ZenFlow: A Visual Web Service Composition Tool for BPEL4WS*

Alberto Martínez, Marta Patiño-Martínez, Ricardo Jiménez-Peris, Francisco Pérez-Sorrosal
Facultad de Informática
Universidad Politécnica de Madrid (UPM), Spain
{amartinez,fpsorrosal}@zipi.fi.upm.es, {mpatino,rjimenez}@fi.upm.es

Abstract

Web services have become a very powerful technology to build service oriented architectures and standardize the access to legacy services. Through web service composition new added value web services can be created out of existing ones. Examples of these compositions are virtual organizations, outsourcing, enterprise application integration, business process definitions and business to business inter/intra-enterprise relationships. In order to enable the construction of business processes as composite web services, a number of composition languages has been proposed by the software industry. However, the handiwork of specifying a business process with these languages through simple text or XML editors is tough, complex and error prone. Visual support can ease the definition of business processes. In this paper, we describe ZenFlow, a visual composition tool for web services written in BPEL4WS. ZenFlow provides several visual facilities to ease the definition of a business process such as multiple views of a process, syntactic and semantic awareness, filtering, logical zooming capabilities and hierarchical representations.

1. Introduction

Service oriented architectures (SOA) define systems that allow the linking of resources on demand. SOA resources are available in the network as independent services. This provides a more flexible loose coupling of resources than in traditional systems architectures.

Web service technology is nowadays the most well-known connection technology for implementing SOA. Web services allow loosely coupled interaction between services implemented by different vendors. This is possible because the interface provided by web services is defined in terms of an XML-based language called WSDL [18]. Today, business to business (B2B) relationships are one of the most important areas for companies. If companies expose their services through web service technology, these services may be integrated in business processes and B2B relationships, defined inside and outside organizations. However, it is not easy to define a business process out of invocations to web services. In order to abstract the applications from the structure of processes, some kind of composition language for the business processes definition is needed. In the last few years, the software industry has proposed several XML-based composition languages for web services: BPML [3], XLANG [12], WSFL [9], WSCDL [17], BPEL4WS [8], ... But, even with these composition languages, it is still hard for designers to define business processes by hand with the only help of an XML editor.

In this paper we present ZenFlow, a visual web service composition tool. ZenFlow is a full system for the composition and execution of web services. This paper focuses on the visual facilities to navigate and edit business processes. The visual composer of ZenFlow is a multi-view graphical editor that is aware of the syntax and semantics of the composition language, BPEL4WS. The multiple views enhance the productivity of process designers by enabling the editing and navigation on the most appropriate view. All the views are synchronized and have the same notion of the editing focus. The user of ZenFlow may choose to filter some process elements in order to simplify the visual representation of a process. Zoom facilities and hierarchical representation of a process are also available. The language awareness enables the ZenFlow composer to enforce the syntactic integrity of the process being edited. ZenFlow supports sophisticated visual editing operations that guarantee process consistency throughout the editing process. For instance, parts of a graphical process may be cut and pasted somewhere else. Both the cut and paste operations check the syntax before being effective in order to prevent syntactic errors. The visual composer has been designed to enable both bottom-up and top-down web service composition. In the bottom-up approach the designer builds the pro-

*This work has been partially supported by the European Commission under the Adapt project grant IST-2001-37126 and by the Spanish Research Council (MEC) under grant TIN2004-07474-C02-01.

cess step by step adding web service invocations till completing the process definition. Our visual composer eases this task through a visual UDDI browser that can connect to any UDDI repository (standard web service directory) through its web service interface. The UDDI browser allows the visual navigation of the contained web services. In the top-down approach the designer starts from a process only representing the choreography of external web services and enriches it with the internal business logic.

The rest of paper is structured as follows. Section 2 presents an introduction to web service technology and composition of web services. A summary of the features of the composition language BPEL4WS is found in Section 3. Section 4 presents a description of the visual composer of ZenFlow. Section 5 compares ZenFlow with related approaches. Finally, conclusions are presented in Section 6.

2. Web Service Technology

A web service is a self-described application that can be published and accessed through the web. The basic building blocks of web services are WSDL, SOAP and UDDI.

The description of a web service is done using the WSDL language [18]. WSDL exposes the service interface. SOAP [19] provides the communication layer based on XML to access web services. Publication and discovery of web services are made through UDDI repositories [16]. An UDDI repository is a web service catalogue. There are several toolkits to develop and deploy applications as web services, like AXIS[1], Java WSDP[15] or .NET[11]. Legacy services can be wrapped using any of these systems in order to provide a web service interface.

Web services can be either basic or composite. Basic web services do not rely on any other web service. On the other hand, composite web services invoke other web services. They are structured in a workflow-like fashion. Composite web services are used among other things to model business processes. This is a graph structure where the nodes are modelled as activities. The arrows of the graph represent the relations between the activities. Activities represent tasks to be performed in the process.

Business processes may become fairly complex. They may involve many activities and complex relationships between them. Moreover, business processes usually have to be updated and evolve over time in order to adapt to changing requirements. In general, process designers only have XML editors to maintain the business processes defined by means of XML-based composition languages. Without the appropriate tools, the task of maintaining a business process may become a tough task.

3. BPEL4WS

The Business Process Execution Language for Web Services (BPEL4WS or just BPEL) is a language for business process specification based on web service composition. Today, BPEL is a de facto standard language for web service composition that is powerful enough to define complex business processes [21].

A BPEL process defines the implementation of a web service that can invoke other web services. There are two main parts in a BPEL process description: the *definitions* part and the *process description* part. The definitions part exposes a WSDL of the process being defined. So, the process is a web service. In this section, the relationships between the process and other web services are described. A BPEL process uses partners to represent web services a process interacts with. The roles played by each service are defined using *service link types*.

The process description section defines the structure of a BPEL process. The process itself is specified by means of its constituent activities. The activities can be simple (cannot be decomposed) or complex (allow the nesting of other simple or complex activities).

Simple activities allow the invocation of an operation on a web service (invoke), processing an operation invocation (receive), generate a response for an operation (reply), copy data from one place to another (assign), exception raising (throw), wait an interval of time (wait) and terminate the process instance (terminate). On the other hand, complex activities allow the definition of a sequence of activities (sequence), case-like selection (switch), select-like multi-reception (pick), loops (while), or concurrent activities (flow). BPEL allows the nesting of activities (scope) inside a process.

BPEL provides two mechanisms to deal with errors during the process execution: exceptions and compensations. Exceptions are treated by means of the throw-catch blocks, like in any other programming language. A compensating action allows the process designer to implement actions that make amends for the previously committed actions.

4. The ZenFlow Toolkit

The ZenFlow toolkit enables the visual composition and execution of business processes based on BPEL. The visual composer and the engine supporting the deployment and execution of BPEL processes are both implemented in Java. The visual composer can be run either as a standalone Java application or as a plugin in the Eclipse platform [5]. The visual composer is enriched with BPEL and WSDL compilers as well as the corresponding code generators to fully support BPEL processes.

4.1. Environment Overview

The programming environment provided by the ZenFlow visual composer looks like a classical programming environment except in that it supports visual representation of the business processes (Fig. 1). In addition to the menu and toolbars located on the top, there are three main working areas (panes): the file explorer, the process design area and the error pane. The file explorer is usually located on the left side of the window. The file explorer shows all the BPEL business processes currently open. The designer can change the current process by selecting it in this view. All other views are updated accordingly. Each BPEL *syntactic element* has an icon that identifies it (BPEL visual element). The toolbar offers a palette of BPEL visual elements. Each visual element has an associated form with its definition.

The process design area is the main working area in ZenFlow. This area shows the visual representation and/or other representations of a business process. For example, Figure 2 shows a split view of a process definition (graph + tree). The visual representation consists of a flow chart view that shows a BPEL process as a set of BPEL visual elements representing data and control flow.

The visual composer generates a BPEL file and is equipped with a BPEL compiler that enables to import, visualize and edit BPEL process descriptions. When ZenFlow reads a BPEL process description created with any other tool, it might contain errors. The compiler checks the syntactic and semantic correctness of the BPEL process. During this process, an internal object representation of the BPEL process is built. This internal representation (the model in the Model-View-Controller jargon) is used to support the visualization and editing of that process. The error pane (located at the bottom) shows the errors and warnings in the process that is being developed.

4.2. Visualization and Navigation of Business Processes

The implementation of the visual composer is based on the Model-View-Controller (MVC) design pattern and inherently supports an arbitrary number of different views of a business process. In this section, we examine this multiview support as well as other visualization facilities. Currently, ZenFlow supports five different views: flow chart, form, text tree, error, free text, and execution. The flow chart view is a graphical view of the control flow of a business process (Fig. 1). The control flow is shown as a graph in which there are activities and other BPEL syntactic elements. They can be simple statements such as assignments or web service invocations, or structured, such as sequences or parallel execution activities. These syntactic elements are connected among them with arrows representing the control flow. The visual composer provides automatic layout support in order to visualize the flow chart as a planar graph according to a set of aesthetic rules. One of these aesthetic rules is that elements of a process are vertically centred in a recursive fashion. Another rule is that branches in flow statements (parallel execution construction) are aligned with respect to the minimum bounding boxes of each branch. This means that the user can attain automatically, without any effort, a high quality visualization of a business process. Additionally, manual layout is also allowed in order to provide the necessary degree of flexibility. The user can move joints between activities, then the activities are relocated according to the new joints. The positions of the joints manually set are recorded till an automatic layout is triggered by the user. Since syntactic elements might have a high number of attributes, these attributes are not presented in the flow chart view. Instead they are accessible through the form view. The form view is a dialog that provides access to all the attributes of a syntactic element. The form can be presented as a pop-up dialog or as a pane.

The text tree view provides a text view of a process (Fig. 2). This view is structured as a tree similar to XML editors, but it does not show the full XML text just a projection with the most relevant elements of a BPEL process to ease its navigation and readability. The root of the process is the process itself. The branches of the root are the activities in the process description. The contents of this view is synchronized with all other views including the flow chart view. Therefore, when the user switches views (or multiple views are shown simultaneously) the focus is in the same point. The programmer may add an operation in the flow chart and switch to the text tree view to see the code inserted. Any change done in any of the views is automatically updated in the rest of the views. All views are just different visualizations of the same model and the focus of visualization is set on the model, so it is unique for all views.

Syntactic and semantic errors are shown in the error view (Fig. 3). There are two kinds of errors: warnings and syntactic errors. Warnings are non-critical inconsistencies, and they are shown in yellow. On the other hand, syntactic errors are process definitions incompatible with the BPEL syntax, and are identified in red. Errors can be filtered according to different criteria to ease their processing. ZenFlow identifies two types of syntactic errors; those associated with the BPEL process itself or associated with WSDL files. The error view provides a projection of the process in which only the syntactic elements with associated errors are shown. Errors are associated to the corresponding syntactic elements. The error view is also synchronized with the other views. So, clicking on this list the focus of the navigation is changed in all other views resulting in the presentation of the syntactic element in which the error has been found.

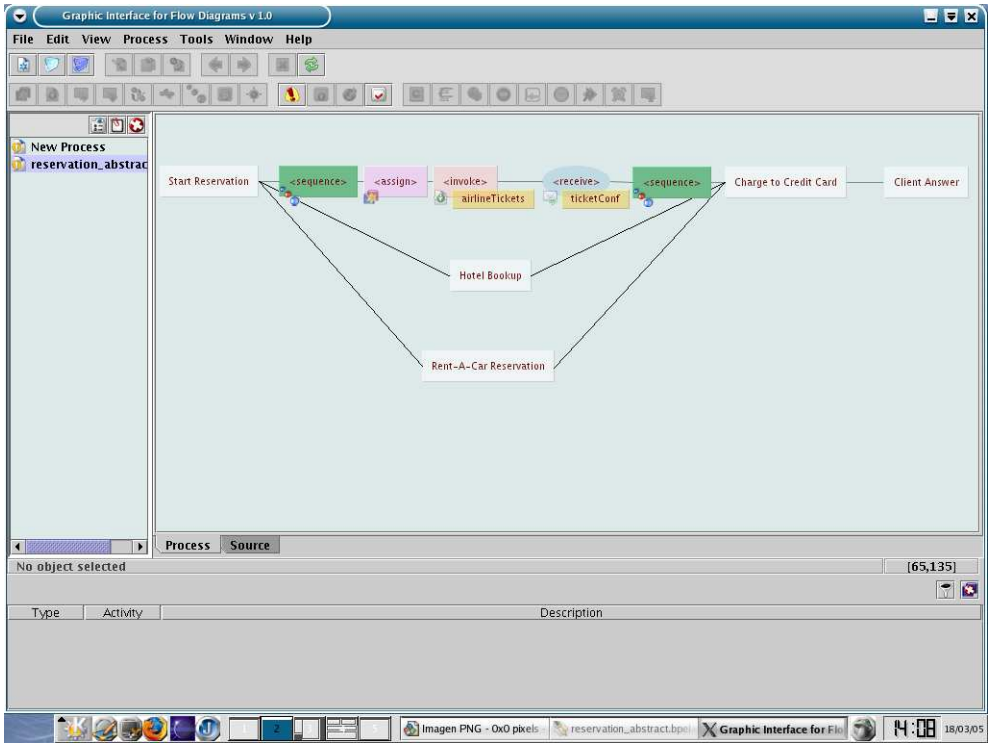

Figure 1. A business process

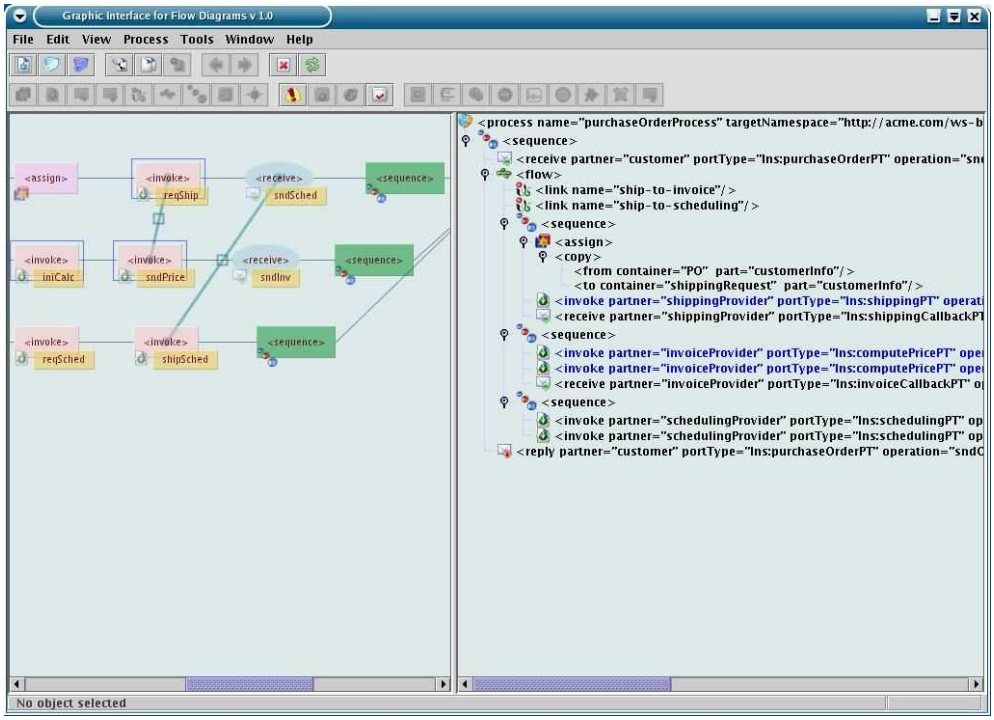

Figure 2. Tree and visual views of a process

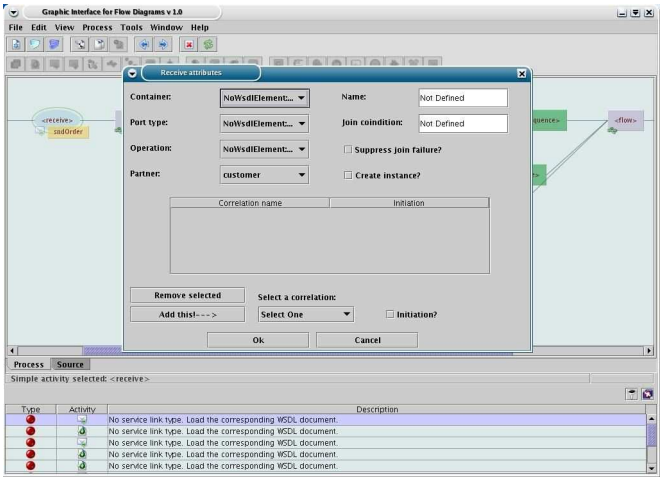

Figure 3. The Error View

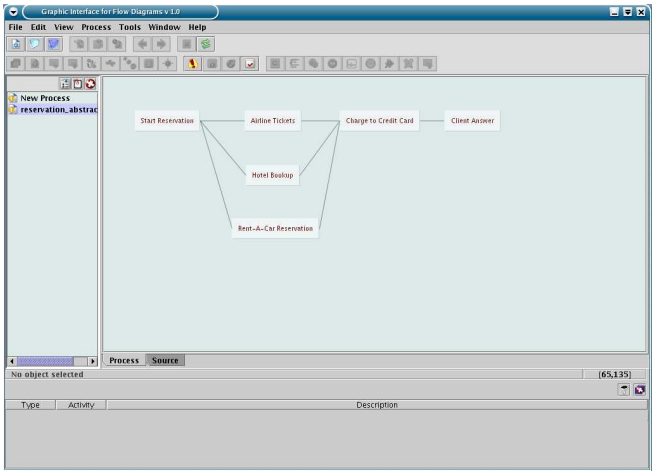

Figure 4. Abstract process definition

In some cases, it may be useful to be able to freely type and edit the BPEL text without any constraint. The visual composer supports this functionality through a text editor in which the XML code of the BPEL process is shown and can be freely modified. In this case, the consistency of the BPEL process is not guaranteed. In this case, the text must be saved and compiled after being edited in order to generate the associated views. If some error is found, it is shown in the error view.

Finally, the ZenFlow BPEL engine is connected to the visual composer to enable visual inspection and debugging of running processes. The visualization is done in the flow chart view. The path and activities already executed are colored and the value of their parameters can be inspected.

4.3. Facilities for Enhancing the Scalability of Views

All views can be navigated through the panel scroll bars. However, the scalability of scroll bars is known to be poor [6], since it is easy to lose track of the context of the current focus of the visualization. Some facilities have been included in ZenFlow in order to improve the scalability of the views, more particularly, the flow chart, text tree and execution views. The first one is a syntactic filtering tool that enables to select which activities should be shown in either the flow chart or text tree views and which ones should be hidden. For instance, the programmer may be interested in just seeing web service invocations of the process under development (e.g. to show only the web service choreography) or all the parallel statements (e.g. to observe the process concurrency). The filtering tool is accessed through a dialog that allows the selection through check boxes of the syntactic entities to be shown (e.g. web service invocations, invocations to the process, sequence bracketing, etc.). This filtering tool enables the projection of the visualized process removing details that are not of interest at a particular instant. These elements are shown as ellipsis in the graphical view. The hidden elements can be shown again either by using the same dialog or clicking in the flow chart in that element. Figure 5 shows a simplified view (only basic activities) of the process shown in Figure 2.

The error view also has a filtering tool that enables users to restrict the set of errors shown according to different criteria (severity, group of syntactic elements, etc.). This also helps the designer to focus only on the errors that she is tackling at the moment.

The abstraction tool is another facility targeted to enhance the scalability of these views. The abstraction tool enables to group and name a bunch of connected activities, as it happens with procedures and complex statements in imperative programming languages. Groups can also be nested. This grouping enables to define different levels of abstraction on the process definition. In this way, it becomes possible to fold (zoom in) and unfold (zoom out) to any level of depth each particular group by just clicking on it. For instance, the view in Figure 4 is an abstract definition of a process to book a holiday with three tasks: hotel, airline ticket and car booking. The BPEL code of the airline ticket task is shown just clicking in that task (Fig. 1). Fig. 6 shows the complete BPEL process after unfolding the other two tasks. This facility enables the navigation of a BPEL process at different levels of abstraction and to have a higher level of detail on the current focus of the navigation and a fading detail on the rest of the process. It is possible both to have the necessary detail on the current focus of the navigation and lower level of detail around the focus providing the context of the focus. This enables a kind of manual logical fish-eye views. This support is the base for future automatic

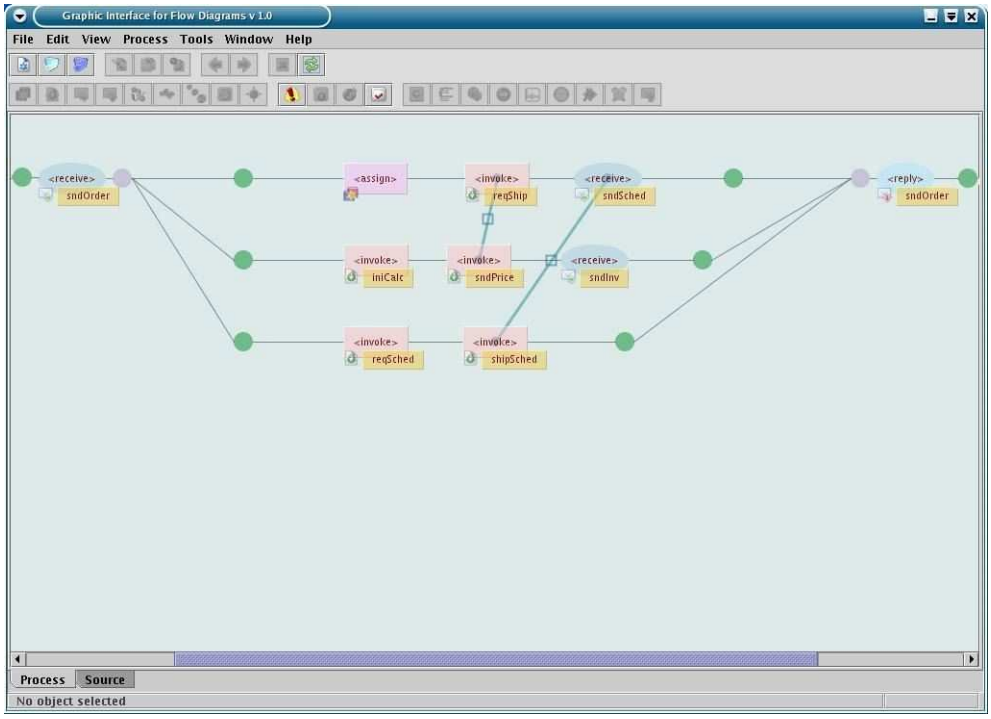

Figure 5. Visual representation of a process

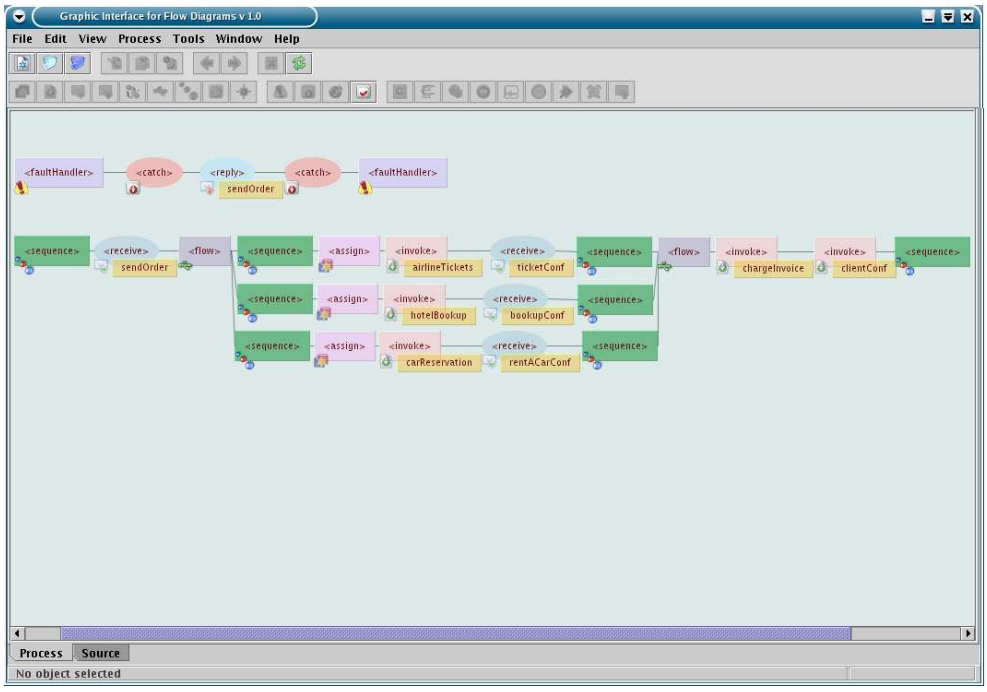

Figure 6. Holiday reservation BPEL process

fish-eye view facilities [6] that will fold and unfold automatically the different groups of activities depending on the distance to the current foci of interest.

4.4. Drag and Drop Visual Composition of Business Processes

The ZenFlow visual composer is based on the aforementioned visualization and navigation facilities. When navigating a process it can be updated and extended as needed. The navigation focus is therefore, also the editing focus. This focus plays the same role of cursors.

The composer has a palette in the menu with all the activities that can be used in a BPEL process. The user just needs to drag the appropriate activity from the palette and drop it on the process being edited in either the flow chart view or the text tree view of the process design area. In both cases, the programmer is freed from typing the whole activity, she just types the information that depends on the process (e.g. the name and parameters of a web service invocation but, not the invoke syntax).

The visual composer was designed to support two styles of business process composition: bottom-up and top-down. Bottom-up composition consists in building business processes from scratch activity by activity. Under this composition style structured statements (sequences or scopes) are added as empty constructions. And then, more syntactic elements are added to them. However, in the top-down composition the designer starts with a specification of the business process in which only the interactions (the choreography) with other web services are shown and then, the business process logic is added. In this case, it happens that existing activities should be surrounded by new complex activities. For instance, invocations to the process being defined may happen in parallel and therefore, they are subsumed within a flow activity. Usually editors do not support this kind of editing and force to create an empty complex activity (i.e. an empty sequence) and then cut and paste. This kind of editing results a very inconvenient and unproductive way of work in case several complex activities are nested (e.g. a sequence within a flow that is contained by a scope). For this reason, the ZenFlow composer supports the insertion of complex activities surrounding blocks of existing activities. That is, it is possible to mark a group of activities and then add a complex activity (i.e. a sequence, flow . . .) that will enclose them. The effect is that the newly added complex activity will enclose that group of activities preventing the painful and error prone cut and paste editing.

The composer enforces the consistency of the process being edited, as far as possible. In this regard, some BPEL activities can be used or not depending on the focus of editing. ZenFlow disables the BPEL activities from the palette that do not fulfil the BPEL syntax at the current focus of editing. For example, when the programmer starts a new process, selects the *new process* icon and the visual representation for a BPEL process is shown in the design area. In the BPEL toolbar, only the BPEL elements corresponding to *activities*, *fault handlers* and *compensations* are enabled. If the activity icon is selected, then all the activity icons are shown. Hence, syntactic errors in the composition of a BPEL processes are avoided because only the elements that match the BPEL syntax are enabled. However, a BPEL process should not be enforced to be entirely correct while editing, since this can be inconvenient for the programmer. Only global consistency of the process structure is enforced as well as syntactic well-formedness. In this way, a web service invocation can be inserted in the process without providing all its details (e.g. parameters). As the process is being created, the language constraints are controlled by the composer to check that the process conforms to the BPEL syntax. The error view pane is updated dynamically to show the current inconsistencies found in the process.

The selection of multiple BPEL activities is allowed. This multiple selection also enables to cut or copy a group of BPEL activities in either the flow chart or the text tree. Interestingly, the composer checks that the cut is consistent. This consistency lies in the fact that complex activities have to be selected completely. Similarly when pasting, the composer checks that the cut is syntactically and semantically correct. That is, the paste functionality is disabled at those editing foci at which the paste would be syntactically incorrect. In addition, a clipboard window gives access to previously cut or copied BPEL activities.

ZenFlow integrates a UDDI browser to access UDDI registries. The visual UDDI browser may connect to any standard UDDI repository through the standard web service interface. The browser enables queries and the navigation of the results of these queries. Once the programmer has found the web service she was looking for, she can drag and drop it from the visual UDDI browser in the business process. As a result, the composer will ask which kind of interaction the user wants to use for that web service (invoke or receive). Additionally, the BPEL process definition is enriched accordingly to support the interaction with that web service (e.g. its web service interface).

5. Related Work

Although web service technology has appeared very recently, the area of visual composition of web services has attracted a lot of attention and several tools have been developed. The industrial products BPWS4J [7] and BPEL Process Manager suite [13] are both based on BPEL. BPWS4J [7] provides a simple visual environment where processes are shown as a tree structure. There is no graphical workflow-like view of a process. The environment synchro-

nizes the tree and the BPEL code views. BPWS4J lacks of other external helper tools for process composition like a UDDI browser and a clean error management. The error list shows all the errors of the currently open BPEL files and errors have to be deleted manually from the list. No redirection to the corresponding code is done. The BPEL Process Manager suite [13] provides a visual composition tool called BPEL Designer. The process design environment offers two views of a BPEL process. The first one is similar to the graphical one of ZenFlow. The process designer can drag BPEL visual elements from a toolbar to the working area. However, none of the advanced visual ZenFlow facilities are available, like the ones for the scalability of views. For instance, simplified views of processes are not allowed and the designer cannot select and manipulate multiple activities. The second view shows the whole process as a unique visual element and just allows the definition of the *partner links*. Error management is complex. Errors are shown in the properties of each activity of the process. The process designer must open the properties of an activity in order to find an error.

The BioOpera Flow Language (BFL) [14] is a visual composition language for web services. The visualization is based on a dataflow language. A single visual BPEL element of ZenFlow can represent a few dataflow elements. The Self-Serv environment [2] supports the composition of web services. The business logic of a composite service is represented as a state chart. The tool lacks any advanced visual composition facility. Triana [10] is a graphical web service composition and execution toolkit. A composite service is created by dragging the services and connecting them. Triana supports loops and conditional constructs. In contrast with ZenFlow, the Triana GUI is not based on BPEL. However, a workflow created with this GUI can be saved as a BPEL process. Websight [4] is another tool for visualizing the execution of web services. The Business Process Modelling Notation (BPMN) has been proposed very recently [20]. BPMN is a specification to standardize a graphical notation for process developers. It defines a business process diagram, which is based on flowcharts for creating graphical models of business process operations. The design of ZenFlow eases the incorporation of this graphical notation as another view.

6. Conclusions

ZenFlow frees the process designer from dealing with XML code. ZenFlow provides visual abstract views of it that ease the task of design. The composer offers multiple synchronized views of a process so, the designer can choose the most convenient one to navigate and edit a given process. The scalability of the visualization is supported by several means among which it should be highlighted the nested hierarchical visualization of activities and its selective level of detail. This mechanism provides the power of logical fisheye views handled manually which enables a high level of detail on the current focus of attention and lower levels of detail of the surrounding context. The visual composer of ZenFlow supports two styles of composition: top-down and bottom-up. Powerful editing primitives are provided to support both styles. The editor guarantees the syntactic correctness of processes by enabling and disabling the activities that can be inserted in the process and by enforcing consistent cut and paste.

References

[1] Apache. *AXIS*. http://ws.apache.org/axis.

[2] B. Benatallah, Q. Z. Sheng, and M. Dumas. The self-serv environment for web services composition. *IEEE Internet Computing*, 7(1):40–48, 2003.

[3] BPMI. *Business Process Modelling Language for Web Services (BPML) 1.0*. 2001. http://www.bpmi.org/BPML.htm.

[4] W. de Pauw. Visualizing the execution of web services. In *Workshop on Testing, Analysis and Verification of Web Services. Invited presentation*, 2004.

[5] Eclipse. *Eclipse Platform*. http://www.eclipse.org.

[6] G. W. Furnas. Generalized fisheye views. In *Proc. Of ACM CHI Conference*, pages 16–23, 1986.

[7] IBM. *BPWS4J*. http://www.alphaworks.ibm.com/tech/bpws4j.

[8] IBM, Microsoft, and BEA. *Business Process Execution Language for Web Services*. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/ .

[9] F. Leymann. *Web Services Flow Language (WSFL) 1.0*. 2001. http://www-3.ibm.com/software/solutions/ webservices/pdf/WSFL.pdf.

[10] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A graphical web service composition and execution toolkit. In *Int. Conf. on Web Services*, pages 514–521, 2004.

[11] Microsoft. *.NET Framework*. http://www.microsoft.com/net/.

[12] Microsoft. *XLANG: Web Services for Business Process Desing*. 2001.

[13] Oracle. *Oracle BPEL Process Manager*. 2004. http://www.oracle.com/technology/products/ias/bpel/index.html.

[14] C. Pautasso and G. Alonso. Visual composition of web services. In *IEEE Symp. on Human Centric Computing Languages and Environments*, 2003.

[15] Sun. *Java Web Services Developper Pack (Java WSDP)*. http://java.sun.com/webservices/jwsdp/index.jsp.

[16] UDDI. *Universal Description, Discovery and Integration of Web Services*. http://uddi.org/pubs/uddi_v3.htm.

[17] W3C. *Web Services Choreography Description Language*. http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/.

[18] W3C. *Web Services Definition Language (WSDL)*. 2001. http://www.w3.org/TR/wsdl.

[19] W3C. *Simple Object Access Protocol (SOAP) 1.2*. 2002. http://www.w3.org/TR/soap12.

[20] S. White. *Business Process Modelling Notation (BPMN)*. 2004. http://www.bpmi.org/BPMN.htm.

[21] P. Wohed, W. Aalst, M. Dumas, and A. Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In *22nd Int. Conf. on Conceptual Modeling (ER 2003)*, 2003.