A component library to improve the reusability in the development of converged services

Laura Díaz-Casillas
Departamento de Ingeniería de Sistemas Telemáticos
Univ. Politécnica de Madrid
Avenida Complutense 30, Madrid (Spain)
ldcasillas@dit.upm.es

Carlos A. Iglesias
Departamento de Ingeniería de Sistemas Telemáticos
Univ. Politécnica de Madrid
Avenida Complutense 30, Madrid (Spain)
cif@dit.upm.es

Miguel Nieto
Departamento de Ingeniería de Sistemas Telemáticos
Univ. Politécnica de Madrid
Avenida Complutense 30, Madrid (Spain)
mnieto@dit.upm.es

ABSTRACT

The evolution of communications networks to Next Generation Networks (NGN) has encouraged the development of new services. Nowadays, several technologies are being integrated into telecommunications services in order to provide new functionalities, resulting in what are known as converged services. The objective is to adapt the behavior of the services to the necessities of different users, generating customized services.

Some of the main technologies involved in their development are those related to the Web. But due to this type of services implies the combination of different technologies, their development is a very complex process that has to be improved to reduce the time and cost required, with the aim of promoting the success of such services.

This paper proposes to apply software reuse through the utilization of a component library and presents one focused on ECharts for SIP Servlets (E4SS). It is a framework, based on the SIP Servlet specification, which uses finite state machines for the definition of converged communications services. Also, to promote the use of the library, a methodology is proposed in order to facilitate the integration between the library operations and the software development cycle.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Communications Applications; D.2.13 [**Reusable Software**]: Reusable libraries

General Terms

Management, Theory

Keywords

software reuse, services, components, library, SIP, E4SS, ECharts

1. INTRODUCTION

The development of converged services is a complex task, requiring the knowledge and integration of different technologies. And the time employed, i.e. time to market, is decisive to the success of the services.

Currently, there are various standards to facilitate the development of these services, such as OSA/Parlay [14], JSLEE (JAIN Service Logic Execution Environment) [4] and SIP (Session Initiation Protocol) Servlets [8]. Also, several initiatives have appeared recently with the purpose of accelerating this process, such as SailFin CAFE [10] and ECharts for SIP Servlets (E4SS) [2], providing frameworks to implement telecommunications services based on SIP [13]. But these frameworks are still under development, being necessary to offer new mechanisms through which facilitate the task of developing converged services.

TelcoBlocks [5] is a research project which aim is to provide an open source platform for the development and deployment of components of telecommunications services based on VoIP. In the context of this project, this article tries to promote the software reuse with a library of components focused on the implementation of converged services with E4SS.

Software reuse reduces the development and maintenance time and cost, improves the quality and reliability of the software and the efficiency in the use of the available resources, increasing the productivity. But, in order to reach these objectives, reuse has to be a consistent and repeatable process, i.e. a systematic reuse, included into the software development cycle [12]. The origin of this type of reuse goes back many years [7], but it has not a widespread use due to it demands an extra effort compared with the normal development process. Systematic reuse is an abstract activity that requires a high-level knowledge that usually comes from experience. Also, other factors have to be taken into account, which depend on the developers, the working environment, and external parameters, such as the characteristics of the market and the domain in which the application to develop will be applied.

Libraries of components are one of the main tools used to promote the software reuse. They facilitates the management of a set of elements, enabling developers to store and retrieve components with a specific functionality in a high-level way. Libraries of components have been applied to several areas, proving their efficiency to manage different kinds of elements. Mili et al. present a software library classification [9] and evaluate various libraries using technical, managerial, and human criteria. They conclude that most of the libraries are focused on a very specific objective, but are too inaccurate to be useful or too intractable to be usable. In order to avoid this problematic, we present a component library related to a very specific area, the development of converged services using ECharts technology, but with a set of operations and a related methodology with the purpose of improving its use.

The remainder of this paper is structured as follows. Section 2 explains more in detail the characteristics of E4SS, the selected technology for the implementation of the services in the context of the TelcoBlocks project. Then, section 3 describes the library of components proposed and section 4 explains a methodology about how to use it to get optimal results. Finally, section 5 presents the conclusions and future lines of work.

2. ECHARTS FOR SIP SERVLETS (E4SS)

2.1 ECharts

ECharts [1] is a programming language based on state machines derived from UML. ECharts is a high-level language that depends on a lower level language, for example, Java.

State machines consist of a set of states and transitions between them. ECharts presents two basic types of state machines, `OR` and `AND`, for one or more than one active states simultaneously, respectively. Also, there are `MIXED` machines, which contain both types of machines. Transitions can occur between states of the same machine or different machines. And it is possible to include predicates, so changes will take place only if the specified conditions are satisfied, and actions, with tasks to execute during the transition. In addition, states can contain input and output actions, that will be executed when the related state will be the destination or the origin of a transition, respectively. Moreover, each machine can present one or more constructors, which can include actions too. In order to react to events that occur in the environment, machines receives messages in their ports. These messages may come from outside or inside of the machine. Also, ECharts is able to generate automatically a diagram related with a machine, showing the states and transitions and the relations between them in a graphical way.

2.2 E4SS

E4SS combines the potential of ECharts with the SIP Servlets interface in order to facilitate the development of telecommunications services.

In E4SS, each application presents a `FeatureBox`, a container where the elements required for its instantiation are located. Specifically, these elements are a state machine, to specify the application behavior, and a set of ports, to interchange messages. There are four types of ports, dedicated to manage SIP messages (`SipPort`), initial requests (`BoxPort`), time events (`TimerPort`), and other types of messages (`NonSipPort`).

With the aim of promoting the development of converged services, in which SIP applications are integrated with other components of different nature, E4SS defines two types of interfaces: `SIP-To-Java` and `Java-To-SIP`. The use of one or the other depends on the origin of the interaction. For example, if a SIP component wants to store some information in a database, it will use a `SIP-To-Java` interface, while if a Web service wants to interact with a SIP component, a `Java-To-SIP` interface will be utilized. In addition, E4SS provides discovery mechanisms for managing these components.

Applications can be combined using the application router or DFC (Distributed Feature Composition). Following this approach, each application is independent of the others, but it is possible to define a series of filters according to the origin and destination SIP addresses to determine which one has to be executed in each case. The router will be in charge of determining the order of execution to achieve the desired functionality.

3. LIBRARY OF ECHARTS COMPONENTS

The objective of defining a component library is to facilitate the development of applications, reducing the complexity of the process through the encouragement of component reuse. To reach this objective, the library provides access to components, establishing mechanisms to describe and select them, and also to promote their evolution, due to new components are easily integrated with the rest.

Using ECharts to implement an application improves the management of its complexity and promotes the reuse of its components. Moreover, E4SS presents a set of initial state machines that perform basic functionalities in order to manage SIP-based communications, and some complex state machines, also called features, which make use of the basic ones to implement higher-level functionalities.

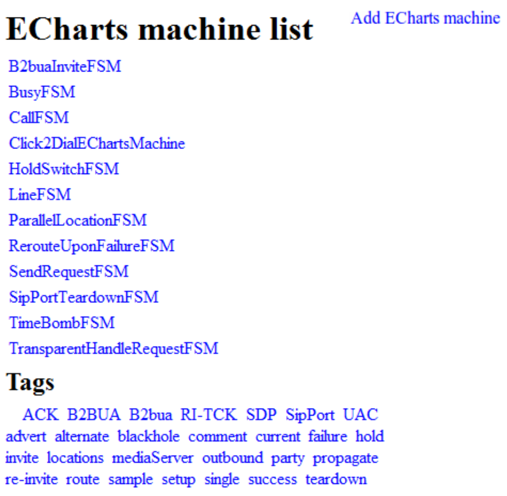
But finding the appropriate component to the requirements of each case is a difficult task. It involves knowing the specific characteristics of each component and the environment in which is going to be applied, being necessary to have experience in the use of the related technology. Using a component library promotes the reuse of its elements, due to it facilitates their management, their selection and also the inclusion of new components, created during the development of the applications.

The operations supported by the proposed library are explained in the following sections.

3.1 Describing components

Firstly, it is necessary to analyze the properties of the components of the library, in order to identify the most relevant parameters to characterize them. In this case, these parameters are:

- Name, identifies the component.
- Author, indicates the developer of the code.
- Version, facilitates the management of multiple variants of the same component.
- Description, explanatory text about the component.
- Tags, keywords that define the functionality performed by the component.
- Dependencies, list of the components used by the component.
- Nature, indicates the component type: feature or basic.


EСharts machine list

Add ECharts machine

B2buaInviteFSM
BusyFSM
CallFSM
Click2DialEChartsMachine
HoldSwitchFSM
LineFSM
ParallelLocationFSM
RerouteUponFailureFSM
SendRequestFSM
SipPortTeardownFSM
TimeBombFSM
TransparentHandleRequestFSM

Tags

ACK B2BUA B2bua RI-TCK SDP SipPort UAC advert alternate blackhole comment current failure hold invite locations mediaServer outbound party propagate re-invite route sample setup single success teardown

Figure 1: List of state machines and related tags

- Application, according to their functionality, represents the work area in which the component is involved.

This information is indicated by annotations, which are included in the source code of the component, allowing an automatic processing.

3.2 Adding components

After describing a component, it has to be added to the library. For this purpose, the user will upload the corresponding file, which will be processed to extract the information and inserted into a database, allowing future references. Moreover, there is a state diagram related with each component that shows its functionality. This diagram is generated automatically from the information contained in the description file and through a tool provided by E4SS. If a component depends on others, these will be required to generate its diagram.

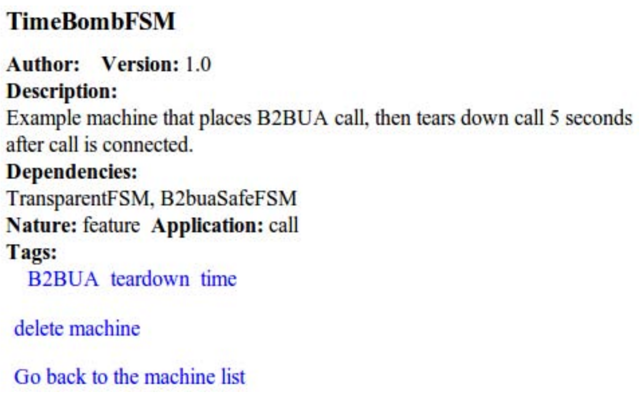
3.3 Searching components

Initially, the library displays a list of the names of the components that have been stored. These names correspond to links that allow access to detailed information about each of them. In addition, the library shows a list of keywords related to the components that can be used as a search tool to facilitate their location. A specific keyword can describe more than one component, in this case, a list with the name and the version of each result will appear, and the user will be responsible for analyzing their characteristics and select the one that best matches the needs of the application to develop.

Figure 1 presents a list of the available machines and their keywords or tags. The information displayed after accessing one of them is reflected in figure 2 (the related diagram has been omitted).

3.4 Updating components

In most of the time, the development of an application involves to implement new components and modify existing ones, being necessary to update the library. For this purpose, the user will have to specify the new values of the characteristic parameters of the component, i.e. name, version,


TimeBombFSM

Author: Version: 1.0
Description:
Example machine that places B2BUA call, then tears down call 5 seconds after call is connected.
Dependencies:
TransparentFSM, B2buaSafeFSM
Nature: feature Application: call
Tags:
B2BUA teardown time

delete machine

Go back to the machine list

Figure 2: Description of the state machine TimeBombFSM

author, and other annotations that describe it, modifying the file that contains its source code. And then, add this file to the library, following the procedure described in section 3.2. The file will be processed to extract the information and inserted into the database.

3.5 Deleting components

Moreover, the library supports the deletion of components. Figure 2 shows how there is a button related to each component to enable the remove of its information. Although the state of the original file remains.

In addition, it is necessary to consider the connections between the components available in the library. For this aim, Velasco Elizondo and Lau propose a list of high-level connectors through which encourage the reuse of elements [3]. With E4SS, this functionality is provided by the application router, which facilitates the interconnection between applications. And at a lower level, the library shows the dependencies between components.

The library has been developed using Python 2.6 [11], in combination with the framework Bottle 0.6.4 [6], to implement a Web interface to enable its access, and SQLite 3, to support the storage of the information.

4. USING THE COMPONENT LIBRARY

A library of components can facilitate the development process, but having such tool does not ensure a good use of it. In order to improve its utilization, it is advisable to follow a methodology that sets out the steps to take to obtain optimal results.

In the context of the development of converged telecommunications services, we propose to follow the next actions:

1. Component Search. Firstly, a list of keywords will be extracted from the description of the application. It will be used to search the most appropriate components to build the application.

2. Component Selection. Then, the characteristics offered by each component will be analyzed in order to select the most suitable to fulfill the requirements of the application.

3. Component Adaptation. Selected components will be modified in order to adapt their functionalities to the application under development.