e-Informatica Software Engineering Journal, Volume 1, Issue 1, 2007

A User-Centered Approach to Modeling BPEL Business Processes Using SUCD Use Cases

Mohamed El-Attar*, James Miller*

*STEAM Laboratory – Electrical and Computer Engineering Deparment, University of Alberta

melattar@ece.ualberta.ca, jm@ece.ualberta.ca

Abstract

BPEL is being widely used to specify business processes through the orchestration, composition and coordination of web services. It is now common practice to begin the process of modeling the "workflows" within a set of BPEL business processes using UML Activity Diagrams since they can be automatically mapped down onto BPEL code. However activity diagrams were not intended to explicitly model user goals and interactions with external systems offering web services. However, since the chief purpose of BPEL business processes is to first and foremost provide services to their users, using activity diagram modeling alone will not allow an E-commerce analyst to explicitly capture and model the users' goals. In this paper we propose an approach to solve this issue; initially model BPEL business processes using Use Cases to capture users' perspective, and to systematically develop activity diagrams from Use Case models. A Travel Agency system case study is presented illustrates the feasibility of the proposed approach.

1 Introduction

Web services offer their users an efficient means to solicit and research publicly available services. A user maybe interested in acquiring the best deal on a particular service or a product from a number of competitors that offer that service or product. For example, a customer interested in purchasing a particular book will be interested in obtaining the best price from a number of book vendors. Alternatively, users can be interested in the collaboration of a number of web services to attain a higher level goal. For example, a user can be interested in a set of web services provided by couriers that can interact with each other to provide tracking and history details of a current shipment. BPEL processes can be created to specify the invocation order of web services to achieve the desired goal. Using BPEL, a great deal of interaction information between web services and the BPEL process user can be specified, commonly known as defining a business process. Every BPEL business process has a purpose to achieve; this purpose is usually to provide a service to the process's user. It is not necessary that the user must be human; the user of a business process can be another system. In any case, it is the responsibility of an E-commerce analyst to define BPEL business processes that provide the services that are in demand.

BPEL provides support to specify complex business processes that contain sequences, loops, conditions, exception handling, variable declarations and data editing. In essence,

a BPEL business process defines a workflow. Therefore, it is common practice to model these workflows using the Unified Modeling Language (UML) activity diagrams, since it can be mapped directly onto BPEL code. In this paper, an activity diagram that represents a BPEL business process will be referred to as BPEL activity diagram. A BPEL activity diagram needs to posses a great deal of quality. A high quality BPEL activity diagram can be defined as one that accurately represents the workflow required to satisfy a business requirement. In the analysis phase, an E-commerce analyst will develop BPEL activity diagrams directly from a set of business requirements. This might be problematic for the following reasons:

- Activity diagrams are not geared towards capturing the user-centric perspective of business workflows. It is important to model such a perspective since the user is the principle beneficiary of the BPEL business process. Therefore, it is crucial to understand the means by which the user(s) will interact with the host system(s) during the execution of a BPEL business

- Activity diagrams do not capture the intricacies of interactions occurring between web services (external systems) and the host system that runs the BPEL business process process.

- It is common to define a set of related services using a set of BPEL business processes. This concept is illustrated through the Travel Agency System case study presented in Section 4. Activity diagrams are not designed to provide a mechanism to discover common activities, interactions and sub-services provided by a set of related BPEL services. Not being able to discover and factor out such commonalities might result in unnecessary effort required for implementing redundant functionalities at the end system; and since BPEL processes potentially have highly extended execution periods, these redundancies can be significant.

Therefore, in addition to knowing what the goals are, it is essential to know how the user will utilize the BPEL business process. It can be argued that understanding the user's participation in the business process will actually yield to creating higher quality activity diagrams, in the sense that the activity diagrams created will more accurately represent the user's involvement in the business process and the involvement of the available web services. To combat the issues presented above, we propose using Use Cases to model the user's perspective. Whereby, consequent activity diagrams can be developed based upon the Use Case models. Use Case modeling has become the de-facto technique for modeling user-centric systems. A Use Case model will detail the intricacies of interactions that occur between the BPEL business process users and the web services provided by external systems. Use Case modeling prompts E-commerce analysts to consider common behavior and sub-services allowing the development of simpler and more modular systems.

There are two main requirements to develop high quality BPEL activity diagrams. Firstly, it is crucial to develop a high quality Use Case model. It is intuitive that if one artifact is developed based on another artifact, the quality of the source artifact will extensively influence the quality of the resultant artifact. Utilizing the Structured Use Case

Descriptions (SUCD) form to develop Use Case descriptions can help achieve this goal. Use Cases described in the SUCD form will be referred to as SUCD Use Cases. Secondly, an equally important requirement is to provide a systematic transition between the Use Case model and the corresponding set of BPEL activity diagrams. This can be achieved by using the AGADUC (Automated Generation of Activity Diagrams from Use Cases) process. AGADUC is a systematic approach to transform Use Case models into UCADs (Use Case Activity Diagrams). UCAD is a notation introduced in [9] that represents the workflows embedded in SUCD Use Cases. The UCADs produced can be considered as BPEL activity diagrams upon which the implementation of the BPEL processes will be based. A principle advantage of using AGADUC is that it is supported by the tool AREUCD (Automated Reverse Engineering of UC Diagrams). Automating the transformation process reduces the time and effort required to develop BPEL activity diagrams. Therefore, E-commerce analysts can further focus on the development of high quality Use Case models. Furthermore, automating the transformation process will ensure consistency between Use Case models and activity diagrams by eliminating errors injected by analysts, and eliminate "inspection-like" effort required to ensure such consistency. An overview of the AGADUC process is shown in Figure 1.

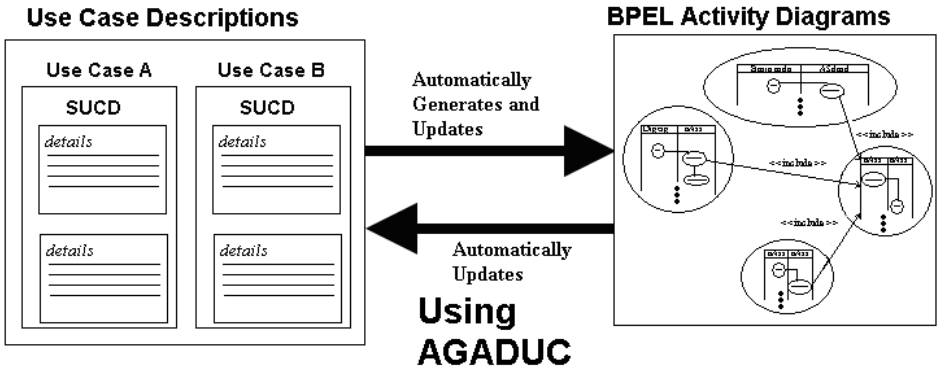

Figure 1: An overview of the AGADUC process

Activity diagrams are traditionally used to model BPEL processes since activity diagrams are designed to support the specification of workflows. Workflows that may contain loops, branches and conditions, and concurrent flows, and can be easily visualized using activity diagrams. E-commerce analysts have avoided adopting Use Cases to model their BPEL process, since Use Case models are not intended for expressing workflows, but rather actor-system interactions. It is rather cumbersome to express workflow features using Use Cases in a concise and understandable manner. Traditional Use Cases are described using unstructured natural language that makes it impossible to automatically extract and represent their embedded workflows using activity diagrams. SUCD helps overcome the limitations endured when using Use Cases to model workflows. As will be discussed in great detail in Section 3, SUCD contains enough structure to support the concise modeling of workflows. There will be no additional effort in creating the corresponding activity diagrams, since AREUCD automatically performs this operation. Therefore, SUCD allows

E-commerce analysts to utilize the user-centered approach to modeling their BPEL processes without sacrificing the benefits of visualizing their BPEL processes using activity diagrams!

The remainder of this paper is structured as follows: Section 2 discusses previous work related to modeling BPEL processes. Section 3 introduces the SUCD structure and the AGADUC process and how it can be utilized to describe BPEL business processes and automatically generate BPEL activity diagrams. In Section 4, a Travel Agency System case study is presented to illustrate feasibility of our proposed approach, and to demonstrate the application of the AREUCD tool. Section 5 concludes and discusses future work.

2 Related Work

Schader et el. [3] and Aagedel et al. [13] have shown through a number of case studies the limitations of using activity diagrams for business process modeling. Dumas et al. [4] extended this work by examining the expressiveness and adequacy of UML Activity Diagrams with respect to specify workflows in particular. The result of their study shows that activity diagrams possess features that allow for capturing situations arising in practice that usually cannot be captured by most Workflow Management Systems. However, their study also revealed that activity diagrams suffer from limitations inherited from statecharts, which stem from the fact that activity diagrams are a special type of state machines.

Korherr et al. [5] presented an extension to the UML Activity Diagram. The extension allows for modeling process goals and performance measures in a visual manner by the definition of a UML profile. The profile allows the extended activity diagrams to be mapped onto BPEL. Mantell also created a UML profile that supports the systematic transformation of activity diagrams to BPEL code [8]. Mantell's work featured tool support that automates the transformation process and generates skeletons of BPEL code.

In 1997, it was argued by Keung et al. that there was little contribution made to make goals explicit during the modeling of business processes [7]. Today, a large number of business modeling languages have been introduced such as the Business Process Modeling Language [6], the Event-driven Process Chain [11] and UML Activity Diagrams [10], however none of them provide means to model business process goals [1]. Towards achieving this objective, this paper proposes a use-centered approach to modeling BPEL business process using Use Cases.

3 The Structural Elements of SUCD

SUCD is in large based on the structure presented by [2]. Use Cases described using the SUCD structure contain five structured main sections, these are: (a) Use Case Name, (b) Basic Flow, (c) Alternative Flows, (d) Subflows and (e) Extension Points. Meanwhile, other sections in a Use Case description that do not require structure are described using natural language. There have many templates presented in the literature for describing Use Cases. This section will provide details about the SUCD structural elements that

are relative to supporting the specification of workflows. A complete description of the SUCD structure and a mini-tutorial can be located at [12]. The systematic mapping of the SUCD structural elements to activity diagram notation and the underlying transformation algorithm is also presented in this section. The formalized mapping rules also serves as a mechanism to ensure inter-diagrammatic consistency between the Use Case model and the activity diagrams.

The following list outlines the structural elements of SUCD that support the specification of workflows which will be described in detail in the subsequent sections:

- Headers and Actions
  - Transforming Headers and Actions
  - Swimlanes
  - Nested Activity Diagrams
- Concurrency and Loop Support using the 'RESUME' and 'AFTER' Statements
- Condition Evaluation and Branching Support using 'AT' and 'IF' Statements

3.1 Headers and Actions

The basic building block comprising all structured components is headers. A header contains a number of actions that carryout certain behavior. The name of the header indicates the behavior that is carried out by its actions. A header is comprised by a pair of matching tags. An 'opening' tag is comprised of curly brackets that contain the header's name <header> prefixed with the keyword 'BEGIN'. Its corresponding 'closing' tag must contain the same header name and is prefixed with the keyword 'END'. A header's enclosed actions are normally listed in bullet form. For example, in a library system, a header may represent the actions required to enter information regarding how a new library member:

{BEGIN enter member information}
- Librarian → enters member's name
- Librarian → enters member's address
- Librarian → enters member's phone number
{END enter member information}

This header Enter Member Information contains three actions. In this paper, performing a header indicates that all of its enclosed actions are performed. Each header inside a Use Case must have a unique name. It can be easily deduced that the purpose behind performing the three actions shown above is to enter a member's information into the system. Moreover, all three actions must be performed to carryout the underlying purpose of the header. A header groups together a set of actions that must all be performed in order to carryout complete and meaningful behavior.

A header may contain other lower-level headers that comprise parts of the behavior required to carryout the main behavior represented by the higher-level header. Therefore,

a Use Case description will contain a virtual tree of headers, whereby actions become the roots (see Fig. 2). A high-level header may have actions of its own. Performing a higher-level header forces the all of its lower-level headers in addition to its own actions to be performed. When a lower-level header is completely performed, its higher-level header resumes performance. Actions listed under a header represent the actual behavioral details

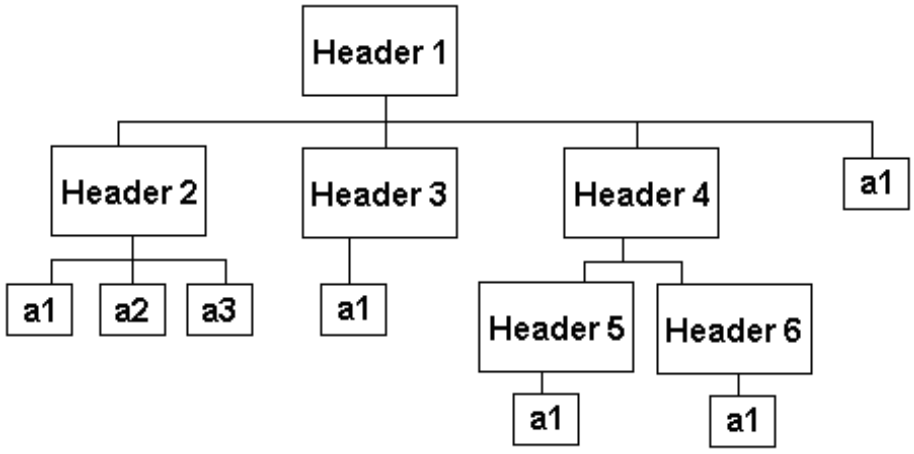

Figure 2: Headers in a Use Case descriptions form a virtual tree structure

of a Use Case. Actions must be listed in bullet form and described using natural language. Listing actions in bullet form will allow analysts and designers to trace back design artifacts and decisions to individual actions in a Use Case description. Only one actor may perform an action, unless the action is performed by the system itself. The name of the actor that performs a given action is prefixed to that action. For the Enter Member Information header shown above, the Librarian actor performs all three actions shown. Actions that are performed by the system itself are prefixed with the keyword 'SYSTEM'.

3.1.1 Transforming Headers

Since headers are the basic building blocks for the three types of flows, it is only appropriate to start with the transformation of headers to activity diagram elements. Each action enclosed in a header is directly represented by an activity in an activity diagram. The Enter Member Information header presented in section 3.1 is translated into the following activities as shown in Fig. 3.

3.1.2 Swimlanes

Swimlanes are used to associate each activity with an actor. This assigns the responsibilities of each actor. In SUCD Use Cases, each action is designated an actor, unless it is performed by the system (where the keyword 'SYSTEM' is used). Hence, assigning each action to the appropriate swimlane is straightforward (see Fig. 3).

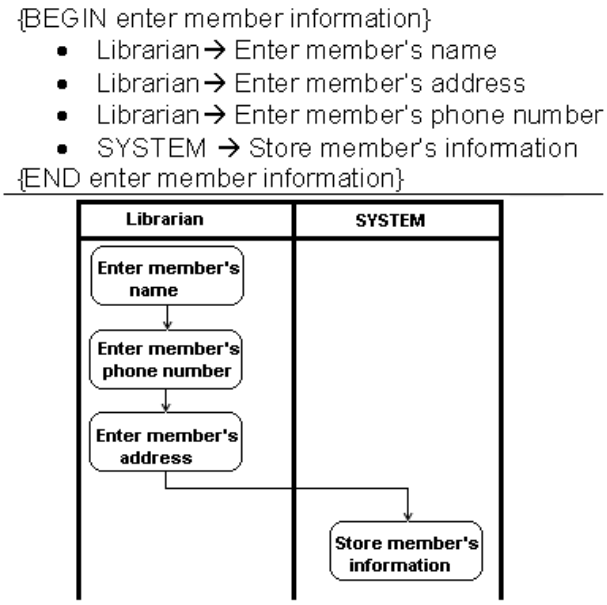

Figure 3: Headers and swimlanes

3.1.3 Nested Activity Diagrams

Each activity diagram maybe nested. Nesting in activity diagrams is used to model a hierarchy between the activities and to manage the complexity of the activity diagram. Nesting activity diagrams do not change the semantics behind the activity diagrams. Therefore, whether the activity diagrams were nested or not, should not change the underlying concepts and workflows presented by the diagrams. Hence, there is no 'hard and fast' rule as to what sections in an activity diagram should be nested. Nesting sections of an activity diagram is a judgment call made by the E-commerce analysts and designers. The proposed Use Case description structure only provides guidelines to what can be nested.

SUCD uses headers only to show what can be nested, as supposed to what should be nested. A set of actions can be abstracted to show their corresponding header as an activity. Lower-level headers can be abstracted to show corresponding higher-level header as an activity. Assuming a header named Enroll New Member that is composed of three lower-level headers; Enter Member Information, Enter Record into Library Database and Produce Library Card For New Member. As shown in figure 4, the lower-level headers can be abstracted to show the higher-level header as an activity. The Enter Member Information header presented in Section 3.1 is composed of actions. The actions can be abstracted to show its header as an activity (see Fig. 4).

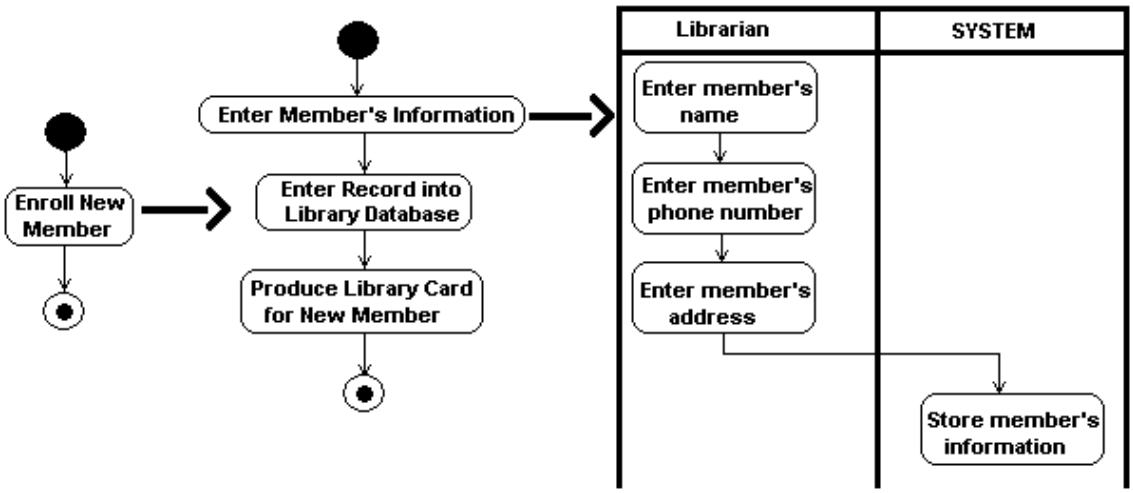

Figure 4: Different nesting levels for presentation purposes

3.2 Concurrency and Loop Support

Activity diagrams have features such as forking and joining to support activity synchronization. As already discussed, forking and joining execution flows in activity diagrams are modeled using the 'RESUME' and 'AFTER' statements in the Use Case descriptions.

A header can explicitly state the header(s) to be performed next. This can be achieved using the 'RESUME' statement. The 'RESUME' statement consists of the keyword 'RESUME' followed by a list of header(s) that will follow. This is used to model the concept of flow forking. Finally, a header may explicitly state the headers that must be completed before it can commence. This is achieved using the 'AFTER' statement. Similarly, the 'AFTER' statement consists of the keyword 'AFTER' followed by a list of headers that need to be completed first (see Fig. 5):

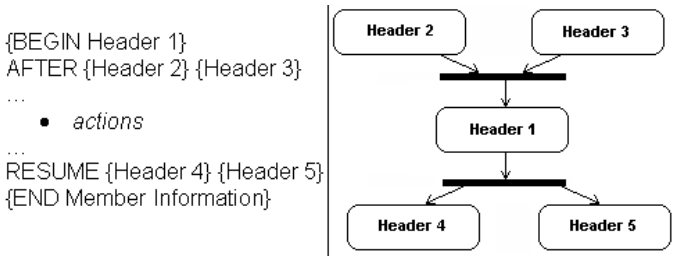

Figure 5: Forking and joining using SUCD

3.3 Supporting Condition Evaluation and Branching

Workflows may contain decision points where a condition is evaluated. Decision points are indicated using the 'AT' statement. The 'AT' statement consists of the keyword 'AT' followed by a header where the stated conditions are evaluated. By default, the conditions are evaluated for each action within the specified header. Alternatively, the conditions can be evaluated only at certain actions within the specified header. This is achieved by specifying these actions after the 'AT' statement.

Conditions evaluated at each 'AT' statement are indicated using 'IF' statements. An 'IF' statement consists of the keyword 'IF', followed by a condition described in natural language.

An 'AT' statement will cause the creation of a decision diamond following the activity(s) that represent the header(s) listed in the 'AT' statement. An 'IF' condition will be displayed as an activity diagram condition at the corresponding decision diamond (see Fig. 6). A 'RESUME' statement is used afterwards to specify the action where the flow will be heading towards.

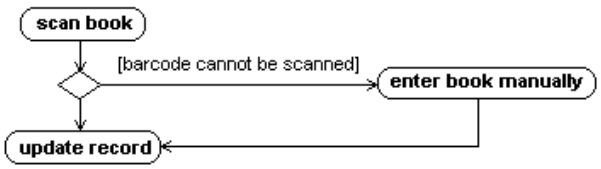

Figure 6: Modeling conditions and braches

3.4 Formalizing the SUCD Structure

It is essential for the grammar and constructs of the SUCD structure to be formalized. Formalizing the SUCD structure will provide a strict guideline to Use Case authors in composing Use Case descriptions, so that there is no disagreement or ambiguity as to what is allowed and what is not. The grammar of the SUCD structure is defined below in E-BNF (see Table 1). Due to space restrictions, only the grammar of SUCD's higher structural constructs are shown below, while the grammar of minor structural constructs are excluded. However, the entire E-BNF is located at [12].

3.5 The AGADUC Process Mapping Rules

In order for the AGADUC process to be tool supported, the mapping rules for transforming SUCD Use Cases in activity diagrams must be formalized. A complete specification of the implementation of the mapping rules will require many pages in length. A summarized pseudo code version of the mapping rules are presented below (see Fig. 7).

The mapping process is carried out by four main algorithms [12]. The first algorithm is responsible for scanning through the headers and actions of a SUCD Use Case and creating a hierarchy of activities that represent this hierarchy. Algorithm 1 will also assign activities representing actions to their corresponding swimlanes. Finally, the first algorithm will

S ::= UseCaseDescrption+ Actor+
Actor ::= Abstract? ActorName Implements? Specializes?
UseCaseDescrption ::= NameSection BasicFlowSection? AlternativeFlowSection? SubflowsSection? ExtensionPointsSection?
NameSection ::= 'Use Case Name:' Abstract? UseCaseName Implements? Specializes?
Abstract ::= 'ABSTRACT'
Implements ::= 'IMPLEMENTS' UseCaseName
Specializes ::= 'SPECIALIZES' UseCaseName
BasicFlowSection ::= 'Basic Flow:' 'BEGIN Use Case' Header* 'END Use Case'
Header ::= 'BEGIN' HeaderName '' AfterStatement? Contents* ResumeStatement? 'END' HeaderName ''
AlternativeFlowsSections ::= 'Alternative Flows:' AF*
AF ::= AtStatement IfStatement AFHeader

Table 1: E-BNF grammar for the SUCD structure

set control flow link between the actions of a header, and save any information regarding RESUME and AFTER statements detected. The second algorithm is responsible for building and maintaining a list of swimlanes detected from the actions. The third algorithm is responsible for creating synchronization bars and creating control flow links that connect activities with the synchronization bars. The execution of this algorithm depends on the information saved earlier regarding the RESUME and AFTER statements. The third algorithm requires that Algorithm 1 must be executed first. Finally, the fourth algorithm is responsible for creating the decision diamonds and conditions according to the AT and IF statements. The fourth algorithm requires that Algorithm 1 and Algorithm 3 are executed first. Due to space restricitions, only Algorithm 1 is shown below; the remaining Algorithms can be found at [12].

4 Business Traveler Case Study

The following case study is used to demonstrate how SUCD Use Cases can be used to by the AGADUC process to generate UCADs. The case study is about a simplified Business Traveler System that allows employees of a certain company to travel using the best offers for plane ticket(s) and travel insurance. The system provides three BPEL business processes. The first business process allows an employee to retrieve the best plane ticket offer

Algorithm 1: Transforming *headers* and *actions* into *activities*.
Input: The description of a SUCD Use Case.
Output: A hierarchy of *activities* based on the given hierarchy of the given *headers* and their *actions*.

```
Top_ Activity = NULL
Current_Parent_Activity = NULL
LET Statement = getFirstStatement(SUCD Use Case)

WHILE (EOF is not reached)
{
      IF Statement = Opening header tag
            Detect header name
            Create an activity and set its name as:
            Activity.name = header.name
            IF (Top_Header found)
                  Set activity as child of Current_Parent_Activity
                  Set activity as Current_Parent_Activity

            ELSE Set activity as Top_ Activity
                  Set activity as Current_Parent_Activity

      ELSE IF Statement = Closing header tag
            Set Current_Parent_Activity as
getLastActivity().getParent()

      ELSE IF Statement = action
            Detect action name
            Create an activity and set its name as:
            Activity.name = action.name

                  Detect action's actor
                  Create currentSwimlane = action.actor
                  Check for currentSwimlane (perform Algorithm 2)
                  currentSwimlane = swimlane reference returned from
Algorithm 2
                  Set activity.swimlane = currentSwimlane
                  currentSwimlane.addAcitivty (activity)
                  Set activity as child of Current_Parent_Activity
                  IF Not First or Last action in
Current_Parent_Activity
                        Create flow link from previous action in parent
to self
                  ELSE IF Last action in Current_Parent_Activity
                        Create flow link from previous action in parent
to self
          Look for any RESUME statements following it. If a RESUME
          statement exists, the save the headers specified as a property
          of that action.
                  ELSE IF First action in Current_Parent_Activity
          Look for any AFTER statements before it. If an AFTER statement
          exists, save the headers specified as a property of that
          action.

      ELSE //Statement is just an unstructured statement

      Statement = getNextStatement();
```

Figure 7: Algorithm 1. Detecting activities and creating main flow links

from two web services provided by American Airlines and Delta Airlines. The second business process allows an employee to receive the best travel insurance offer for an upcoming trip from two web services provided by Northern Insurance and Pacific Insurance. When searching for the best plane ticket offer or the best travel insurance offer, the employee's travel status must be checked. The travel status of an employee allows the company to determine which class that employee can use to travel, such as: business, first or economy class. The travel status also allows the company to determine the corresponding travel insurance package that the employee is entitled to receive. The travel status checking process is performed by the third BPEL process called "Check Employee Travel Status". The Use Case diagram of the system is presented below in Figure 8.

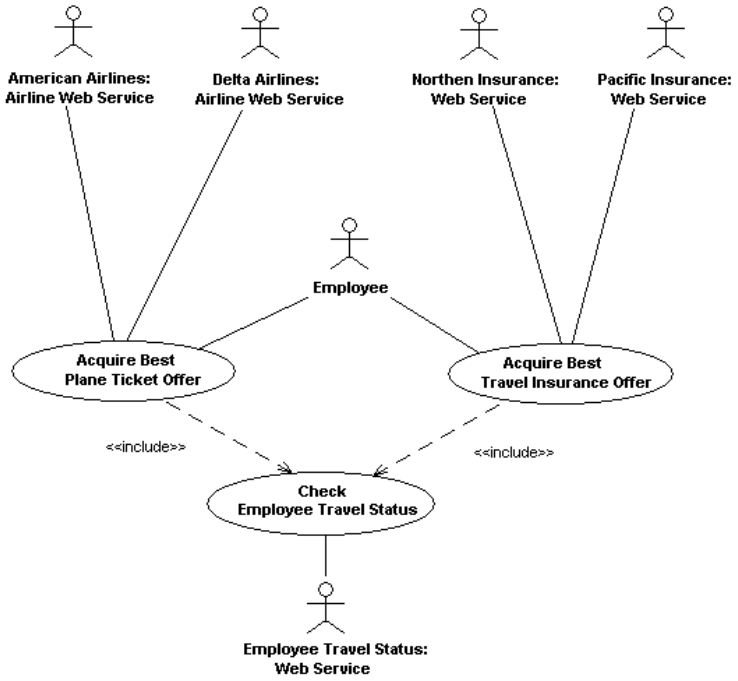

Figure 8: The Travel Agency System Use Case Diagram

As discussed in Section 3, BPEL business processes represent services that are offered by a system to its user(s). Therefore, the BPEL business processes are represented as Use Cases. The web services and the user of the BPEL processes interact with the "Use Cases" to attain their services. Therefore the web services and users are modeled as actors.

This case study is an expansion of the BPEL business process example presented in [14]. In [14], only one BPEL business process was discussed which combined the "Acquire Best Plane Ticket Offer" and "Check Employee Travel Status" BPEL processes shown below. The reason the "Check Employee Travel Status" BPEL process was created in our case study is to allow the "Acquire Best Travel Insurance Offer" to use its offered service. Therefore, using a Use Case driven approach, it was possible to identify and factor out

common behavior, which will help avoid the implementation of any redundant functionality and save development costs.

The BPEL activity diagram for the "Acquire Best Travel Insurance Offer" BPEL process is presented in Fig. 9. The BPEL business process was then implemented. Therefore, in this case study we will shed the light on the "Acquire Best Travel Insurance Offer" BPEL process since it was discussed in [14], to show that using a Use Case driven approach can produce the same BPEL activity diagram. Moreover, this case study will show how using Use Cases can overcome the limitations suffered by using the traditional development approach presented earlier in Section 1. Once again due to space resatrictions, the Use Case descriptions for the "Acquire Best Plane Ticket Offer" BPEL process is presented below using the SUCD structure, while the Use Case description of the "Check Employee Travel Status" BPEL process can be located at [12].

Use Case Name:
Acquire Best Plane Ticket Offer

Brief Description:
This Use Case describes a simple business process that selects the best airline flight ticket offer. The business process is carried out as a web service. Currently, there are two competing Airline companies that have subscribed to this web service, namely (a) American Airlines, and (b) Delta Airlines.

Preconditions:
The Employee must have approval to travel.

Basic Flow:
{BEGIN Use Case}

{BEGIN Receive the initial request}
- Employee → requests to search for the best plane ticket offer
- Employee → specifies his/her name
- Employee → specifies the destination
- Employee → specifies the departure date
- Employee → specifies the return date
{END Receive the initial request}

{BEGIN Prepare the input for the Employee web service}
- SYSTEM → retrieves the information inputted by the Employee and prepares for the Employee Travel Status Web Service
{END Prepare the input for the Employee web service}

{BEGIN Retrieve the employee travel status}
- SYSTEM → sends the Employee Travel Status Web Service the Employee information to check for the Employee's travel status

- INCLUDE Check Employee Travel Status
{END Retrieve the employee travel status}

{BEGIN Prepare the input for both Airline web services}
- SYSTEM → uses information provided by the Employee and the Employee Travel Status Web Service to prepare inquiry requests for both Airlines
RESUME {Acquire plane ticket offer from American Airlines} {Acquire plane ticket offer from Delta Airlines}
{END Prepare the input for both Airline web services}

{BEGIN Acquire plane ticket offer from American Airlines}
- American Airlines: Airlines Web Service → retrieves information about the requested plane ticket(s)
- American Airlines: Airlines Web Service → checks for tickets availability
- American Airlines: Airlines Web Service → if the requested ticket(s) are available the web service sends back an offer for the ticket(s). Otherwise, if the tickets were unavailable, the web service sends back a response indicating that
{END Acquire plane ticket offer from American Airlines}

{BEGIN Wait for a callback from American Airlines}
- SYSTEM → waits for a response from the American Airlines Web Service {END Wait for a callback from American Airlines}

{BEGIN Acquire plane ticket offer from Delta Airlines}
- Delta Airlines: Airlines Web Service → retrieves information about the requested plane ticket(s)
- Delta Airlines: Airlines Web Service → checks for tickets availability
- Delta Airlines: Airlines Web Service → if the requested ticket(s) are available the web service sends back an offer for the ticket(s). Otherwise, if the tickets were unavailable, the web service sends back a response indicating that
{END Acquire plane ticket offer from Delta Airlines}

{BEGIN Wait for a callback from Delta Airlines}
- SYSTEM → waits for a response from the Delta Airlines Web Service
{END Wait for a callback from Delta Airlines}

{BEGIN Select the best offer}
AFTER {Wait for a callback from American Airlines} {Wait for a callback from Delta Airlines}
- SYSTEM → after receiving a response from both Airlines web services, the SYSTEM selects the best offer
{END Select the best offer}

{BEGIN Return the offer}
- SYSTEM → returns to the Employee a response indicating the best offer for the requested plane tickets
{END Return the offer}

{END Use Case}

Postconditions:
A response must be provided to the Employee with the best offer for the requested plane tickets. If no tickets were available, a response must be provided to the Employee indicating that.

Special Requirements:
An internet connection must be available for the BPEL process to operate.

Upon inputting the SUCD Use Cases (all three of them) into the tool AREUCD, the AGADUC process will be performed and the UCADs for the SUCD Use Cases are generated. Due to space limitations, only the UCADs of the "Acquire Best Plane Ticket Offer" and "Check Employee Travel Status" BPEL processes are shown below (see Fig. 9 and Fig. 10).

In contrast with using the activity diagram for the "Acquire Best Plane Ticket Offer" business process, it can be deduced that describing the business process using a SUCD Use Case have provided an explicit representation of the user goals and have also provided much more details about the interactions between the BPEL process and the user and other web services.

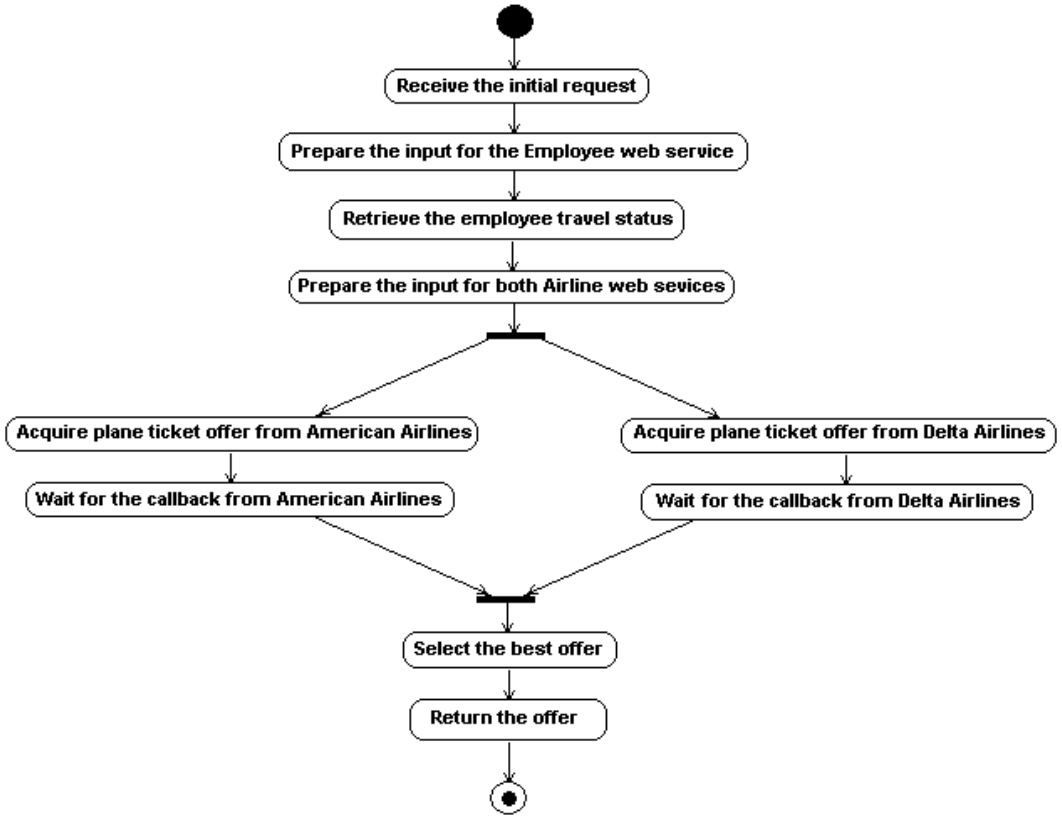

Figure 9: The resulting BPEL activity diagram for the "Acquire Best Plane Ticket Offer" BPEL business process

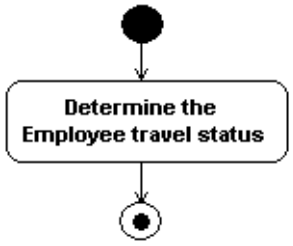

Figure 10: The resulting BPEL activity diagram for the "Check Employee Travel Status" BPEL business process

5 Conclusion

In this paper we presented proposed a user-centered approach to modeling BPEL business processes. The approach is based on using Use Cases to explicitly model the services offered to the business processes' users. Traditionally, BPEL business processes are modeled using UML activity diagrams only, which did not support the explicit modeling of user goals. Another advantage of using Use Cases is that the interaction intricacies between the business processes and the web services offered by other systems can be captured. Use Case modeling provides a high level overview of the services that are provided by a set of related BPEL business processes. This will allows E-commerce analysts to discover common services and functionality which in turn will save time and effort by avoiding the implementation of redundant functionality.

Modeling BPEL business processes using activity diagrams appeals to E-commerce analysts since activity diagrams is an excellent technique to model and visual workflows. Moreover, it can be directly mapped to BPEL code to kick start the implementation phase. Traditional Use Cases are described using unstructured natural language. Describing workflows concisely using unstructured natural language is difficult since workflows contain features such as loops, conditions and branches and concurrent flows. In this paper, we utilize the SUCD structure to describe Use Cases. The SUCD structure features structural elements that allow E-commerce analysts to describe the BPEL workflows concisely and accurately. Using the AGADUC process, which is implemented by the AREUCD tool, E-commerce analysts will be able to effortlessly generate activity diagrams that accurately represent the workflows described by SUCD Use Cases. This allows E-commerce analysts to utilize a user-centered approach to model their BPEL processes without losing the advantages of activity diagrams.

Future work can be directed towards developing a Use Case driven approach to systematically generate test suites that will check the validity of the BPEL processes.

References

[1] A-W Scheer. *ARIS – Business Process Modeling.* Springer Verlag, 1999.

[2] J. Aagedal and Z. Milosevic. ODP enterprise language: An UML perspective. In *In Proc. of The 3rd International Conference on Enterprise Distributed Object Computing.* IEEE Press, 1999.

[3] K. Bittner and I. Spence. *UC Modeling.* Addison-Wesley, 2002.

[4] M. Dumas and A. ter Hofstede. UML Activity Diagrams as a Workflow Specification Language. In *Proc. of the UML 2001 Conference*, 2001.

[5] B. P. M. Initiative. *BPMI: Business Process Modelling Notation – Specification v1.0*, November 2004.

[6] B. Korherr and B. List. Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL. In *2nd International Workshop on Best Practices of UML (BP-UML'06).* Spinger Verlag, Lecture Notes in Computer Science, November 2006.

[7] P. Kueng and P. Kawalek. Goal-based business process models: creation and evaluation. *Business Process Management Journal*, Volume 3(1):17–38, 1997. MCB Press.

[8] B. List and B. Korherr. An Evaluation of Conceptual Business Process Modelling Languages. In *Proceedings of the 21st ACM Symposium on Applied Computing (SAC'06).* ACM Press, April 2006.

[9] M. B. M. B. Juric and P. Sarang. *Business Process Execution Language for Web Services. Second Edition.* PACKT Publishing, 2006.

[10] J. M. M. El-Attar. AGADUC: Towards a More Precise Presentation of Functional Requirements in Use Case Models. In *4th ACIS International Conference on Software Engineering, Research, Management and Applications*, 2006.

[11] K. Mantell. *From UML to BPEL – http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel*, September 2003.

[12] I. Object Management Group. *UML 2.0 Superstructure, http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf*, November 2006.

[13] M. Schader and A. Korthaus. Modeling business processes as part of the BOOSTER approach to business object-oriented systems development based on UML. In *Proc. of The Second International Enterprise Distributed Object Computing Workshop.* IEEE Press, 1998.

[14] STEAM laboratory website at the University of Alberta. *http://www.steam.ualberta.ca/main/research_areas/Requirements_Capture.htm*, Dec 2006.