes within a host application. They offer a user interface for providing input values and displaying output values of Web Service operations and contain all information needed for enabling the interaction of a human user with a remote Web Service. After an interactive component has been generated, it is transmitted to a host application (such as the universal appliance control application) via an application specific directory Service.

As soon as an interactive component is physically available on a target device, it runs through the following steps:

1. **Registration:** The interactive component communicates the information needed for its execution to the host application. The registration information covers:
   a. **Identifier**: In order to identify a component and to replace old versions on a target platform, a unique identifier is assigned to every interactive component.
   b. **Component Dependencies**: An interactive component may demand functionality that has to be provided by the host application. This includes for example functionality to determine the current geolocation or specific data storage facilities.
   c. **Input-/Output-Data**: An interactive component may interact with a host application via exchanging data. To identify the data needed as input and returned as output, references to data described within the associated functional interface description is used.
   d. **Context information**: Every interactive component may be generated for a concrete usage context such as a specific language, a specific target platform or a selected geographical location. A context description is expressed as a set of context type and value information.
   e. **Information for visualization purposes**: An interactive component is included dynamically into the navigation menu of a host application so that a user can activate it. For visualizing the component, data such as a label text or an icon is communicated to the host application.

   If a host application accepts the registration information and fulfills the requirements (thus it can offer the necessary components (b) and the appropriate input data (c)), the registration is successful. If the interactive component is not registered successfully, it is removed from the host application.
2. **Activation**: After an interactive component has been registered successfully and the current application context conforms to the context the component is intended for, the component can be activated by a user. The host application hands over the input parameters demanded by the interactive component during the registration procedure.
3. **Usage:** After a successful activation of the interactive component, a user can input data via the user interface and invoke remote Web Service operations. The operation results are being visualized in the user interface.
4. **Deactivation:** The interactive component can be deactivated at any time by the host application or the user. After the deactivation has been triggered, its user interface is not visible anymore. During deactivation of

an interactive component, it returns the output data specified during the registration process to the host application.

We have specified a Meta-model named Service-based Interactive Component Model (SBIC) for representing interactive components during the generation process. The structure of the model is sketched in Figure 2. The SBIC has been developed as a model representing an interactive component in a way close to a runtime representation. Thus, a model-to-code transformation mapping a SBIC instance to a platform specific interactive component can be implemented with little effort.

Besides the registration information, the SBIC describes the structure, navigation flow and data flow of the user interface. Furthermore, it contains layout and design information that is weaved into the model during the generation process. Thus, the layout and design of a generated interactive component can be adapted to the layout and design of a host application.

DEVELOPMENT APPROACH

As aforementioned in the state-of-the-art analysis, the development of Service-based interactive applications for various target platforms is currently a time-consuming task involving huge manual effort. Our approach is characterized by the novel idea to only use Service annotations for modeling user interfaces. As the state-of-the-art does not offer any Service annotation model enabling the specification of application-specific aspects such as navigation or data flows, we introduced new annotation types.

To not reinvent the wheel, these types were introduced into the annotation model originating from the ServFace project [8]. Besides others, the following annotation types have been specified and included into an extension of the ServFace annotation model:

1. **Navigation Flow:** Enables the possibility to define a navigation flow between the user interfaces derived from different Service operations. With every specified navigation flow, a data flow may be associated.

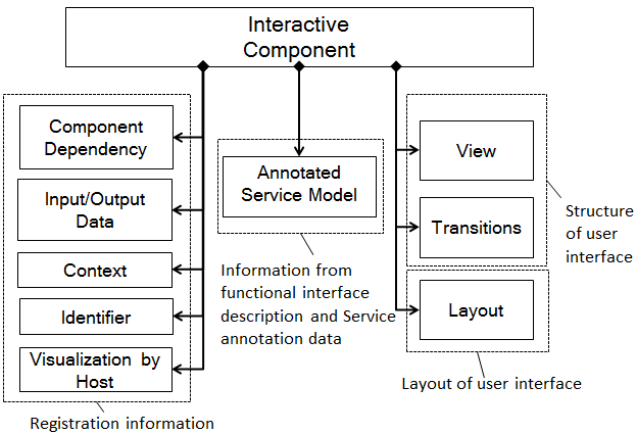

Figure 2. Structure of interactive component Meta-model

2. **Bundling:** Renders it possible to merge the user interfaces of different operations into one user interface. Furthermore, the view for input and output of one operation might be merged in one view.
3. **Provided by Host:** Marks an input parameter of a Service operation as input to the interactive component provided by the host application.
4. **Returned to Host:** Marks output parameters to be returned to the host application when the interactive component is deactivated.
5. **Component Dependency:** Defines dependencies to functionalities the host application has to provide.
6. **Initial Operations:** A subset of the operations defined in a functional interface description may be marked as initial. These operations can be accessed directly from the host application. Every initial operation is an entry point to activate the interactive component. After activating it, the user interface for the appropriate initial operation is visualized.

In order to specify these annotations, an authoring tool named Interactive Component Editor (ICE) [2] has been created. One of the views of the ICE is shown in Figure 3. After importing one or more functional interface descriptions into the tool, the tree structure of these descriptions is visualized. The nodes on the various levels (Service, operation, input/output parameters, data types) of the tree representation can be included into the modeling of the interactive component.

The screenshot in Figure 3 shows various navigation flows and data flows in a specific editor view during modeling an interactive component for a DVB-T device. The editor makes it possible to specify any of the annotations defined within the ServFace project and of the abovementioned extensions. Thus, a full featured development approach has been created enabling the model-based specification of interactive components solely with annotated functional interface descriptions.

After the modeling has been realized, the resulting annotations are serialized to a file and transferred to a remote annotation repository.

GENERATION APPROACH

The generation of interactive components is triggered on demand once an interactive component has to be embedded into a host application. In the case of the automated home scenario, the generation chain is activated as soon as a new device and thus a new annotated Web Service appears in the home infrastructure. Due to the usage of platform independent Meta-models and the platform specific parameterization of the generation approach, it can be reused for various target platforms and different host applications. The platform and host specific parameters have to be provided only once for every host application. The target specification can be reused during the generation of each interactive component that should be embedded into this host application.

The generation chain is summarized in Figure 4.

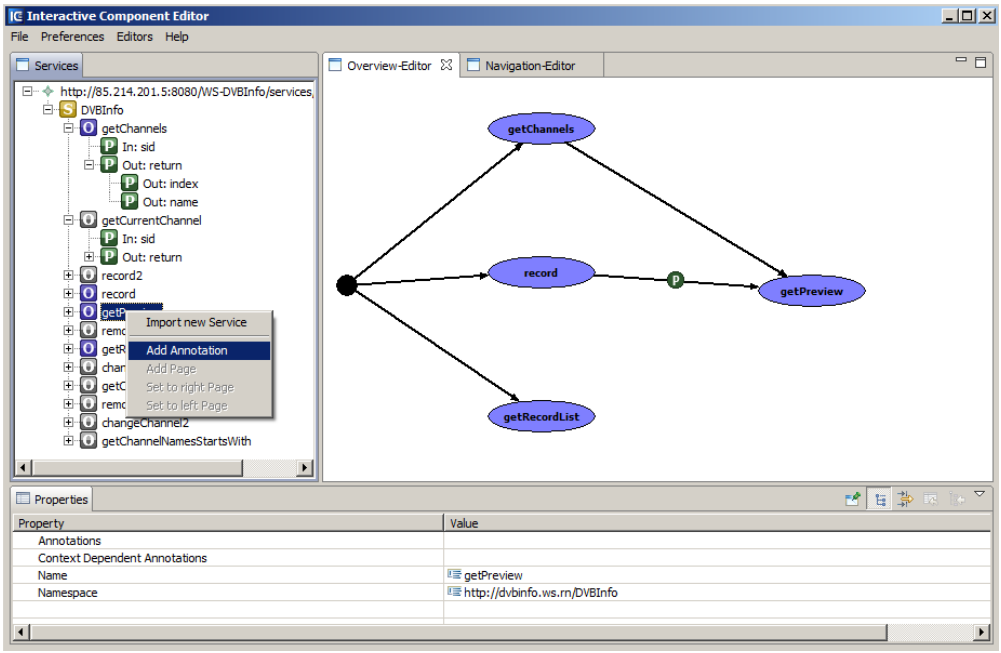

Figure 3. Screenshot of the Interactive Component Editor

In a first step, the functional interface description and the annotation file fetched from an annotation repository are parsed and merged together into a common model. For this purpose the Annotated Service Model (ASM) has been developed. It enables the representation of all ServFace annotations and the mentioned extensions. An ASM instance contains, on the one hand, the tree representation of the functional interface description with the different hierarchies (Service, operation, input/output parameters, data types) and, on the other hand, the annotations referencing a subset of the tree's nodes. This subset has been determined during the development approach applied e.g. by using the ICE.

After the ASM has been instantiated, a basic structure of the user interface with all its interactors is inferred in a first model-to-model transformation step. For representing the structure of the UI, a domain-specific model named Intermediate Service Frontend Model (ISF) adapted to the generation chain has been developed.

For instantiating the ISF, the ASM instance is traversed. By default for every Service operation one container for all input parameters and one container for all output parameters are derived and embedded into the ISF instance. This default behavior may be modified by annotations of the *Bundling* type (see previous section).

In a next step, appropriate interactors are derived from all input and output parameters passed during traversing of the ASM instance. They are embedded into the containers created in the previous step.

As the selected interactors are described within the ISF using platform specific vocabulary, this step is parameterized by a set of inference rules which determine which interactor should be selected under which condition. Besides the data types of an input or output parameter, the decision depends particularly on the set of annotations referencing this parameter.

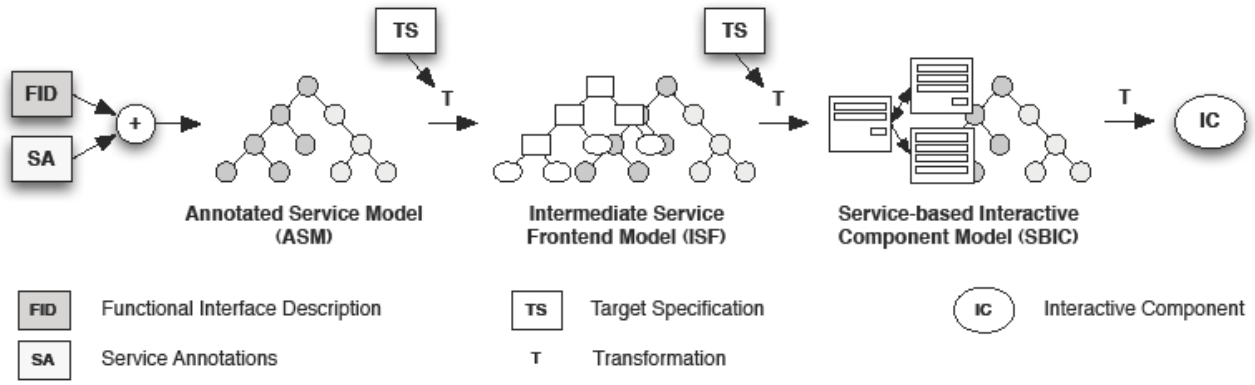

Figure 4. Generating interactive components from annotated functional interface descriptions

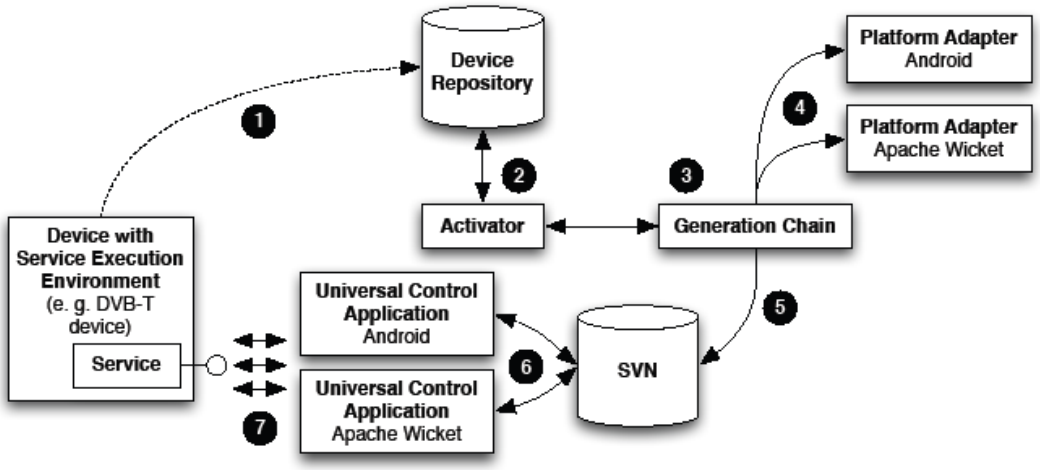

Figure 5. Usage of the generation approach within the home appliance scenario

In a second model-to-model transformation step, an instance of the SBIC Meta-model mentioned above is created from the ISF instance. Just as the transformation steps before, the implementation of this step can be reused for various target platforms due to the parameterization of the transformation. The provided parameters contain especially layout and design information for adapting the appearance of the interactive component to the appearance of a host application. This information can be made available a-priori and updated as soon as the layout and design of the host application is modified.

Finally, every SBIC instance is transformed to a platform specific interactive component that can be made available via a component repository to a host application. For every supported platform a model-to-code transformation has to be provided that takes an SBIC instance as input and returns a packaged and deployable interactive component as output.

The described generation chain builds the core of the home appliance scenario introduced above. Figure 5 shows the different steps used to make interactive components available in different universal control applications.

The approach runs through the following seven steps:

1. A device (e.g. a DVB-T device) is newly introduced into the infrastructure of Web-Service-enabled appliances. As soon as it is switched on, it is registered by transmitting the Uniform Resource Identifier (URI) of its functional interface description plus a reference to an annotation file to a device repository.
2. The device repository is monitored by a specific system component named Activator. As soon as this component detects a new device registration, it forwards the functional interface description referenced by the URI plus the annotation file to the generation chain described above.
3. The generation chain parses the functional interface description and the annotation file and transforms it first to an ISF and then to a SBIC instance.
4. Using platform-specific model-to-code transformations (named platform adapters) for all desired target platforms, the SBIC instance is transformed to platform specific interactive components.
5. These components are forwarded to a repository that is checked in fixed intervals by the used universal control applications for updates.
6. As soon as a new interactive component is available within the repository, it is downloaded to the target platform and registered in the universal control application.
7. After a successful registration of an interactive component it can be used to interact with the remote Web Service thus enabling device control.

REALIZATION AND EVALUATION

The approach has been implemented and evaluated using two heterogeneous target platforms as shown in Figure 5. As proof-of-concept for mobile devices, a universal control application and a platform adapter for the smartphone platform Android has been realized and as proof-of-concept for Web applications, the same parts of software have been implemented using the Apache Wicket framework. The repository for devices has been realized as a simple relational database. As technology for the repository for interactive components Subversion (SVN) was used. SVN has the central advantage that no additional mechanism for versioning of interactive components had to be introduced.

Figure 6 shows a screenshot of the prototypically implemented Web-based universal control application. As it is depicted in the figure, interactive components are embedded once they are registered in this application into its navigation menu shown on the left side. The initial operations (see fourth section) of each interactive component are displayed by a navigation option each. As soon as a user selects one of these options, the appropriate user interface is displayed in the center of the Web page.

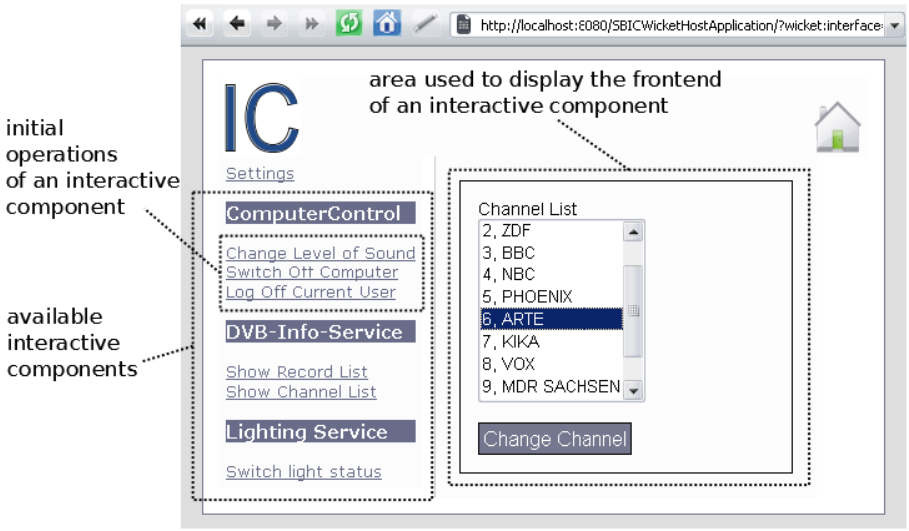

Figure 6. Prototype for the Web-based universal control application

The approach has been evaluated based on the prototypical implementation within an end user study. 25 people in the age between 19 and 31 years have used the two universal remote controllers to interact with a set of simulated home appliances. After fulfilling several minor tasks, the users participated either in a questionnaire or in an interview to get an insight into their experiences with the generated interactive components. Summarizing the results, the interactive components have been reviewed as easily and efficiently usable. All users stated that the user interface makes a consistent impression and looks like a monolithic application. No user criticized any form of inconsistency of the user interface. Thus, the component-based nature of the user interface is not recognized by end users.

SUMMARY AND OUTLOOK

This paper presented an approach to generate consistent universal controllers for Web-Service-enabled home appliances fully automatically. The approach is based on the idea to generate so-called interactive components from annotated functional interface descriptions. These components can be embedded dynamically during runtime into a host application. The applicability of the approach has been demonstrated by implementing the home appliance scenario using a Web-based universal control application and an appropriate application for a smartphone platform. The implementation has been evaluated in an end user study which confirmed the good quality and usability of the resulting user interfaces and their consistency. The control application made the impression to be monolithic though it consists of components provided for every available device.

Future work will focus in first place on improving the underlying development approach sketched in the third section. In this area it will be analyzed, how the provisioning of Service annotations may be simplified, e.g. by suggestion functionalities. Furthermore, implementing support for further target platforms is intended.

REFERENCES

1. Christensen, E., Curbera, F., Meredith and G., Weerawarana, S. Web Services Description Language (WSDL) 1.1. W3C Note. March 2011.
2. Feldmann, M., Martens, F., Berndt, G., Spillner, J., Schill, A. Rapid Developed of Service-based Interactive Applications using Service Annotations. In: Proceedings of IADIS International Conference WWW/Internet 2010.
3. Nichols, J. Automatically generating high-quality user interfaces for appliances. In CHI ’03 extended abstracts on Human factors in computing systems, pp. 624-625, ACM Press, 2003.
4. Nichols, J., Myers, B. A., Rothrock, B. UNIFORM: Automatically Generating Consistent Remote Control User Interfaces. In Proceedings of CHI'2006, pp. 611-620, Montreal, Canada , 2006.
5. Paternò, F., Santoro, C., Spano, L., D. Designing usable applications based on Web services. In: 1st Workshop on the Interplay between Usability Evaluation and Software Development, pp. 67 - 73. CEUR, 2008.
6. Paternò, F., Santoro, C., Spano, L. D., Exploiting Web service annotations in model-based user interface development. In: EICS'10 - 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 219 - 224. ACM, 2010.
7. He, J., I. Yen, T. Peng, J. Dong und F. Bastani: An Adaptive User Interface Generation Framework for Web Services. Proceedings of IEEE Congress on Services Part II, Seiten 175–182, 2008.
8. ServFace-Konsortium: *Deliverable 2.9 - Models for Service Annotations, User Interfaces, and Service-based Interactive Applications (final version).* Technical Report, SAP AG, Lyria S.A., Consiglio Nazionale Delle Ricerche, The University of Manchester, Technische Universität Dresden, 2010