Cardoso, J., "Poseidon: A framework to assist Web process design based on business cases", International Journal of Cooperative Information Systems (IJCIS), World Scientific Publishing, Vol. 15, No. 1, pp. 23-55, March 2006.

# Poseidon: A framework to assist Web process design based on business cases

Jorge Cardoso
*jcardoso@uma.pt*
*Department of Mathematics and Engineering*
*University of Madeira*
*9000-390 - Funchal*

**Abstract.**
Systems and infrastructures are currently being developed to support Web services and Web processes. One prominent solution to manage and coordinate Web services is the use of workflow technology. For more than two decades now, workflow management systems architectures, language specifications, and workflow analysis techniques have been extensively studied. While these areas of research have made the development of sophisticated workflow systems possible, one important research area that has been overlooked is the lifecycle of process application development. As a result, current process management systems support the analysis, enactment, and ad-hoc design of workflows, but they lack the tools and methods to assist process design. The purpose of our study is to present a framework to assist and guide process analysts and designers in their task. The Poseidon framework includes a participative and an analytical design method. The participative phase uses a clean sheet approach and starts by constructing a business case table which captures all the business cases represented by a process. Afterward, an analytical design phase is followed where scheduling functions are derived from the business case table and are used to build the structure of a process.

## 1 Introduction

With the maturity of infrastructures that support e-commerce, it will be possible for organizations to incorporate Web services as part of their business processes. While in some cases Web services may operate in an isolated form, it is natural to expect that Web services will be integrated as part of workflows (Fensel and Bussler 2002).

Workflow systems are a valuable asset for managing e-commerce applications that span multiple organizations (Sheth, Aalst et al. 1999), since they are capable of integrating business objects for setting up e-services in an amazingly short time and with impressively little cost (Shegalov, Gillmann et al. 2001).

A wide spectrum of workflow system architectures has been developed to support various types of business processes. With small changes, these systems can also manage Web processes. Cardoso, Bostrom et al. (2004) report that more than 200 workflow products are available in the market. Most of the systems provide a set of tools which include a graphical application to design workflows and an engine or enactment system to manage the execution of workflow instances.

Research has targeted three main areas: workflow architectures, specification languages, and process analysis. The workflow architectures developed include fully distributed (Kochut, Sheth et al. 1999), database oriented (Ceri, Grefen et al. 1997), web-based (Miller, Palaniswami et al. 1998), message-based (Alonso, Mohan et al. 1994), and agent-based (Jennings, Faratin et al. 1996) architectures. Workflow research has also contributed to the creation of various specification languages to model processes. The most well-known included PIF, PSL, BPML, WPDL, WSFL, BPEL4WS, and DAML-S. Finally, another area of interest has been the analysis of processes. Since design errors can result in runtime errors which need to be repaired at high cost, it is important for organizations to check the properties (e.g. correctness and quality of service) of process definitions before they become operational. Significant work on Petri-nets and Petri-net-based analysis techniques have been used to establish the correctness of processes (Aalst 1998). Simulation has also been one of the selected approaches (Chandrasekaran, Silver et al. 2002) for process analysis. Recently an increased interest has led to research on the analysis of non-functional properties of processes (Eder, Panagos et al. 1999; Sadiq, Marjanovic et al. 2000; Son, Kim et al. 2001; Cardoso, Sheth et al. 2002).

As can be seen from the previous paragraph, workflow architectures, specification languages, and process analysis have been the major focus of the industry and research. These areas of research are of recognized importance for the construction of sophisticated and robust workflow systems. Nevertheless, one important area has been overlooked, the research of the lifecycle of process application development.

Studies on the lifecycle of process development have been reduced and are almost inexistent. In 1996, Sheth, Georgakopoulos et al. (1996) established that workflow and process modeling was one of the outstanding research issues which should be investigated. Several years later, Casati, Fugini et al. (2002) recognized that despite the diffusion of workflow systems, methodologies covering the phases of workflow application development are still missing.

The lifecycle of workflow applications development is comparable to the lifecycle of software development (Sommerville 2000). It starts with the identification of requirements and goes on through the design of workflows. Later phases include the implementation, testing, and documentation of the developed applications. The use of adequate methodologies to assist the lifecycle of processes development is a key determinant to the success of any workflow project and requires the availability of specific tools – different from the ones used in software development – to model each phase of the cycle.

In this paper, we argue that better methodological support for stepwise creation of Web processes and workflows that can ensure the fulfillment of business processes' strategic goals is necessary. The design of processes can be a challenging task, especially when they are complex. The design methodology needs to guarantee that the created process definitions are correct and meets the real needs of its users. There is a call for frameworks and tools which can assist the process analyst and designer to manage the inherent complexity in Web process applications.

Casati et al. (2002) have undertaken one of the most comprehensive studies on workflow development. They present a methodology for developing workflow applications from analysis to implementation. The methodology uses UML, UML extensions for business process modeling, and introduces new extensions to model particular features of workflows, such as exception handling and transaction management. Their work aims to supply an integrated and uniform method to cover the whole workflow development lifecycle. Casati et al. (2002) also reports that some workflow systems vendors have started to provide methodological support to the design of workflows, such as AdminFlow provided by HP.

Reijers et al. (Reijers 2003; Reijers, Limam et al. 2003) present a method to design or redesign a process from a product-based approach. An example given of a product that can derive a process production is the Bill of Materials (BOM). The method presented can be used to design a process based on an existing process, from a clean-sheet approach, or using a reference model. Once one of these three approaches has been selected, an analytical technique is followed using formal theory to derive a process design.

Sadiq and Orlowska (1999) propose a framework for workflow modeling based on layers. The idea is to partition workflows into several graph layers rather than modeling all the workflow constraints on a single graph. The first phase of the modeling framework – the structural modeling – captures the flow of execution. According to Sadiq and Orlowska this is the primary and, perhaps, the most important aspect of a workflow model. Unfortunately, in their work no methodology is proposed to drive and guide the structural modeling phase. Only an overall discussion on the basic elements of workflow graphs (i.e., tasks, conditions, sequences, choice, nesting, iterations, splits and joins) is given.

Cardoso and Teixeira (1998) describe how a graphical process modeling language (STRIM (Ould 1995)) can be use to model workflows. Later, Cardoso and Sheth (2003) worked on the development of mechanisms and algorithms to facilitate the composition of workflows that model Web process applications. Their approach relies on the use of the emerging Semantic Web and ontologies to overcome some of the integration problems, such as the interoperability of heterogeneous Web services.

Compared to Casati et al. (2002), our work targets, not the modeling of the various phases of the lifecycle of workflow development, but the development of a framework to assist process analysts and designers to design Web processes and workflows. The framework, named Poseidon, is to be used during the requirement and design phase. Poseidon can be viewed as a methodology which structures a comprehensive set of steps that drives the design of workflows based on requirements gathered from communication with staff, managers, and experts.

Compared to Reijers et al. (Reijers 2003; Reijers, Limam et al. 2003) work, our approach is not dominantly analytical and includes a participative and an analytical design method. The process design is not driven by a product, but by business cases.

In contrast to Sadiq and Orlowska's work (Sadiq and Orlowska 1999), we have proposed a framework that not only describes the basic elements of a process, but also provides instructions to guide process developers to place and organize the various elements of a process graph. Compared to our previous research (Cardoso and Teixeira 1998; Cardoso and Sheth 2003), the work presented in this paper is complementary. In (Cardoso and Teixeira 1998) we target the graphical representation of processes and workflows (i.e., design phase), then in (Cardoso and Sheth 2003) our intention is to assist the designer to map tasks to Web services (i.e., implementation phase). Now, our objective is to develop a solution to semi-automatically create a process design from the information gathered during the requirement phase. The framework facilitates the construction of workflows through complexity decomposition. In our approach, firstly, business cases are identifies, and then sequential and parallel building blocks are created, whereupon conditional blocks are established, before, finally simplifying and implementing the workflow.

This paper is structured in the following way. Section 2 presents the requirements and methodology of our framework. In section 3, we describe the design analysis phase of workflows and present the Poseidon framework. Section 4 gives the evaluation of the framework and presents future work. Finally, section 5 presents our conclusions.

## 2 Poseidon: Requirements and Methodology

In organizations where business processes are repetitive and predictable, such as banking and insurance, workflow systems are becoming of critical economic importance. The importance of workflow systems is also becoming clear in the context of Web services and Web process. Nevertheless, experience in designing and deploying workflows has shown that methods to support process development are practically inexistent.

Thus, new techniques and methods for the specification, design, implementation, integration, testing, and maintenance are necessary to increase the quality of workflows and reduce the effort required to produce them.

Due to the lack of suitable tools to assist the development and deployment of Web processes and workflows, we have created the Poseidon framework. The framework is to be used by process analysts and designers during the two first phases of the process development lifecycle, i.e. the requirement gathering and design phases. In the next two sections we enumerate the initial specifications that have originated the Poseidon framework and give a brief overview of the methodology and rationale behind the tool.

### 2.1 Framework Requirements

The Poseidon framework is a basic conceptual structure composed of steps, procedures, and algorithms that determine how process design is to be approached.

An organization can select different frameworks to design different types of Web processes. There is no such a thing as a ‘right’ or ‘wrong’ process development framework. Nevertheless, some frameworks are more suitable than others for some type of process application development. If an inadequate framework is selected and used, this will probably reduce the value and utility of the Web process application deployed.

Since we recognize that one-size-fits-all frameworks cannot meet the needs of diverse organizations, we have customized and developed Poseidon bearing the following requirements in mind:

- **Simplicity and ease of use.** The methodology should be easily understood by its end users, i.e., process analysts and designers and interviewees possibly without workflow expertise (e.g. CEOs, managers, and employees.)
- **Business process size.** The framework should support the modeling of small and medium size business processes. We approximately classify small workflows as having up to 15 tasks and medium size workflow as having roughly 30 tasks.

- **Business process structure.** The framework should be better suited for production and administrative processes (McCready 1992) – than to *ad hoc* ones – since they are more structured, predictable, and repetitive.
- **Degree of automation.** Based on one of the main advantages and aims of workflow systems – automation – we hope to develop a methodology which automates as many of its steps as possible. This will allow for faster process development with lower design costs.

## 2.2 Methodology

The intention of this section is to give an overall description of our framework to construct process graphs based on the knowledge gathered from interviews, group brainstorming sessions, and meetings (in this paper we will use the term 'interview' to designate these three methods of gathering knowledge).

The interviews are essentially carried out between process analysts and people who have the expertise and knowledge of the processes' business logic. The latter group might, typically include people such as administrative staff, department managers, and mid-range managers, and even CEOs.

Our framework involves four main phases and for each one we present a mechanism to support and assist its concretization. In the first phase, by means of interviews, we build a business case table. The table captures the various cases that a business process describes. The basic property of a process is that it is case-based (Aalst 1998). This means that every task is executed for a specific case.

In the second phase, we extract a set of scheduling functions from the business case table. For each task, a scheduling function is extracted. A scheduling function is a Boolean function for which the parameters are business variables from the business case table. Each function models the scheduling of a task at runtime, i.e. for a given set of business variables and their assertion, the function indicates if a task is scheduled at runtime or not.

The third phase consists of using the scheduling functions from the previous phase and identifying the sequential and parallel building blocks (Figure 1) that will make up the process in development. In Figure 1, nodes represent tasks/activities and arrows represent transitions. In this phase, we also establish the non-deterministic routing of a process, i.e. the conditional building blocks.

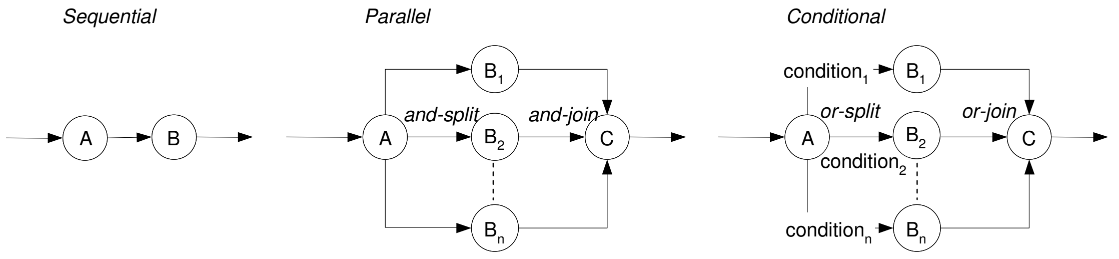


Figure 1. Example of basic process building blocks

The current framework captures the five basic patterns described in (Aalst, Barros et al. 2000): sequential routing, AND-split, AND-join, XOR-split, and XOR-join. Those are the most important patterns, since most process languages are able to model sequential, parallel, and conditional routing. Tasks associated with sequential and parallel building blocks are executed in a deterministic fashion, while conditional blocks are examples of non-deterministic routing. Conditional blocks indicate that the scheduling of a task depends on the evaluation of a Boolean condition or expression (see Figure 1).

The framework can be extended to include and model additional patterns (Aalst describes 21 different patterns). Nevertheless, this extension may transform the framework into a difficult tool to understand. Since one of the principal requirements, expressed in section 2.1, was simplicity, it is important to analyze the benefits and costs of adding new patterns to the framework.

In the last phase, we integrate the basic building blocks previously identified with the non-deterministic blocks. The process is cleanup of any dummy (null) tasks and, if necessary, the process may be slightly restructured or modified for reasons of clarity.

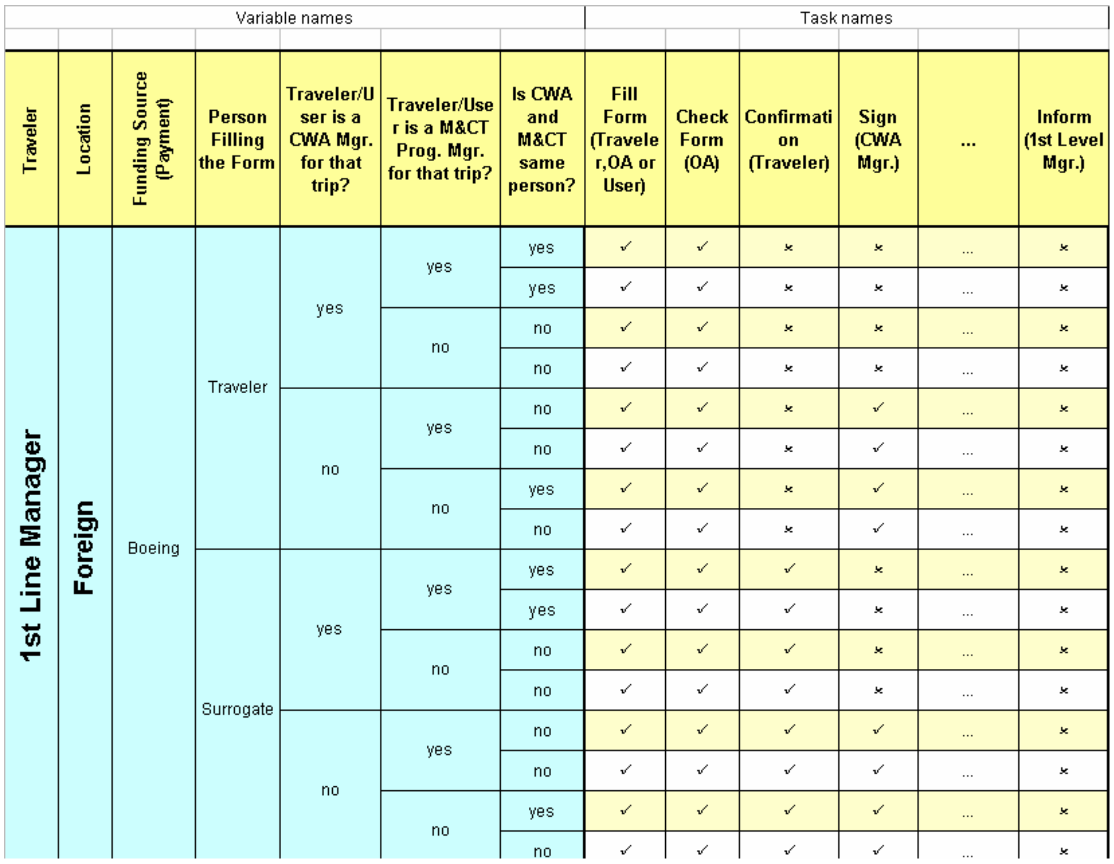
## 3 Poseidon framework

The input of the framework presented in this paper is a set of task names, and the output is a process model or workflow. The process models include tasks or Web services, transitions, control flow variables, and control flow conditions. The framework relies heavily on interviews to supply the knowledge which cannot be inferred automatically.

As already explained, the framework has four major steps which are discussed individually in the following sections. These are the construction of business case tables, extraction of scheduling functions, identification of basic block structures, and the cleaning and implementation of the process graph.

### 3.1 Business case table

To capture all the cases represented in a process, we introduce the concept of business case table (a partial example of such a table is shown in Figure 2). The table has the main advantage of being a simple, yet powerful, tool to capture and describe business cases.



| Variable names | | | | | | | Task names | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Traveler | Location | Funding Source (Payment) | Person Filling the Form | Traveler/User is a CWA Mgr. for that trip? | Traveler/User is a M&CT Prog. Mgr. for that trip? | Is CWA and M&CT same person? | Fill Form (Traveler,OA or User) | Check Form (OA) | Confirmation (Traveler) | Sign (CWA Mgr.) | ... | Inform (1st Level Mgr.) |
| 1st Line Manager | Foreign | Boeing | Traveler | yes | yes | yes | ✓ | ✓ | ✗ | ✗ | ... | ✗ |
| | | | | | | yes | ✓ | ✓ | ✗ | ✗ | ... | ✗ |
| | | | | | no | no | ✓ | ✓ | ✗ | ✗ | ... | ✗ |
| | | | | | | no | ✓ | ✓ | ✗ | ✗ | ... | ✗ |
| | | | | no | yes | no | ✓ | ✓ | ✗ | ✓ | ... | ✗ |
| | | | | | | no | ✓ | ✓ | ✗ | ✓ | ... | ✗ |
| | | | | | no | yes | ✓ | ✓ | ✗ | ✓ | ... | ✗ |
| | | | | | | no | ✓ | ✓ | ✗ | ✓ | ... | ✗ |
| | | | Surrogate | yes | yes | yes | ✓ | ✓ | ✓ | ✗ | ... | ✗ |
| | | | | | | yes | ✓ | ✓ | ✓ | ✗ | ... | ✗ |
| | | | | | no | no | ✓ | ✓ | ✓ | ✗ | ... | ✗ |
| | | | | | | no | ✓ | ✓ | ✓ | ✗ | ... | ✗ |
| | | | | no | yes | no | ✓ | ✓ | ✓ | ✓ | ... | ✗ |
| | | | | | | no | ✓ | ✓ | ✓ | ✓ | ... | ✗ |
| | | | | | no | yes | ✓ | ✓ | ✓ | ✓ | ... | ✗ |
| | | | | | | no | ✓ | ✓ | ✓ | ✓ | ... | ✗ |

**Figure 2. Example of a business case table**

Each business case corresponds to an entry in the table and establishes the task scheduled at runtime based on business variables assertion. Business variables are variables that influence the routing or control-flow in a process. For example, in a banking Web process application, the business variable <u>Loan Amount</u> determines the acceptance or rejection of a loan request. If the variable has a value greater than $500.000, then the loan is rejected and the task '*reject*' is executed, otherwise the task '*accept*' is executed.

Business variables are identified during the participative phase in which CEOs, managers, and staff members are interviewed alone or in small groups. Each business variable has a domain, also identified during the participative phase.

The domain identifies the values that a business variable can take. For example, the business variable <u>Traveler</u> can take the values "1st Line Manager", "2nd Line Manager", "3rd Line Manager", and "Non-Manager", i.e.,

<u>Traveler</u> = {"1st Line Manager", "2nd Line Manager", "3rd Line Manager", "Non-Manager"}

### 3.1.1 Business case table schema

A business case table is based on a two dimensional table. The schema of the table is the following. The columns are divided into two classes. The first class regroups a set of business variables, while the second class includes the tasks that are part of a process. Each entry of the table relates business variables and tasks with information indicating if a task is to be scheduled at runtime or not.

The first cells of each row, corresponding to the columns of the first class, contain values that can be assigned to business variables. For example, in Figure 2, the value "Foreign" has been assigned to the business variable <u>Location</u> and the value "Traveler" has been assigned to the variable <u>Person Filling the Form</u> for the first eight data rows.

The data cells corresponding to the columns of the second class contain information indicating if a particular task is to be scheduled at runtime or not. The idea is to establish if a given task is to be scheduled based on the assertion of a set of business variables. Therefore, each data cell results from the intersection of a column (with a task name) and a row (with a set of asserted business variables.) A business variable is a variable on which the execution of a set of tasks may depend. Formally, we are interested in evaluating for each $task_t$ the following function, where $bv_i$ is a business variable:

$$scheduled(task_t, bv_1, bv_2, \ldots, bv_n) \in \{✓, ✗\} \qquad \text{(function 1)}$$

A data cell corresponding to the columns of the second class may contain the scheduled symbol (✓) or the not-scheduled symbol (✗). The scheduled symbol indicates that a given task is to be scheduled at runtime when the business variables are asserted to particular values. In Figure 2 for example, if <u>Traveler</u> = "1st Line Manager", <u>Funding Source (Payment)</u> = "Boeing", <u>Person Filling the Form</u> = "Traveler", <u>Traveler/User is a CWA Mgr. for that trip?</u> = "Yes", <u>Traveler/User is a M&CT Prog. Mgr. for that trip?</u> = "Yes", and <u>Is CWA and M&CT same person?</u> = "Yes", then the task *Check Form (OA)* is scheduled for execution at runtime; the cell contains the scheduled symbol (✓). On the other hand, this is not true for the task *Confirmation (Traveler)*, since for the same assertions the data cell contains the not-scheduled symbol (✗).

### 3.1.2 Business case table construction

Understanding the business case table schema is relatively easy, whereas its construction is far more challenging and complex. The methodology to construct and fill the table with business cases is an iterative process.

Initially, the table schema has only columns of the second class (i.e., each column represents a process task, see Figure 3.a) and no business variable exists in the table. The columns are identified by the process analyst and staff members of an organization as the necessary functions to archive a goal or an objective. While we do not present a method to establish this initial set of columns, this can be accomplished using the methodology proposed in (Casati, Fugini et al. 2002) for the analysis phase of workflow application development. It is convenient – but not essential – to order the tasks according to their probable chronological order of execution. This allows users to feel more comfortable, thus, making it easier for them to understand the case table.

In the first step, the process analyst should enquire of the interviewees if the tasks presented in the case table are always scheduled during the execution of a process. If a task is always scheduled, it receives the scheduled symbol (✓); if it is never scheduled it receives the non-scheduled symbol (✗). In some cases, both symbols need to be assigned to the same data cell. This is called a symbol conflict. We will see later that this situation suggests that the task is scheduled under some logical condition.

a)

| Task A | Task B | Task C |
|---|---|---|
| ✓,✗ | ✓,✗ | ✓,✗ |

a.1)

| Check form | Sign | Notify |
|---|---|---|
| ✓ | ✓,✗ | ✓,✗ |

b)

| Variable 1 | Task A | Task B | Task C |
|---|---|---|---|
| value 1.a | ✗ | ✓ | ✗ |
| value 1.b | ✓ | ✗ | ✗ |
| value 1.c | ✗ | ✓ | ✓ |

b.1)

| Manager | Check form | Sign | Notify |
|---|---|---|---|
| 1 Line Manager | ✓ | ✓ | ✓ |
| 2 Line Manager | ✓ | ✓,✗ | ✓,✗ |
| 3 Line Manager | ✓ | ✗ | ✓,✗ |

c)

| Variable 2 | Variable 1 | Task A | Task B | Task C |
|---|---|---|---|---|
| value 2.a | value 1.a | ✗ | ✓ | ✗ |
| value 2.b | | ✗ | ✓ | ✓ |
| value 2.a | value 1.b | ✓ | ✗ | ✗ |
| value 2.b | | ✗ | ✗ | ✗ |
| value 2.a | value 1.c | ✗ | ✓ | ✓ |
| value 2.b | | ✓ | ✓ | ✓ |

c.1)

| Travel | Manager | Check form | Sign | Notify |
|---|---|---|---|---|
| National | 1 Line Manager | ✓ | ✓ | ✓ |
| Foreign | | ✓ | ✓ | ✓ |
| National | 2 Line Manager | ✓ | ✗ | ✗ |
| Foreign | | ✓ | ✓ | ✓ |
| National | 3 Line Manager | ✓ | ✗ | ✗ |
| Foreign | | ✓ | ✗ | ✓ |

**Figure 3. Constructing the business logic case table**

After having analyzed and set symbols for each task individually, the table may be in one of three states.

1) All the tasks have received the scheduled symbol (✓). This means that all the tasks are scheduled at runtime. There is no non-determinism. Therefore the process does not have any business variable and the tasks are scheduled in sequence or in parallel. The only remaining step to carry out is to set a total order for the tasks. This is relatively easy since no conditional routing exists. No further steps are necessary and the process design is completed.

2) All the tasks have received the non-scheduled symbol (✗). In this case, the process does not schedule any task at runtime. This is the same as saying that the process does not have any tasks. In practice, this situation should never occur. It indicates that the methodology followed to establish the tasks of a process was inadequate.

3) Finally, the last state indicates that the business case table contains one or more symbol conflicts, since the two available symbols have been assigned to the same data cell.

Let us dwell on this last state. Consider the row of table b.1) in Figure 3 which asserts the value "2 Line Manager" to the business variable <u>Manager</u>. The row contains a symbol conflict for the task *Sign* and *Notify*, since both scheduling symbols, ✓ and ✗, are present. This state of affairs was arose because during an interview the interviewee expressed that the 2nd Line manager did not always need the signature of his supervisor (represented with the *Sign* task) to travel on a business trip. Furthermore, the 2nd Line manager also did not always need to notify his supervisees (represented with the *Notify* task) of his journey. The phrase "… the 2nd Line manager did not always need …" in the previous paragraph indicates the existence of a symbol conflict. In our example, the conflict points out the existence of a business variable – not yet present in the business case table – controlling the 2nd Line manager's actions.

After the first step, if the business case table is in state 1 or 2, then the procedure stops here. On the other hand, if the case table is in state 3, then we need to remove symbolic conflicts using a procedure called symbol conflict resolution discussed in the next section.

### 3.1.3 Symbol conflict resolution

Symbol conflicts indicate that the scheduling of a set of tasks depends on one or more business variables. To resolve a symbol conflict, the process analyst – with the help of interviewees – should identify at least one business variable that controls the scheduling of a conflicting task. The process analyst should also identify the business variable's domain and establish the task which asserts the variable.

When such a variable is identified the following steps are taken. A column is added to the left side of the business case table and rows are added to the table. The column is labeled with the name of the business variable identified. Each row of the table is duplicated *n*-1 times, where *n* is the domain set cardinality of the newly introduced business variable. The data cells corresponding to the new business variable column are set the values of its domain. For example, in Figure 3, table b.1), the variable Manager is assigned to "1 Line Manager", "2 Line Manager", and "3 Line Manager".

Once the table's schema is updated to reflect the introduction of a new business variable, the data cells must also be updated with appropriate scheduling symbols. During the duplication of data rows, the rows with and without symbol conflicts are duplicated, but only the rows with a symbol conflict need to be updated. As previously, the process analyst should carry out (additional) interviews to determine which tasks are scheduled at runtime based on the business variables present in the table. Each row with a symbol conflict is individually analyzed and its contents are discussed with interviewees.

For example, in Figure 3, the business variable Travel has been identified and added to the table. The domain of the variable is {"National", "Foreign"}. Its cardinality is equal to 2. Therefore, each rows of table b.1) is duplicated once (i.e., 2-1 times). The data cells of the variable Travel have been set to "National" and "Foreign". Please note that adjacent business variable data cells with identical content, such as the one from the variable Manager, have been merged for practical reasons. The data row number 1 of table b.1) has been duplicated to the data row number one and two of table c.1). There is no need to update the new rows since no conflict was present before the duplication. On the other hand, the data row number 2 of table b.1) has at least one symbol conflict. Thus, after the duplication, the rows need to be updated. In this particular example, after updating all the duplicated rows, the case table does not contain any symbol conflict. This means that it is not necessary to add any other variable to the table, since all the symbol conflicts have been resolved with the introduction of the Travel and Manager business variables. Consequently, the procedure for constructing the business case table is complete.

After the row duplication, analysis, and update, it may become apparent that some symbol conflicts have been resolved, while others still remain in the table. In such cases, the conflict resolution procedure needs to be reapplied until all the symbol conflicts have been removed (please note that when applying the procedure it is possible to add more than one business variable at a time.)

### 3.1.4 Handling large business case tables

In our example from Figure 3, all the symbol conflicts have been resolved with the introduction of only two business variables. However, in practice and depending on the size and complexity of the process being modeled, various variables may be required to remove all the symbol conflicts. In some cases, the size of the business table can become fairly large. For example, a medium size process (15-30 tasks) with

6 business variables, with the following domain cardinality 4, 3, 3, 2, 2, and 2, can easily generate a table with approximately 288 rows.

Two techniques can be employed to deal with large tables: business case table fragmentation and process restructuring. The first technique consists of fragmenting the table into smaller tables based on business variable values. Each smaller table is then placed on a different sheet. For example, the table c.1) in Figure 3 can be fragmented into smaller table based on the Manager variable. The aim is to place the rows for which the variable Manager has the value "1 Line Manager" on a separate sheet. The same procedure is then carried out for the "2 Line Manager" and "3 Line Manager". This technique has been found to be extremely useful. Interviewees respond in a better way when they are shown smaller tables, mostly because the amount of information presented at a time is smaller. Also, locking one or more business variables to a specific value reduces the complexity of the table, making its interpretation easier for both the process analysts and interviewees.

Our development of workflows has involved the use of fragmentation technique to reduce the complexity of the business case table. We have used a standard spreadsheet application (Microsoft Excel) to design and manage the various fragmented tables. Our initial table had seven business variables and 176 data rows. The fragmentation has generated 11 tables of 16 data rows each.

The second technique consists of restructuring the tasks of a process into sub-processes, creating a tree-like hierarchy of tasks and sub-processes. Once such a structure is created, the methodology presented to construct business case tables can be applied to the smaller sub-processes individually. Nevertheless, problems may arise when sub-processes are not self-contained from a business variable perspective, i.e. a sub-process routing depends on a business variable defined in its parent process. When this occurs, the business case table of a sub-process must refer to the business variable defined in the parent process. This increases the complexity and reduces the semantics of the table.

A useful enhancement that can be performed is to reorganize the business variable columns at any time to increase the clarity of the table's information without affecting its validity. For example, in Figure 3, it is possible to switch the position of the columns Travel and Manager making the information presented clearer to the process analysts and interviewees.

### 3.1.5 Methodology summary

As a summary, Figure 4 describes the main steps involved during the construction of a business case table.

```
Methodology Create_Business_Case_Table( set-of-tasks )
1) Create a two-dimensional table with $t$ columns, where $t$ is the number of tasks in set-of-tasks.
2) Label each column with the name of a task in set-of-tasks.
3) Based on interviews, set the symbols ✓ and × in the first data row of the table
4) If no conflicting symbols exist in the table, then the procedure is over; the workflow is
deterministic and it is only necessary to set sequential and parallel building blocks.
5) Otherwise, while conflicting symbols exist then
    5.1) identify at least one business variable that removes at least one conflicting symbol of the
         table
    5.2) establish the domain and domain cardinality ($n$) of the business variable(s)
    5.3) establish in which task the variable(s) is asserted
    5.4) add the business variable(s) to the left side of the table
    5.5) duplicate each data row $n$-1 times
    5.6) update the rows with conflicting symbols based on information gathered by means of
         additional interviews
6) End while
```
**Figure 4. Methodology to construct a business case table**

## 3.2 Extracting scheduling functions from the business case table

Once the business case table has been created, we are interested in extracting and minimizing the Boolean expressions from the scheduling table (see equation 1) that rule the scheduling of tasks. The extracted functions are logic disjunctions of conjunctions of business variables.

Using Boolean algebra to simplify Boolean expressions can be awkward, apart from being laborious. Furthermore, this approach can lead to solutions, which, though they appear minimal, are not. The Quine-McCluskey (McCluskey 1956) method and Karnaugh maps (Karnaugh 1953) provides a simple and straightforward method of minimizing Boolean expressions.

While these techniques are based on the use of a power of 2 encoding, this low level of detailed is hidden from the business analysts, since it is only used by the Quine-McCluskey method or Karnaugh map to minimize Boolean expressions.

The Karnaugh technique can be employed to construct scheduling functions with 3 and 4 business variables. It is possible to create functions with 5 and 6 inputs, but these can become unwieldy and difficult to construct. Furthermore, this technique is not-automated. As a result, this technique is only considered to be adequate for small process applications.

The Quine-McCluskey method, which is also known as the tabular method, is particularly useful when extracting scheduling functions with a large number of business variables. Additionally, computer programs have been developed employing this algorithm. The use of this technique increases the degree of automation of our methodology. Remember, that this was one of our initial goals.

Another alternative that can be considered to minimize Boolean expressions from the scheduling table is the use of Binary Decision Diagrams, also known as BDD (Drechsler and Sieling 2001). If the reader decides to select this method to minimize Boolean expressions, it is importance to realize that, while this method is more powerful than the two previous techniques, it is also more complex. Based on our requirements and objectives we have decided to use the Quine-McCluskey method.

To extract scheduling functions from the business case table using Karnaugh maps or the Quine-McCluskey method, we first need to map a business case table to a truth table. The mapping can be achieved in the following way.

1) For each business variable, use the formula $\lceil \log_2(\textit{domain cardinality}) \rceil$, where *domain cardinality* is the cardinality of the domain's variable, to determine the minimum number of bits necessary to represent the variable. For example, the variable <u>Travel</u> from Figure 3 has a domain with only two values ("National" and "Foreign"), then only one bit is necessary to represent the variable. The variable <u>Manager</u> has a domain with three distinct values ("1 Line Manager ", "2 Line Manager", and "3 Line Manager") and thus two bits are necessary to represent the variable.

2) Create a mapping between each business variable value and a binary number, starting with '0'. For example, the domain values of the variable <u>Travel</u>, "National" and "Foreign", can be mapped to '0' and '1', respectively. The domain values of the variable <u>Manager</u>, "1 Line Manager ", "2 Line Manager", and "3 Line Manager", can be mapped to '00', '01', and '10', respectively.

3) Map the symbols ✓ and ✗ to the Boolean domain {0, 1}. The symbol ✗ is mapped to '0' and the symbol ✓ is mapped to '1'.

4) Create a new table using the two mappings described previously. Figure 5 shows the result of mapping the table from Figure 3 to a truth table. Please note that besides applying the mappings, we have made the following adjustment. We have added, to the generated truth table, a set of new variables 'a', 'b', and 'c', and functions 'w', 'y', and 'z'. The variable and function symbols have been added to simplify the handling of the truth table in the following steps. Also, we have switched the position of the <u>Manager</u> and <u>Travel</u> columns (as recommended in the previous section), making the truth table input values follow a standard binary sequence (000, 001, 010, 011, …).

5) The functions of the truth table may be incompletely specified; that is, certain input combinations will never occur. Therefore, the output may be undefined for some of the input combinations. In our example, the input combinations '110' and '111' never occur. In this case, the process analyst needs to add the missing input combinations to the truth table and represent the output values associated with question marks in the table.

a)

| Check Form | Check Form | | | |
|---|---|---|---|---|
| Travel | Manager | Check form | Sign | Notify |
| National | 1 Line Manager | ✓ | ✓ | ✓ |
| Foreign | | ✓ | ✓ | ✓ |
| National | 2 Line Manager | ✓ | × | × |
| Foreign | | ✓ | ✓ | ✓ |
| National | 3 Line Manager | ✓ | × | × |
| Foreign | | ✓ | × | ✓ |

↓

b)

| | Check Form | | Check Form | | | |
|---|---|---|---|---|---|---|
| | Manager | | Travel | Check form | Sign | Notify |
| | a | b | c | w(a,b,c) | y(a,b,c) | z(a,b,c) |
| | 0 | 0 | 0 | true | true | true |
| | | | 1 | true | true | true |
| | 0 | 1 | 0 | true | false | false |
| | | | 1 | true | true | true |
| | 1 | 0 | 0 | true | false | false |
| | | | 1 | true | false | true |
| Added rows | 1 | 1 | 0 | true | false | false |
| | | | 1 | true | false | false |

**Figure 5. Mapping a business case table to a truth table.**

Applying one of the presented methods to the truth table b) from Figure 5, we obtain the scheduling functions indicated in the scheduling table from Figure 6.

| Variable | Task | Function |
|---|---|---|
| a,b,c | Check Form | w(a, b, c) = true |
| | Sign | y(a, b, c) = (¬a∧¬b) v (¬a∧b∧c) |
| | Sign(1) | y1(a, b, c) = ¬a∧¬b |
| | Sign(2) | y2(a, b, c) = ¬a∧b∧c |
| | Notify | z(a, b, c) = (¬a∧¬b) v (¬a∧b∧c) v (a∧¬b∧c) |
| | Notify_m | z1(a, b, c) = ¬a∧¬b |
| | Notify_u | z2(a, b, c) = ¬a∧b∧c |
| | Notify_c | z3(a, b, c) = a∧¬b∧c |

**Figure 6. Scheduling table constructed from the truth table**

The table contains the task names and respective scheduling functions. On the left side of the table, the column *Variable* indicates in which tasks the business variables are asserted. For example, in Figure 6, the variable column indicates that the business variables 'ab' and 'c' are asserted in the task *Check Form*. Since 'ab' has been mapped to Manager and 'c' to Travel, this indicates that the task *Check Form* asserts the business variables Manager and Travel.

When the scheduling functions are disjunctions of conjunctions, synonym tasks need to be created. Synonym tasks have exactly the same behavior, execution, semantics, and only their names differ. The number of disjunctions in a scheduling function sets the number of synonyms to be created for a given task. Each of the disjunctions is associated with a synonym task. For example, since the task *Notify* has a scheduling function of the form (¬a∧¬b)∨(¬a∧b∧c)∨(a∧¬b∧c), three synonym tasks are created: *Notify_m*, *Notify_u*, and *Notify_c*. The scheduling functions are decomposed into the terms ¬a∧¬b, ¬a∧b∧c, and a∧¬b∧c, and each term is associated with one of the synonym tasks.

## 3.3 Identify Basic Block Structures

Process languages can be characterized in terms of the fundamentals building blocks they support to model the control-flow of processes. Several workflow patterns have been already identified, analyzed and documented. Aalst, Barros et al. (2000) has done the most comprehensive work in this field. They have identified 21 workflow patterns addressing comprehensive workflow functionality. The expressiveness and power of process languages can be evaluated according to the set of patterns supported.

Business process management systems are process-centric, focusing on the management of control-flow logic. Typical control-flow logic includes sequential, AND-split, AND-join, OR-split, OR-join, XOR-split, and XOR-join primitives. The sequential primitive models serial block structures. AND-split and AND-join primitives model parallel block structures. While, OR-split, OR-join, XOR-split, and XOR-join model conditional block structures.

The aim of this section is to use the scheduling table previously constructed to a) identify sequential and parallel block structures associated with a process and b) organize these basic blocks using conditional block structures.

Before continuing with the explanation of how to identify and construct basic building blocks, let us illustrate a more complex process. Consider the business case table shown in Figure 7. The table systematizes the various cases that may occur in a *Request Travel Authorization* business process. After applying the rules from section 3.2 to the case table, we obtain the truth table shown also in Figure 7.

| *Check Form* | *Sign* | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Role | Signature | Check Form | Sign | Book Flight | Book Hotel | Reservation | Send Tickets | Notify Manager | Reject | Not Auhtorized | Notify |
| Researcher | No | ✓ | ✓ | × | × | × | × | ✓ | ✓ | × | ✓ |
| | Yes | ✓ | ✓ | × | × | ✓ | ✓ | ✓ | × | × | ✓ |
| Manager | No | ✓ | × | ✓ | ✓ | × | × | × | × | × | ✓ |
| | Yes | ✓ | × | ✓ | ✓ | × | × | × | × | × | ✓ |
| User | No | ✓ | × | × | × | × | × | × | × | ✓ | ✓ |
| | Yes | ✓ | × | × | × | × | × | × | × | ✓ | ✓ |

↓

| *Check Form* | | *Sign* | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Role | | Signature | Check Form | Sign | Book Flight | Book Hotel | Reservation | Send Tickets | Notify Manager | Reject | Not Auhtorized | Notify |
| a | b | c | q(a,b,c) | r(a,b,c) | s(a,b,c) | t(a,b,c) | u(a,b,c) | v(a,b,c) | w(a,b,c) | x(a,b,c) | y(a,b,c) | z(a,b,c) |
| 0 | 0 | 0 | true | true | false | false | false | false | true | true | false | true |
| | | 1 | true | true | false | false | true | true | true | false | false | true |
| 0 | 1 | 0 | true | false | true | true | false | false | false | false | false | true |
| | | 1 | true | false | true | true | false | false | false | false | false | true |
| 1 | 0 | 0 | true | false | false | false | false | false | false | false | true | true |
| | | 1 | true | false | false | false | false | false | false | false | true | true |
| 1 | 1 | 0 | true | false | false | false | false | false | false | false | false | true |
| | | 1 | true | false | false | false | false | false | false | false | false | true |

**Figure 7. Business case table for the request travel authorization process**

From the truth table, the scheduling functions represented in Figure 8 are extracted. During the construction of the business case table, the tasks asserting business variables were identified, as described in section 3.1. For example, the business variables 'Role' and 'Signature' are asserted by the tasks *Check Form* and *Sign*, respectively. This entails that the Boolean variables 'a' and 'b' are asserted by the tasks *Check Form* and *Sign*, respectively.

| Variable | Task | Function |
|---|---|---|
| a,b | Check Form | q(a,b,c) = true |
| c | Sign | r(a,b,c) = ¬a∧¬b |
|  | Book Flight | s(a,b,c) = ¬a∧b |
|  | Book Hotel | t(a,b,c) = ¬a∧b |
|  | Reservation | v(a,b,c) = ¬a∧¬b∧c |
|  | Send Tickets | u(a,b,c) = ¬a∧¬b∧c |
|  | Notify Manager | w(a,b,c) = ¬a∧¬b |
|  | Reject | x(a,b,c) = ¬a∧¬b∧¬c |
|  | Not Authorized | t(a,b,c) = a∧¬b |
|  | Notify | z(a,b,c) = true |

**Figure 8. Scheduling table for the Request Travel Authorization business process**

### 3.3.1 Sequential and Parallel Building Blocks

The objective of this step is to identify sequential and parallel structures, and define a partial order for the tasks associated with these structures. To complete this step, the following activities are performed:

1) Create a set S of sets $s_i$, where each set $s_i$ contains all the tasks that have the same scheduling function,
2) Label each set with its scheduling function,
3) For each set $s_i$, establish existing sequential and parallel building blocks, set a partial order for the tasks

In the first activity, we produce a set S of scheduling sets $s_i$, where each set $s_i$ contains all the tasks that have the same scheduling function. The idea is to create sets of tasks with the following property: if a task of set $s_i$ is scheduled at runtime, then all of the tasks in $s_i$ are also scheduled. In our running example, $S = \{s_1 ... s_6\}$, where $s_1$={Check Form, Notify}, $s_2$={Book Flight, Book Hotel}, $s_3$={Sign, Notify Manager}, $s_4$ = {Reject}, $s_5$ = {Reservation User, Send Tickets User}, and $s_6$ = {Not Authorized}.

The second activity associates each set with a scheduling function label. For example, the set $s_1$ is labeled with '1' and the set $s_2$ is labeled with '¬a∧b'.

Finally, the last activity establishes the sequential and parallel building blocks and defines a partial order for each set $s_i$. Each set $s_i$ can be organized using a sequential and/or a parallel basic building block structure. Conditional structures cannot occur for the sets $s_i$ since non-determinism has already been captured with the scheduling functions (the set up of conditional blocks is described in the next section.)

The first two activities can be automated, while the third one requires human intervention. Nevertheless, we believe that this last activity can be partially automated. One possible approach would be to analyze data dependencies and information dependencies between tasks. A data dependency exists between two tasks if the input of a task depends on the output of the other. An information dependency exists between two tasks if the content or presentation of one task logically follows the content of another. For example, let us consider that a sequence of tasks is to be used to display a business contract to a user. Since several sections of the document need to be accepted individually, it has been decided to fragment the document into parts. Each part has been associated with a task requiring human intervention. In this simple case, information dependency exists between the tasks, since the tasks needs to be ordered in such a way that the contract is read in a sequence that logically follows the original document.

The establishment of sequential and parallel building blocks and partial orders may require the use of *null* tasks (also known as *dummy* tasks). A null task does not have a realization. Null tasks can be employed to modify a process to obtain structural property (e.g., well-handled and sound) or to make possible the modeling of specific business process procedures.

For our running example, an interview has led to the identification of the following block structures and partial orders:

a) the task *Notify* is the last task to be executed in set $s_1$;
b) the tasks in set $s_2$ (*Book Flight* and *Book Hotel*) can be scheduled in parallel;
c) the tasks in set $s_3$ are scheduled sequentially and the task *Sign* is scheduled before the task *Notify Manager*;
d) since the set $s_4$ has only one task, no partial order needs to be defined;
e) the tasks in set $s_5$ are scheduled sequentially; the task *Reservation User* is scheduled before the task *Send Ticket User*; and
f) since the set $s_6$ has only one task, no partial order needs to be defined.

Figure 9 illustrates the result of applying the three activities of this step to the scheduling table of Figure 8.

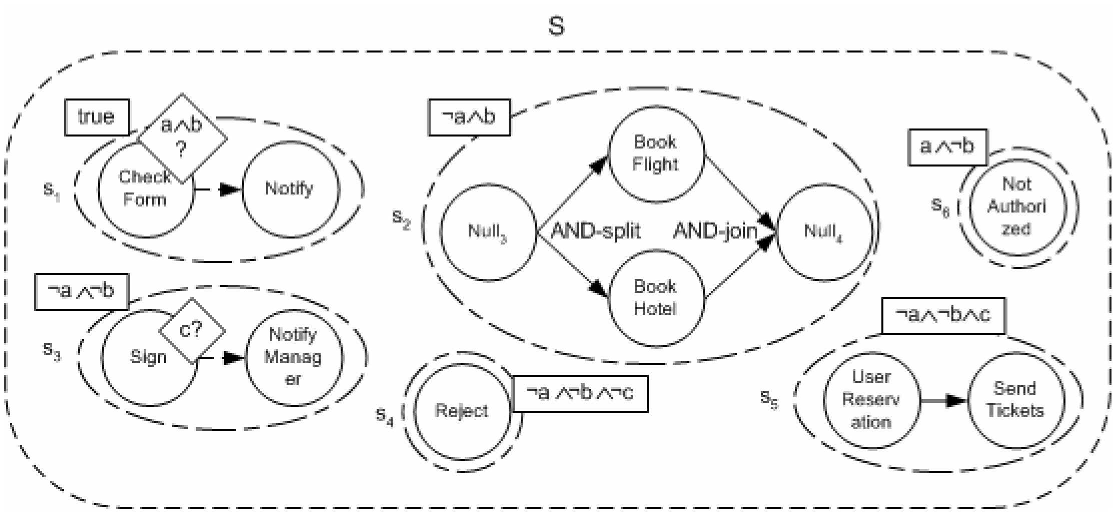


**Figure 9. Parallel and sequential block structures and partial orders for the sets $s_i$**

In each set, two types of transition can exist: permanent transitions (graphically represented with a solid line) and potential transitions (graphically represented with a dashed line)

**Definition 1. Permanent transition.** A permanent transition defines an order between the executions of two tasks. Such a transition is formally defined as,

$$t_b \rightarrow t_e, \text{ where } t_b, t_e \in s_i \text{ and } s_i \in S$$

The semantics of this type of transition is the same as the one of traditional workflow transitions. A permanent transition indicates that $t_e$ is executed immediately after $t_b$.

**Definition 2. Potential transition.** A potential transition defines an order between the executions of two tasks. Such a transition is formally defined as,

$$t_b \succ t_e, \text{ where } t_b, t_e \in s_i \text{ and } s_i \in S$$

The semantics of this type of transition indicate a precedence relationship between two tasks. For example, if $t_b \succ t_e$ then $t_b$ is executed before $t_e$, but $t_e$ does not need to be executed immediately after $t_b$.

### 3.3.2 Conditional Structures

At this point, we have already identified the sequential and parallel building blocks. The next step is to construct a task scheduling graph based on the scheduling sets. The aim of the graph is to identify the conditional building blocks of a process and determine how they control and organize the scheduling sets previously recognized (i.e. sequential and parallel building blocks). The graph is created based on the following rules and assumptions:

**Assumption 1. Business variables must be asserted prior to their use.** For any set $s_i$ to be scheduled, all its tasks must have their business variables asserted. It does not make sense for a task to use a business variable that has never been asserted. Unasserted variables have unknown values (please note that variables with default values are considered to be asserted.)

**Assumption 2. Business variables are only asserted once.** This is a fair assumption since it is often the case, in real world processes, that variables are only asserted once and their value remains unchanged until completion of the process. For example, let us assume that a human task of an administrative business process requests the intervention of an employee to fill in a form with his personal information. The requested information includes his job position title. Let us assume that the job position is a business variable that affects the process's control flow. In this scenario, the job position variable is asserted only once and thereafter it will not be changed again until termination of the process.

The first assumption allows us to determine the tasks where conditional branches to a set $s_i$ may exist. Since all the variables of set $s_i$'s scheduling function need to be asserted for a proper scheduling, a conditional branch can only be attached to a task were all the variables have already been set.

We also define rules, presented below, to impose constraints on how conditional building blocks can organize the scheduling sets.

**Definition 3. Business variables of a set $s_i$.** The set of business variables of the scheduling function associated with set $s_i$ is represented with $s_i^f$, where $f$ is the scheduling function. Graphically, the scheduling function *f* of a set $s_i$ is represented inside a rectangular shape (see Figure 9).

**Rule 1 (Conflicting scheduling sets).** If $s_a^f$ has a scheduling function *f*, the scheduling set $s_b^g$ has scheduling function *g*, and $f \wedge g$ is a contradiction, then a transition between a task of set $s_a$ and a task of set $s_b$ does not make sense (in logic, a contradiction is a Boolean expression or proposition that is always false.)

*Proof.* Let us assume that a transition from a task in $s_a$ to a task in $s_b$ exists. If *f* is true, the set $s_a$ and its tasks are scheduled, but $s_b$ and its tasks can never be scheduled since $f \wedge g$ is a contradiction, i.e. *g* must be false. This holds since business variables can only be asserted once (assumption n° 2). If *f* is false, then obviously $s_b$ and its tasks cannot be executed, since $s_b$ can only be executed if and only if $s_a$ has been previously executed.

**Definition 4.** A scheduling function *f* is a subfunction of a scheduling function *g*, if $f \wedge g$ logically implies *g*. For example, $f = a \wedge \neg b$ is a subfunction of $g_1 = a \wedge \neg b \wedge d$ and of $g_2 = d \wedge a \wedge \neg b$ because $a \wedge \neg b \wedge a \wedge \neg b \wedge d$ logically implies $a \wedge \neg b \wedge d$, and $a \wedge \neg b \wedge d \wedge a \wedge \neg b$ logically implies $d \wedge a \wedge \neg b$.

**Observation.** The transitions between scheduling sets can be viewed as a tree structure. For example, in Figure 9, making $s_1$ the root node of a tree, we can identify 3 out going branches: $s_1$ to $s_2$, $s_1$ to $s_3$, and $s_1$ to $s_6$. Node $s_3$ has 2 outgoing branches: $s_3$ to $s_4$, and $s_3$ to $s_5$. This can be verified in Figure 10.

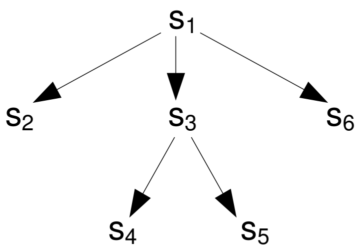


**Figure 10. Tree structure of transitions between scheduling sets**

**Rule 2 (Dependency of scheduling sets).** If $s_a^f$ has a scheduling function *f*, set $s_b^g$ has a scheduling function *g*, the scheduling function *f* is a subfunction of the scheduling function *g*, and one of the tasks in set $s_a$ asserts the variables present in *g* but not present in *f*, then $s_b$ is a node of the sub-tree having root node $s_a$.

*Proof.* If function *f* is a subfunction of function *g*, then a path $p_1$ exists in the tree between the $s_a$ and $s_b$ since if the tasks in $s_a$ are scheduled then the tasks of $s_b$ may also be scheduled. Moreover, a path $p_2$ also exists from the set $s_s$, which sets the variables present in *g* but not present in *f*, and the set $s_b$.

Based on the assumptions and rules presented, we introduce the Conditional Block Identification (CBI) algorithm to assist process analysts and designers in identifying conditional building blocks. Before presenting the algorithm, let us define the following elements,

**Definition 5. Asserted business variables of a task $t_n$.** The asserted business variables of a task $t_n$ is represented with $t_n^v$. For example, if task $t_n$ asserts variable $a$ and $b$ then $t_n^v$ = {a, b}. Graphically, the business variables that a task asserts are the variables of the Boolean function inside a diamond shape (see Figure 9 for an example).

The CBI algorithm, which is described bellow, can be viewed as a methodology describing an iterative process, with human involvement, to structure scheduling sets si into a process graph. Please note that the symbol ≡ is to be read 'takes the value of'.

**CBI Algorithm**

Place the set $s_q \in S$ with $s_q^f = \{\text{'true'}\}$ in the open set $\delta$,

$$\delta = \{ s_q \mid s_q^f = \{\text{'true'}\}, s_q \in S \}.$$

(*) Get a set $s_i$ from the open set $\delta$ and update the open set,

$$\delta \equiv \delta - s_i , s_i \in \delta$$

Propagate the business variables of all the tasks in set $s_i$, i.e. if a task $t_b$ asserts a set of business variables $t_b^v$ then add the set $t_b^v$ to all the reachable tasks ($t_b \rhd t_r$ represents that $t_r$ is reachable from $t_b$) from $t_b$ in $s_i$.

$$\forall t_b, t_r \in s_i, t_b \rhd t_r, t_r^v = t_r^v \cup t_b^v .$$

For each tasks $t_j$ in $s_i$,

Let the set $\zeta$ contain the sets $s_m$ in $S$ such that all the business variables in $s_m^f$ are in $t_j^v$,

$$\zeta = \{ s_m \in S \mid s_m^f \subseteq t_j^v \}.$$

Partition set $\zeta$ such that each partition $\zeta_p$ contains sets $s_m$ with the same set of business variables,

$$\zeta_p = \{s_{m'}, s_{m''}, s_{m'''}, \ldots\}, s_{m'}^{f'} = s_{m''}^{f''} = s_{m'''}^{f'''} = \ldots$$

If the designer decides to do so, allow him to create new permanent transitions from $t_j$ to $t_{pm}$ (i.e., $t_j \longrightarrow t_{pm}$) where $t_{pm}$ is the first task of $s_m \in \zeta_p$.

$$t_j \longrightarrow t_{pm}, \text{ where } t_{pm} \text{ is the first task of } s_m \in \zeta_p$$

If a set $s_m \in \zeta_p$ has a potential transition, place $s_m$ in set $\xi$,

$$\xi = \{ s_m \mid \exists t_b, t_e \in s_m, s_m \in \zeta_p, t_b \succ t_e \}$$

Add the sets $s_m \in \xi$ to the open set $\delta$,

$$\delta \equiv \delta \cup \xi \, .$$

End for each

If $\delta \neq \emptyset$, repeat the process starting at step (*).

A possible scenario for a tool implementing the CBI algorithm is as follows. A business analyst starts a process design tool and sets up all the information that has been described and required up to this point by the Poseidon framework. The tool automatically selects a set $s_i$ (one with a label equal to 'true') from $S$, placing it on the drawing surface. The tasks present in the canvas (from set $s_i$) that may contain *or*-splits are highlighted. The business analyst selects one of the highlighted tasks and, automatically, the tool displays the scheduling sets $s_j$ of tasks for which a transition may exist from the selected task. The analyst can select one or more sets. Automatically an *or*-split structure is created and associated to the highlighted task and the selected sets $s_j$ are placed on the canvas and a transition is drawn. This procedure is repeated until no sets remain in $S$. In our example, the application of the procedure described gives the graph illustrated in Figure 11.

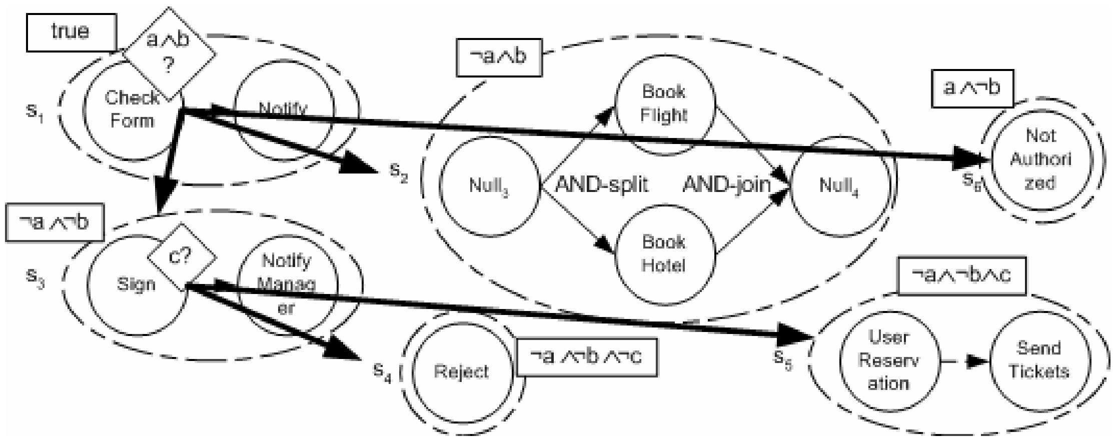


**Figure 11. Task scheduling graph**

Once the dependencies betweens the sets $s_i$ have been established, the structure from Figure 11 can be redrawn to resemble more closely a process; see Figure 12.a). Nevertheless, several process elements are missing. It is apparent in our example that the process does not include any joins matching the *or-splits* and that the process has several ending points.

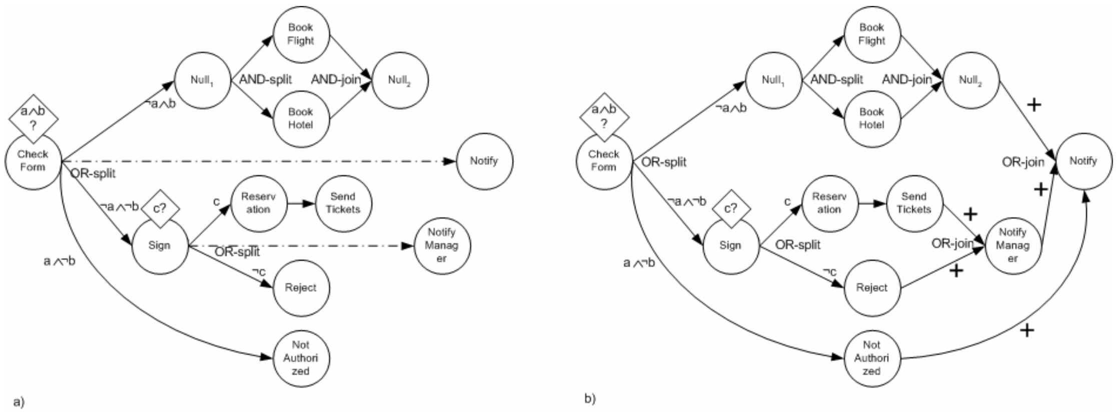


**Figure 12. a) Redrawing the task scheduling graph, b) matching *or-splits* with *or-joins***

Both problems can be solved by matching *or-splits* with *or-joins*. Aalst (Aalst 2000) has pointed out the importance of balancing *or/and-splits* and *or/and-joins* to obtain what is called a 'good' process. For example, two conditional flows created via an *or-split*, should not be synchronized by an *and-join*, but an *or-join* should be used instead. Matching *or/and-splits* may require the use of *null* tasks. Figure 12.b) gives an example of how every *or-split* should be complemented by an *or-join* using a null task.

## 3.4 Cleaning and Implementing

In the last phase, we cleanup of any dummy (null) tasks and, if necessary, the process may be slightly restructured or modified for reasons of clarity. The process design is ready to be implemented. The WIDE methodology (Casati, Fugini et al. 2002) can be used to this end. The method proposed supports the workflow design from the initial analysis phases to its implementation on specific workflow management systems.

The methodology covers the business process pre-analysis, workflow analysis, workflow and external application design, and mapping to workflow implementation phases. The design and mapping to workflow implementation phases are shown in Figure 12.

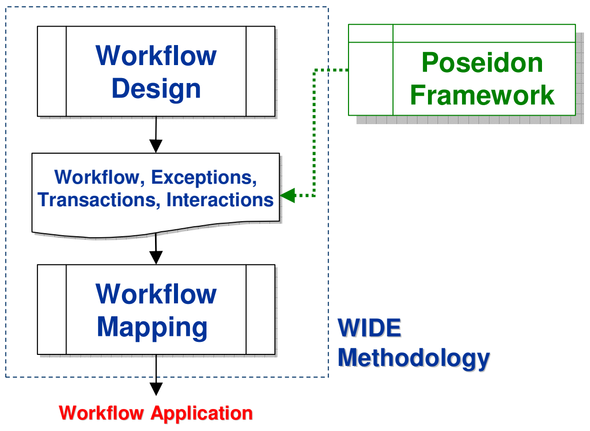


**Figure 13. Integration of Poseidon framework and WIDE methodology**

WIDE methodology provides a set of essential guidelines to design workflows, specify exceptions, transactions, interactions, etc., and map a workflow model into a workflow management system. The workflow design phase of WIDE can be replaced, or complemented, with the Poseidon framework. Compared to WIDE workflow design phase, the Poseidon approach is more powerful since it provides the semi-automatic design of workflows (or processes). Since Poseidon assists the design of processes, it is more suitable to help designers and business process analysts in their tasks. After using the Poseidon framework to design a process, the WIDE methodology can be used to interface processes with existing information systems and external applications, and map the process model, the exceptions, the transactions, and the interactions into a workflow management system.

### 3.4.1 Design phase

In the design phase, the methodology provides concepts that allow the mapping of the resulting workflows onto several different workflow management systems, taking into account the different features of the target system. At the end of the design phase the following results are obtained: workflow schemas, the specification of exceptions and transactions, and specification of interactions with external applications.

*Workflow schemas*. The initial workflow schemas are decomposed into tasks and sub-processes. If a decomposition of the activity of a company into processes already exists, a mapping can be provided from this documentation to a decomposition for the workflow.

*Exceptions*. An exception refers to facts, situations, or abnormal events not modeled by the underlying workflow management system, deviations between what we plan and what actually happens (Luo 2000). Exceptions are low-probability events that are unexpected, nonrepetitive, and infrequent (Strong and Miller 1995). Such exceptional situations may be anticipated and inserted in the workflow schema specification, at design time or later. One of the points of interest in this context is the ability of inserting handlers of exceptions which typically occur given a workflow schema.

*Transactions.* The design of a transactional structure adds semantics to a workflow describing how an instance behaves with respect to atomicity, consistency, isolation, and durability. Transaction design includes the design of compensating transactions needed to rollback completed business transactions.

*Interactions with external applications*. The design of the interaction between the workflow and external information systems is a critical aspect of workflow development. The main issue in the analysis of external information systems and external applications is the specification of the interactions between the external systems and the workflow. A second issue is the management of external information systems data and workflow data which need to be exchanged and may be in different formats, since external information systems data are usually more complex than workflow data.

### 3.4.2 Mapping Phase

The mapping phase maps the workflow model, the exceptions, the transactions, and the interactions into workflow products or applications.

*Mapping the workflow model*. The large majority of workflow systems support basic constructs for defining a workflow, such as and-split, or-split, and-join, or-join, and allows conditions based on workflow data to be associated to paths in order to define when a given path is enabled. These elements are mapped to specific components of the workflow management systems.

*Mapping exceptions*. Exceptions are mapped into the workflow system. Exceptions handling of asynchronous events, temporal events related to task deadlines, based on the state of the task or case, and quality of service thresholds are considered.

*Mapping transactions*. Atomic tasks are the smallest parts into which a process is broken down. Every task is atomic and performed in isolation. These tasks are mapped one to one onto transactions. In case part of the workflow execution needs to be rolled back, those basic tasks need to be undone using compensation techniques.

*Mapping interactions with external applications*. Most workflow management systems include the notion of connector. A connector is a piece of software that allows an easy communication between tasks and external applications. The mapping consists in configuring properly the connectors to interact with external applications. Usually, workflow systems make available to the designer several types of connectors. For example, to access databases, flat files, spreadsheets, Web sites, ERP systems, etc. When specific connectors are need, but are not supplied by the workflow systems, it is possible to developed them using dedicated development tools.

## 4 Framework Evaluation and Future Work

### 4.1 Framework Evaluation

Since we recognized that one-size-fits-all design methods cannot meet the needs of diverse organizations, we present an evaluation of the Poseidon framework. The evaluation answers and discusses the set of requirements presented in section 2.1.

**Simplicity and ease of use.** During the design of processes at Phantom Works – the R&D unit of The Boeing Company, we have tried several methods to capture knowledge from non-technical persons. The methods ranged for the textual description of processes, activities, control-flow, and business variables to the formal representation of processes using graph-based visual notations. The first method revealed to be too descriptive and was not well understood by managers and staff members of the organization. The second method was to complex for non-technical persons. After trying to use several variations of these two methods, we reached the conclusion that the use of a table interrelating business variables and activities (i.e., tasks) was a better approach. CEOs, managers, staff, and non-technical persons in general demonstrated to understand and feel comfortable with the concept of business case table due to its simplicity and ease of use.

**Business process size.** The framework was successfully applied to small and medium size processes. We also feel that the framework is suitable to support the design of large processes. Unfortunately, we did not have an opportunity to model processes of this size.

**Business process structure.** The framework has been used at Phantom Works to develop administrative processes. One interesting administrative process that has been developed was the Travel Request Authorization (Cardoso and Bussler 1999). This process can be seen as a reference model for travel requests. The process created followed all the initial requirements. Furthermore, the framework automatically optimized the process, since Poseidon found that a set of tasks that were carried out sequentially in the original process could be executed in parallel.

**Degree of automation.** The framework has two main phases: the participative and the analytical phase. The participative phase seeks to develop a process design within the settings of workshops, where group studies composed of CEOs, managers, and staff work together. The analytical phase seeks at using automated techniques to model and derive the process design. In Poseidon, the results of participative phase are automatically moved to the analytical phase. The analytical phase is fully automated and its technical details are hidden from end users.

## 4.2 Future Work

### 4.2.1 Quality of Service

One important requirement of business processes is the management of Quality of Service (QoS). Organizations operating in modern markets, such as e-commerce, require QoS management. This management allows organizations to translate their vision into their business processes more efficiently, since workflow can be designed according to QoS metrics. An appropriate control of quality leads to the creation of quality products and services; these, in turn, fulfill customer expectations and achieve customer satisfaction.

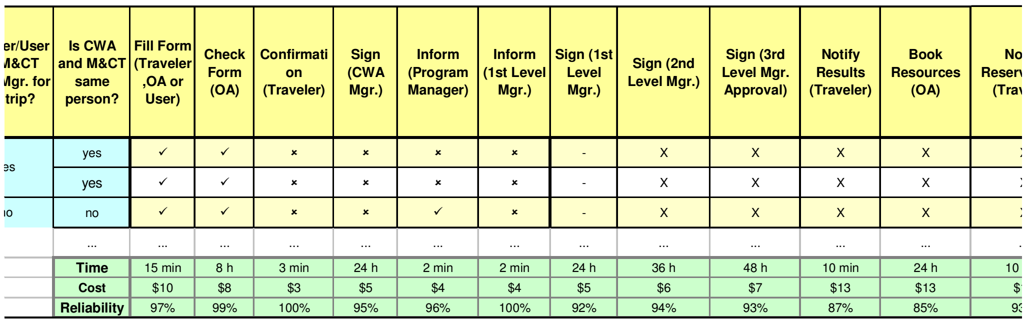
As future work, the QoS model and algorithm described in (Cardoso, Miller et al. 2004) can be integrated with the Poseidon framework. The idea is to create a QoS model and QoS estimates for each task/activity during the participative phase, i.e. during the construction of a business case table. The estimates characterize the quality of service that a task will exhibit at runtime. This information is then used in the analytical phase to automatically compute the the QoS of the overall business process. For e-commerce processes (and other type of processes) it is important to know the QoS an application will exhibit before making the service available to its customers.

Quality of service can be characterized according to various dimensions. In our framework, we have used a QoS model (Cardoso, Miller et al. 2004) composed of the following dimensions: *time*, *cost*, and *reliability*.

- *Time***.** Time is a common and universal measure of performance. The philosophy behind a time-based strategy usually demands that businesses deliver the most value as rapidly as possible.
- *Cost***.** Task cost represents the cost associated with the execution of tasks. The cost of executing a single task includes the cost of using equipment, the cost of human involvement, and any supplies and commodities needed to complete the task.
- *Reliability***.** To describe task reliability we follow a discrete-time modeling approach. The stable reliability model used relates the number of times a task was successfully executed and the number of times the task was executed (Nelson 1973).

Table 1 shows a partial business case table with QoS information.

**Table 1.** Example of a business case table with QoS information



| er/User M&CT Mgr. for trip? | Is CWA and M&CT same person? | Fill Form (Traveler ,OA or User) | Check Form (OA) | Confirmati on (Traveler) | Sign (CWA Mgr.) | Inform (Program Manager) | Inform (1st Level Mgr.) | Sign (1st Level Mgr.) | Sign (2nd Level Mgr.) | Sign (3rd Level Mgr. Approval) | Notify Results (Traveler) | Book Resources (OA) | No Reserv (Trav |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| es | yes | ✓ | ✓ | × | × | × | × | - | X | X | X | X | X |
| | yes | ✓ | ✓ | × | × | × | × | - | X | X | X | X | X |
| o | no | ✓ | ✓ | × | × | ✓ | × | - | X | X | X | X | X |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | . |
| | **Time** | 15 min | 8 h | 3 min | 24 h | 2 min | 2 min | 24 h | 36 h | 48 h | 10 min | 24 h | 10 |
| | **Cost** | $10 | $8 | $3 | $5 | $4 | $4 | $5 | $6 | $7 | $13 | $13 | $ |
| | **Reliability** | 97% | 99% | 100% | 95% | 96% | 100% | 92% | 94% | 93% | 87% | 85% | 93 |

## 4.2.2 Process Mining

With the mergence of Web services, workflow management systems become essential to support, manage, and enact Web processes, both between enterprises and within the enterprise (Sheth, Aalst et al. 1999). Workflow systems are capable of both generating and collecting considerable amounts of data describing the execution of business processes, such as Web processes. This data is stored in a process log, an independent component that records the events for all of the processes being executed by the enactment service.

Process logs are vast data archives that are seldom visited. Yet, the data generated from the execution of processes are rich with hidden information that can be used for making intelligent business decisions. Data analysis may uncover important process patterns, contributing greatly to business strategies.

An important and useful knowledge to discover and extract from process logs is the set of implicit rules that govern the executions of tasks during the execution of a process. This type of mining is called process mining or workflow mining (Aalst, Dongen et al. 2003).

Process mining can automatically determine the causal relationship between sequential tasks. The Poseidon framework is unable, in some cases, to determine if a task $t_a$ should be executed before, or after, a task $t_b$. This happen if there are no data dependencies between the two tasks, i.e. from a data flow point of view, task $t_a$ can be executed after task $t_b$ or the other way around. But in most processes there is a causal relationship between sequential tasks. Let us consider the following very simple and naïve example, at some point in time, a process executes the tasks $t_{show\_page\_1}$ and $t_{show\_page\_2}$. Both tasks show a textual page to the user. The first task shows the first page of a contract and, naturally, the second task shows the second page of a contract. From the data point of view, task $t_{show\_page\_2}$ can be executed before $t_{show\_page\_1}$, but from the logical point of view this sequence is not sound.

Therefore, when designing a process using the Poseidon framework, process mining can be used effectively to find causal relationships between tasks that have been previously executed in the context of other processes. In our simple example, process mining techniques can found that, in the context of existing processes, $t_{show\_page\_1}$ has always been executed before $t_{show\_page\_2}$ and, therefore, a causal relationship exists.

## 5 Conclusions

New economies and global markets with their accompanying intense competition have generated the need for organizations to adopt new working models. Most companies have recognized the need for business process management to increase efficiency and to survive intense competition. Process design and management are key issues in emergent architectures, such as e-Commerce. Organizations need well-specified and documented methods to guide the design of (Web) process applications.

Although major research has been carried out to enhance workflow systems, the work on process and workflow application development lifecycles and methodologies is practically inexistent. The development of adequate frameworks is of importance to guarantee that processes are constructed according to initial specifications. Furthermore, it would be advantageous for process analysts to have tools to support – automatically or semi-automatically – the design of applications.

Unfortunately, it is recognized that despite the diffusion of workflow systems, methodologies and frameworks to support the development of process applications are still missing. In this paper, we describe a framework to assist process analysts during their interviews with administrative staff, managers, and employees in general to design processes. The framework includes a set of procedures that guide the process analyst during his interviews and supply (automatic) methods to ease the design process. As a result, processes can be developed and implemented more rapidly and accurately.

The core of the framework presented has been employed successfully to design medium size processes at The Boeing Company (Seattle, WA, USA). Currently, the framework is being used by M.Sc. students at the University of Madeira (Portugal) to design business processes. We believe that the framework is also appropriate to design large processes and that it represents a good step towards the modeling of business processes.

## 6 References

Aalst, W. M. P. v. d. (1998). "The Application of Petri Nets to Workflow Management." The Journal of Circuits, Systems and Computers **8**(1): 21-66.

Aalst, W. M. P. v. d. (2000). Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. Business Process Management: Models, Techniques, and Empirical Studies. W. M. P. v. d. Aalst, J. Desel and A. Oberweis. Berlin, Springer-Verlag. **1806:** 161-183.

Aalst, W. M. P. v. d., A. P. Barros, et al. (2000). Advanced Workflow Patterns. Seventh IFCIS International Conference on Cooperative Information Systems.

Aalst, W. M. P. v. d., B. F. v. Dongen, et al. (2003). "Workflow Mining: A Survey of Issues and Approaches." Data & Knowledge Engineering (Elsevier) **47**(2): 237-267.

Alonso, G., C. Mohan, et al. (1994). Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. IFIP WG8.1 Working Conference on Information Systems for Decentralized Organizations, Trondheim, Norway.

Cardoso, J., R. P. Bostrom, et al. (2004). "Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications." Information Technology and Management Journal. Special issue on Workflow and E-Business (Kluwer Academic Publishers) **5**(3-4): 319-338.

Cardoso, J. and C. Bussler (1999). MARATHON - Workflow Management System, The Boeing Company. **2004**.

Cardoso, J., J. Miller, et al. (2004). "Modeling Quality of Service for workflows and web service processes." Web Semantics: Science, Services and Agents on the World Wide Web Journal **1**(3): 281-308.

Cardoso, J. and A. Sheth (2003). "Semantic e-Workflow Composition." Journal of Intelligent Information Systems (JIIS). **21**(3): 191-225.

Cardoso, J., A. Sheth, et al. (2002). Workflow Quality of Service. International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02), Valencia, Spain, Kluwer Publishers.

Cardoso, J. and J. C. Teixeira (1998). Workflow Management Systems: A Prototype for the University of Coimbra. 5th International Conference on Concurrent Engineering, Tokyo, Japan.

Casati, F., M. Fugini, et al. (2002). "WIRES: a Methodology for Designing Workflow Applications." Requirements Engineering Journal **7**(2): 73-106.

Ceri, S., P. Grefen, et al. (1997). WIDE-A Distributed Architecture for Workflow Management. Proceedings of the 7th International Workshop on Research Issues in Data Engineering, Birmingham, UK.

Chandrasekaran, S., G. Silver, et al. (2002). Service Technologies and their Synergy with Simulation. Proceedings of the 2002 Winter Simulation Conference (WSC'02), San Diego, California.

Drechsler, R. and D. Sieling (2001). "Special section on BDD: Binary decision diagrams in theory and practice." International Journal on Software Tools for Technology Transfer **3**(2): 112-136.

Eder, J., E. Panagos, et al. (1999). Time Management in Workflow Systems. BIS'99 3rd International Conference on Business Information Systems, Poznan, Poland, Springer Verlag.

Fensel, D. and C. Bussler (2002). The Web Service Modeling Framework, Vrije Universiteit Amsterdam (VU) and Oracle Corporation.

Jennings, N. R., P. Faratin, et al. (1996). ADEPT: Managing Business Processes using Intelligent Agents. Proc. BCS Expert Systems 96 Conference, Cambridge, UK.

Karnaugh, M. (1953). "The Map Method for Synthesis of Combinational Logic Circuits." Transaction IEEE **72**(9): 593-599.

Kochut, K. J., A. P. Sheth, et al. (1999). ORBWork: A CORBA-Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR. Athens, GA, Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia.

Luo, Z. (2000). Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes. Department of Computer Science. Athens, GA, University of Georgia**:** 171.

McCluskey, E. J. (1956). "Minimization of Boolean functions." Bell System Technical Journal **35**(5): 1417-1444.

McCready, S. (1992). There is more than one kind of workflow software. Computerworld. **November 2:** 86-90.

Miller, J. A., D. Palaniswami, et al. (1998). "WebWork: METEOR2's Web-based Workflow Management System." Journal of Intelligence Information Management Systems: Integrating Artificial Intelligence and Database Technologies (JIIS) **10**(2): 185-215.

Nelson, E. C. (1973). A Statistical Basis for Software Reliability, TRW Software Series.

Ould, M. A. (1995). Business Processes: Modelling and analysis for re-engineering and improvement. Chichester, England, John Wiley & Sons.

Reijers, H. A. (2003). Design and Control of Workflow Processes: Business Process Management for the Service Industry. Berlin, Springer-Verlag.

Reijers, H. A., S. Limam, et al. (2003). "Product-based Workflow Design." Journal of Management Information systems **20**(1): 229-262.

Sadiq, S., O. Marjanovic, et al. (2000). "Managing Change and Time in Dynamic Workflow Processes." The International Journal of Cooperative Information Systems **9**(1, 2): 93-116.

Sadiq, W. and M. E. Orlowska (1999). On Capturing Process Requirements of Workflow Based Business Information Systems. Proceedings of the 3rd International Conference on Business Information Systems (BIS '99), Poznan, Poland, Springer-Verlag.

Shegalov, G., M. Gillmann, et al. (2001). "XML-enabled workflow management for e-services across heterogeneous platforms." The VLDB Journal **10**(1): 91-103.

Sheth, A., D. Georgakopoulos, et al. (1996). Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Athens, GA, Deptartment of Computer Science, University of Georgia.

Sheth, A. P., W. v. d. Aalst, et al. (1999). "Processes Driving the Networked Economy." IEEE Concurrency **7**(3): 18-31.

Sommerville, I. (2000). Software Engineering, Addison-Wesley Pub Co.

Son, J. H., J. H. Kim, et al. (2001). "Deadline Allocation in a Time-Constrained Workflow." International Journal of Cooperative Information Systems (IJCIS) **10**(4): 509-530.

Strong, D. and S. Miller (1995). "Exceptions and exception handling in computerized information processes." ACM Transactions on Information Systems **13**(2): 206-233.