A CASE STUDY IN WORKFLOW MODELLING USING CONTROL-FLOW PATTERNS

Y. Callero[(a)], I. Castilla[(b)], R.M. Aguilar[(c)]

[(a)(b)(c)]Department of Systems Engineering and Automation, and Computer Architecture.
Universidad de La Laguna. Spain
[(a)]ycallero@isaatc.ull.es, [(b)]ivan@isaatc.ull.es, [(c)]raguilar@ull.es

ABSTRACT
Business Process Reengineering is a field where state-of-the-art powerful tools are required to obtain valid and profitable results. In this sense, Business Process Simulation is becoming one well-known instrument to analyze and return valuable solutions, although a potential user has to face the lack of a standard modelling approach. This paper introduces the most important difficulties that arise when dealing with the practical implementation of one of these approaches, the Synchronizing Workflow Models; and presents an example illustrating how these problems have been solved with the Discrete Event Simulation library, SIGHOS.

Keywords: Business Process Management, Discrete Event Simulation, Workflow Patterns, Simulation Tool, Synchronized Workflow Models, Business Process Reengineering.

1. INTRODUCTION

The workflow concept is inherently related to the notion of process, as it was originally conceived since industrialization in manufacturing and the office (Georgakopoulos, Hornick and Sheth 1995). "Processes" separate work activities into well-defined tasks, roles, rules and procedures in order to increase efficiency.

Over time, this way of working leads to a fragmentation of the business that negatively impacts on cost and on the motivation of the personnel.

In this sense, Business Process Reengineering (BPR) makes its appearance. As stated by Muthu, Whitman and Cheraghi (2006), reengineering *is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance such as cost, quality, service and speed*. A Business Process (BP) *is a market-centred description of an organization's activities, implemented as an information process and/or a material process* (Medina-Mora, Winograd, Flores and Flores 1992).

Business Process Simulation (BPS) is an important tool within BPR. As described by Wynn, Dumas, Fidge, Hofstede and Aalst (2008), BPS is focused on the achievement of two main goals:

1. the analysis of the behaviour of a process, by means of the development of accurate simulation models;
2. the understanding of the effects of running that process through the performance of simulation experiments.

Three components describe a BPS (Wynn, Dumas, Fidge, Hofstede and Aalst 2008):

1. basic model building blocks, such as entities, resources, activities, and connectors;
2. activity modelling constructs, such as split, join, branch and assemble;
3. and advanced modelling functions, such as attributes, expressions and resource schedules.

Russell, Hofstede and Mulyar (2006) delineate the fundamental requirements that arise during business process modelling on a recurring basis and describe them in an imperative way. A pattern-based approach is used to describe these requirements. Such approach offers both a language- and technology- independent means of expression in a form that is sufficiently generic to allow for a wide variety of applications.

However, there is no generally accepted modelling technique for implementing workflow patterns as part of a simulation tool. Different notations, such as EPCs (Event-Driven Process Chain) or BPMN (Business Process Management Notation) have been posed that try to overcome the difficulties arisen when implementing such models.

Precisely, the EPC notation has served as a basis for adapting SIGHOS, a discrete event simulation library developed by the Simulation Group from the University of La Laguna, to support the use of workflow patterns in the simulation of business processes. The rest of this paper is structured as follows. The first section defines the main concepts related to

workflows modelling. Next, the methodology used to model the functionalities proposed by the control-flow patterns based on Synchronized Workflow Models is detailed. Once the strategy is explained, a practical approach is taken to present the structures designed in SIGHOS for modelling said patterns. A modelling example is presented in the next section. The last section contains a summary of the conclusions drawn from our research.

2. WORKFLOW BASICS

Having set the definition of business process, the Workflow Management Coalition (WfMC) defines "workflow" as *the partial or total automation of a business process during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural* rules. (Workflow Management Coalition 1999)

There is little consensus in the workflow specification due to the lack of universal concepts for modelling business processes (Aalst, Hofstede, Kiepuszewski and Barros 2003).

One of the most widespread terminologies used to describe the concepts and general structure of a workflow is the WfMC terminology (Workflow Management Coalition 1999). The concept of "process instance" is especially important to understand the behaviour of SIGHOS models. The WfMC terminology describes a process instance as a single enactment of a process, or activity within a process, including its associated data. Each process instance is executed on a separate thread. However, if a process includes parallel activities, the corresponding process instance would include multiple concurrent threads of execution.

2.1. Control flow perspective

Among the different perspectives (control-flow, data, resource and operational) to the workflow specifications stated in (Aalst, Hofstede, Kiepuszewski and Barros 2003), this research work focuses on the control-flow perspective.

Russell, Hofstede and Mulyar (2006) studied countless practical cases involving real companies, and proposed 43 patterns which define the requirements for modelling the different scenarios defined within the control flow perspective. Many workflow definition languages, modelling tools and workflow engines have resulted from this practical approach like: BPEL (Hinz, Schmidt and Stahl 2005), XPDL (Aalst 2003), ARIS (Davis 2001), Gridant (Amin, Hategan, von Laszewski, Zaluzec, Hampton and Rossi 2004) and WW-FLOW (Kim, Kang, Kim, Bae and Ju 2000).

Kiepuszewski, Hofstede and Aalst (2003) expose the basic control flow constructs, common to most of these approaches:

- AND-Split is a point within the workflow where a single thread of control splits into two or more threads which are executed in parallel.
- AND-Join is a point in the workflow where two or more parallel executing tasks converge into a single common thread of control.
- OR-Split is a point within the workflow where a single thread of control decides on which branch to take when encountered with multiple alternative workflow branches.
- OR-Join is a point within the workflow where two or more alternative task workflow branches re-converge to a single common task as the next step within the workflow.

2.2. Theoretical foundations

Whilst the patterns proposed by Russell et al. are convenient for a practical approach, some more robust theoretical foundation is required that characterizes control flow modelling. Therefore, formal tests about expressiveness limits and properties of a model may be performed.

Petri nets have been traditionally employed to specify control flows, since activities can be seen as transitions; and causal dependencies as places, transitions and arcs. Kiepuszewski (2003) uses the properties of the Petri nets to distinguish among four different techniques employed to model control-flow patterns.

First, Kiepuszewski defines the *Standard Workflow Models* as the most "natural" interpretation of the WfMC definitions. Standard Workflow Models have the ability to create multiple concurrent instances of one activity.

Safe Workflow Models, on the contrary, never create multiple concurrent instances of an activity. Consequently, the corresponding Petri net is safe.

Intuitively, a *Structured Workflow Model* is a model where each OR-Split has a corresponding OR-Join and each AND-Split has a corresponding AND-Join, with no arbitrary cycles allowed.

Last, *Synchronizing Workflow Models* appear from a different interpretation of the WfMC definitions of basic control flow constructs. An AND-Join typically follows an AND-Split and can be seen as a construct that synchronizes a number of active threads. To synchronize that kind of construct, commercial

tools generally use a token-based technique. The semantics of Synchronizing Workflow Models can be easily captured by using Coloured Petri nets.

3. SIGHOS AND SYNCHRONIZING WORKFLOW MODELS

3.1. SIGHOS, a discrete event simulation tool

The potential of Simulation as a tool for business process modelling has not been yet recognized by much of the business community (Hlupic and Robinson 1998). However, BPS may be used to achieve a higher level of understanding when studying and analyzing businesses which are inherently complex. Not only this, simulation has several characteristics that make it appealing for business process modelling. Indeed, a process-based world view, as defined from a simulation modelling perspective, can be seen as *a time-ordered sequence of interrelated events which describe the entire experience of an entity as it flows through the system* (Balci 1988). This definition can be easily matched with the flow of entities through business processes.

SIGHOS (SIGHOS project homepage 2010) is a process-oriented discrete event simulation tool which was originally intended to simulate hospital management (Aguilar, Castilla, Muñoz, Martín and Piñeiro 2006). Being Java-based, SIGHOS takes advantage from the known benefits of using Java to implement a discrete event simulation tool (Buss 2002)(L'ecuyer and Buist 2005)(Goes, Pousa, Carvalho, Ramos and Martins 2005). Robustness, portability, and ease of implementation and documentation for programmers are some of the strengths usually associated with Java. Moreover, Java is inherently Internet- and Thread- aware, that is, the language includes in its core definition primitives to deal with network communications and concurrent programming.

SIGHOS evolved from its original conception to a broader range of applications such as call centres (Castilla, Muñoz and Aguilar 2007) by generalizing the structures employed to define processes (Castilla, García and Aguilar 2009). However, these first simple structures were not powerful enough to deal with the multiple complexities that a real business process may pose to a modeller. Thus, taking control-flow patterns as a reference, the business process modelling capacity of this tool was expanded. The difficulties arise when coping with the problems posed, by definition, by synchronizing workflow models.

3.2. Why synchronizing models?

Synchronizing Workflow Models allows the modelling of multiple instances of a process unlike Safe Workflow, Structured Workflow or Standard Workflow models as posited by Kiepuszewski (2003).

A priori, the Synchronizing Workflow Models are best suited to the SIGHOS tool since they solve the deadlock problem and efficiently allow for the problems of modelling multiple-instance patterns and loops to be overcome.

Kiepuszewski advised to use the Token-based strategy in the Synchronizing Workflow Models implementation. The goal of using this strategy is to solve the possible appearance of deadlocks during the runtime. This strategy implies that each node in the model has to propagate one or several tokens indicating the state of the outgoing threads.

3.3. Token-based strategy implemented by SIGHOS

SIGHOS adopts the token-based strategy from two basic nodes: *SingleSuccessor* or *MultiSuccessor*. The former can only be linked to a single node, while the latter may have several linked nodes.

Based on these two kinds of nodes, SIGHOS implements four basic constructs:

- *Initializer node*
- *Finalizer node*
- *Task node*
- *Structured node*

An *Initializer* is a *MultiSuccesor* node that represents a source of new tokens, so that the generation of a token involves the creation of a new thread. *AND-Split*, *OR-Split* and *Conditional* nodes are based on this node. If a true token is received, the node propagates a true token through the active outgoing branches, and a false token through the inactive ones. In case a false token is received, the node propagates a false token through each outgoing branch.

Conditional nodes are a special case of *OR-Split*. Upon receipt of a true token, the node checks each associated condition changing the resulting token value depending on the check result. Upon receipt of a false token, a false token is sent to each outgoing branch.

A *Finalizer* is a *SingleSuccesor* node that represents a sink of tokens. A token removal represents the finalization of thread execution. This node is the basis for AND-Join or OR-Join nodes. The propagation of tokens depends on the criteria present at each node for yielding flow control:

- An AND-Join node generates a new token from the confluence of incoming tokens. This node defines an acceptance value as the total amount of true tokens that it must receive through the incoming branches. When enough tokens are received, a true token is sent to the node's successor and the account of tokens is reset. This type of node has two operating modes: safe and unsafe. Using one or another operating mode affects the way the concurrent receipt of tokens through the same incoming branch is treated. In safe mode, only one token per branch and simulation timestamp is taken into account; the rest are simply discharged. In unsafe mode, all the tokens are considered and taken as valid.
- An OR-Join node simply propagates the same token which arrived to the successor. The only control that is performed affects the concurrent arrival of tokens. At that moment, the control state has to decide if all or only one thread is subsequently sent to the successors. If the criteria for yielding control is not met once every incoming branch token has been received, a false token is propagated.

A Task node is also a SingleSuccesor node and represents the execution of some kind of activity. The activity is executed upon receipt of a true token. Once the execution is complete, a true token is propagated. However, if the execution is cancelled or the node receives a false token - meaning the execution is not to be carried out - a false token is propagated.

Structured nodes are SingleSuccessor nodes which consist of one node that defines the structure's starting point, and another one that defines its end. All kind of complex branches can link such starting and end points. Only upon arrival of a true token, the control is yielded to the sub-flow contained in the structure. Otherwise, the false token is simply propagated to its successors.

3.4. Improving Synchronizing Workflow Models

One of the problems that Kiepuszewski (2003) associates with Synchronizing Workflow Models is the impossibility of handling multiple process instances without running into serious limitations. Such limitations result from his formal definition involving Petri nets. SIGHOS takes advantage from not assuming such a formal base. One goal of the tool is the conduct of efficient simulations from the standpoint of parallelism and concurrence (Castilla, García and Aguilar 2009). That is why the functional core of the tool is optimized to simulate various processes and their multiple instances without limitations. For example, with the token-based strategy, a process could have several active instances executing the same task given enough available resources. The tokens that indicate the validity of each instance are totally independent, which allows the model to be coherent.

Another important problem associated with Synchronizing Workflow Models is the control of different iterations within a loop which is solved by treating each node as a structured one. This leaves the handling of the possible arbitrary cycles that are defined in the model as the only outstanding problem. SIGHOS handles the patterns of arbitrary cycles by implementing a system of environment variables, associated with the model or defined by the user, and an expression set which allows for the conditions associated with these variables to be defined. There is also a set of user events that allows the values of these variables to be modified at different points in the model. These structures allow arbitrary cycles to be modelled, maintaining control over the possible appearance of infinite loops. On the one hand, when a true token enters one of these cycles, the set of conditions associated with this cycle will control the exit from the cycle for that token. On the other hand, when a false token enters one of these cycles, this token propagates, both to the exit branch for a cycle as well as to the branch that generates the loop. This leads to an infinite propagation of the false token, which results in an improper execution of the simulation. This undesirable situation can be solved if each false token keeps track of the nodes it has been to. Should a false token return to a node through which it has already passed, it is assumed to be immersed in a loop and deleted from the simulation since it is no longer producing relevant information.

4. CONTROL-FLOW PATTERNS SUPPORTED BY SIGHOS

This section analyzes which control flow patterns are supported by SIGHOS. Patterns are divided into categories depending on its functionality. These categories are: basic control flow patterns, advanced branching and synchronization patterns, multiple instance patterns, iteration patterns, termination patterns and trigger patterns.

Each category is discussed in a separate subsection, including an explanation about whether each specific workflow control pattern (WCP) is supported by SIGHOS or not.

4.1. Basic Control Flow Patterns

These basic patterns capture elementary aspects of process control.

Table 1. Basic control flow patterns support

Basic Control Flow Patterns	Supported by SIGHOS
WCP1: Sequence	YES
WCP2: Parallel Split	YES
WCP3: Synchronization	YES
WCP4: Exclusive Choice	YES
WCP5: Simple Merge	YES

No further explanation is required since the support of these patterns is trivial.

4.2. Advanced Branching and Synchronization Patterns

These patterns characterise more complex branching and merging concepts which arise in business processes.

Table 2. Advanced branching and synchronization patterns support

Advanced Branching and Synchronization Patterns	Supported by SIGHOS
WCP6: Multi-Choice	YES
WCP7: Structured Synchronizing Merge	YES
WCP8: Multi-Merge	YES
WCP9: Structured Discriminator	YES
WCP28: Blocking Discriminator	YES
WCP29: Cancelling Discriminator	NO
WCP30: Structured Partial Join	YES
WCP31: Blocking Partial Join	YES
WCP32: Cancelling Partial Join	NO
WCP33: Generalised AND-Join	YES
WCP37: WCP: Local Synchronizing Merge	YES
WCP38: General Synchronizing Merge	NO
WCP41: Thread Merge	YES
WCP42: Thread Split	YES

SIGHOS focuses large part of its current operating potential on the ability to represent patterns of this class. Modelling primitives are provided for almost the entire set of patterns, but three. The Cancelling Discriminator, the Cancelling Partial Join and the General Synchronizing Merge are not considered because of their non-local semantics limitations. This problem will be revisited in section 5.

4.3. Multiple Instance Patterns

These patterns describe process models including an activity with multiple active threads of execution.

Table 3. Multiple instance patterns support

Multiple Instance Patterns	Supported by SIGHOS
WCP12: Multiple Instances without Synchronization	YES
WCP13: Multiple Instances with a Priori Design-Time Knowledge	YES
WCP14: Multiple Instances with a Priori Run-Time Knowledge	NO
WCP15: Multiple Instances without a Priori Run-Time Knowledge	NO
WCP34: Static Partial Join for Multiple Instances	YES
WCP35: Cancelling Partial Join for Multiple Instances	NO
WCP36: Dynamic Partial Join for Multiple Instances	NO

SIGHOS offers the possibility of handling multiple instances as long as they are pre-defined in the model. The system is not yet able to deal with instances created dynamically during the simulation. That is why the remaining patterns in this class are not accepted, and why a process for cancelling activities has not been defined.

4.4. State-based Patterns

State-based patterns are more easily accomplished in process languages that support the notion of state. In this context, the state of a process instance is considered to include the broad collection of data associated with current execution, that is, the status of various activities as well as process-relevant working data such as activity and case data elements.

Table 4. State-based patterns support

State-based Patterns	Supported by SIGHOS
WCP16: Deferred Choice	NO
WCP17: Interleaved Parallel Routing	YES
WCP18: Milestone	NO
WCP39: Critical Section	NO
WCP40: Interleaved Routing	YES

Only 2 out of 5 patterns (the interleaved ones) are implemented in SIGHOS. The remaining patterns are beyond the scope of the library.

4.5. Cancellation and Force Completion Patterns

Several of the patterns above have variants that utilize the concept of activity cancellation where enabled or active activity instances are withdrawn. Various forms of exception handling in processes are also based on cancellation concepts.

Table 5. Cancellation and force completion patterns support

Cancellation and Force Completion Patterns	Supported by SIGHOS
WCP19: Cancel Task	NO
WCP20: Cancel Case	NO
WCP25: Cancel Region	NO
WCP26: Cancel Multiple Instance Activity	NO
WCP27: Complete Multiple Instance Activity	NO

SIGHOS does not provide any cancellation method, neither for cases nor for activity executions.

4.6. Iteration Patterns

The following patterns represent repetitive behaviour in a workflow.

Table 6. Iteration patterns support

Iteration Patterns	Supported by SIGHOS
WCP10: Arbitrary Cycles	YES
WCP21: Structured Loop	YES
WCP22: Recursion	NO

SIGHOS allows for the modelling of loops and arbitrary cycles. The recursive case is not considered.

4.7. Termination Patterns

These patterns face completion of workflows.

Table 7. Termination patterns support

Termination Patterns	Supported by SIGHOS
WCP11: Implicit Termination	YES
WCP43: Explicit Termination	NO

Being designed as an event-oriented simulator, SIGHOS directly supports the implicit termination pattern. Explicit termination, understood as a generalization of the cancellation patterns, is discharged due to the limitations in terms of non-local semantics of the library.

4.8. Trigger Patterns

Trigger patterns deal with the external signals that may be required to start certain tasks.

Table 8. Trigger patterns support

Trigger Patterns	Supported by SIGHOS
WCP23: Transient Trigger	NO
WCP24: Persistent Trigger	NO

Triggers also rely on non-local semantics which cannot be solved by the single use of tokens. Thus, their implementation goes beyond the scope of the library.

5. CONTROL-FLOW PATTERNS NOT IMPLEMENTED BY SIGHOS

As shown in the previous section, SIGHOS has been proved to be a valid tool to implement most of the patterns defined by Russell, Hofstede and Mulyar (2006). However, some patterns have not been implemented yet due to different reasons with a remarkable influence of non-local semantics (Aalst, Desel, Eichstätt-ingolstadt, Kindler and Paderborn 2002). Non-local semantics make reference to those situations where a node requires information which is not local to the node (that is, which belongs to a different part of the model) in order to take a decision on the propagation of a process instance. The token strategy introduced in section 3.2 copes with some of the problems derived from such semantics but others remain unsolved.

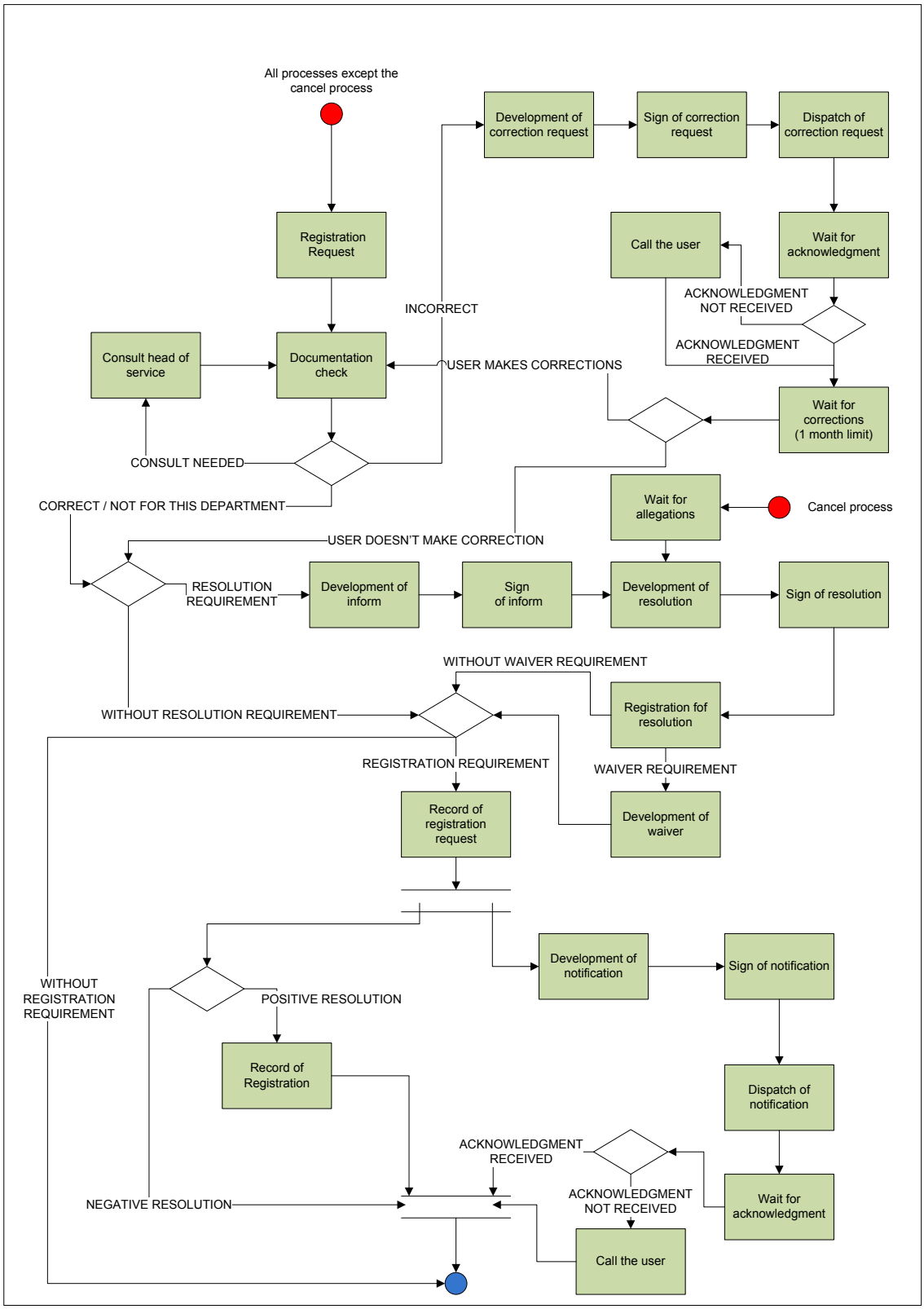

Figure 1. Registry of Associations workflow diagram

Cancellation patterns (WCP19, WCP20, WCP25, WCP26, WCP27, WCP29, WCP32 and WCP35) perfectly illustrate the problems derived from the use of non-local semantics. These patterns require explicitly breaking the execution logic of the simulator. A cancel pattern normally affects a process instance or a set of process instances. However, the process instance that activates the cancellation can be located in a completely different part of the

model. Locating such instances implies complex memory structures and an efficient search algorithm, thus increasing the memory and CPU requirements for the simulator.

6. MODELLING EXAMPLE: REGISTRY OF ASSOCIATIONS

Several civil service processes have been reorganized into electronic processes in Canary Islands government. A specific case study was described in (Callero and Aguilar 2009). Upon the basis of the workflow modelling of this case, a deeper analysis is presented in this section.

6.1. Description of the problem

Spain and the autonomous regions governments share the competences about the control of associations. The autonomous region government takes all responsibility for the establishment of different ways to set up, manage or dissolve associations. The Registry of Associations was defined to manage all of these operations.

Focusing in Canary Islands region, each province (Las Palmas and Santa Cruz de Tenerife) has its own Registry of Associations head office and its own staff. Whereas Las Palmas office has two government employees and two technicians, Santa Cruz office has three government employees and one technician. In addition, there are one head of service and one general director for the entire autonomous region. All the activities of the Registry of Associations are carried out by the described staff.

6.2. Flow modelling

The workflow of the process is shown in Figure 1. The Registry of Associations has nineteen different activities. The activities are modelled using Task Nodes.

The execution of the activities is started by a customer request but the execution flows depends on the process characteristics. Sixteen different processes can be executed in Registry of Associations, thus creating decision points to route the flow depending on the process characteristics: does the subprocess documentation needs to be corrected or consulted? Is the process resolution negative or positive?

Using Conditional nodes, associated to the environment variables, it is possible to emulate the ExclusiveChoice pattern in order to model the decision points.

Notification and registration subprocess at the end of the process are another key point in the flow of the Registry of Associations. This subprocess generates a parallel execution of tasks. One parallel branch is involved in the notification tasks and the other branch is involved in the registration tasks. Both branches are synchronized at the end of the flow.

SIGHOS Structured Nodes give to the modeller the possibility to define Structured Synchronizing Merge pattern behaviour to simulate processes like notification and registration subprocess.

7. CONCLUSIONS

Workflow technology is still under development in its traditional application areas (business process modelling and business process coordination), but also in emergent areas of component frameworks and inter-workflow, business-to-business interaction (Aalst, Hofstede, Kiepuszewski and Barros 2003).

In this paper, a practical example of an application of synchronized models for the modelling of workflow patterns has been presented.

It is clear that the token-based strategy provides an enormous freedom of design, although not all of its potential has been exploited yet. In other words, from the point of view of the developing group, the SIGHOS library can still be significantly improved in the area of modelling workflow patterns.

A modelling example has been also introduced to illustrate the use of SIGHOS and the synchronized models to simulate a real problem. Future work will further develop this model.

ACKNOWLEDGMENTS

This work is being supported by a project (reference DPI2006-01803) from the Ministry of Science and Technology with FEDER funds.

Iván Castilla is being supported by an FPU grant (ref. AP2005-2506) from the Ministerio de Educación y Ciencia of Spain.

Yeray Callero is being supported by a postgraduate grant from CajaCanarias Canary Island bank.

REFERENCES

Aalst, W. M., Desel, J., Eichstätt-ingolstadt, K. U., Kindler, E. and Paderborn, U., 2002. On the semantics of EPCs: A vicious circle. *Proceedings of the EPK 2002: Business Process Management using EPCs*, 71-80.

Aalst, W. M., Hofstede, A., Kiepuszewski, B. and Barros, A., 2003. Workflow Patterns. *Distributed and Parallel Databases*, 14, 5 - 51.

Aalst, W.M., 2003. Patterns and XPDL: A Critical Evaluation of the XML Process Definition Language, *QUT Technical report*, Brisbane.

Aguilar, R. M., Castilla, I., Muñoz, R., Martín, C. A. and Piñeiro, J. D., 2006. Verification and validation in discrete event simulation: A case study in hospital management. International Mediterranean Modeling Multiconference 2006 proceedings.

Amin, K., Hategan, M., von Laszewski, G., Zaluzec, N., Hampton, S. and Rossi, A., 2004. Gridant: A client-controllable grid workflow system. *37th Hawai'i International Conference on System Science*, 5–8.

Balci, O., 1988. The implementation of four conceptual frameworks for simulation modeling in high-level languages. *Proceedings of the 20th conference on Winter simulation - WSC '88*, 287-295.

Buss, A., 2002. Component based simulation modeling with simkit. *Proceedings of the Winter Simulation Conference*, 243-249.

Callero, Y. and Aguilar, R., 2009. Use of simulation in egovernment process development. A case study using the simulation tool SIGHOS. *21st European Modeling & Simulation Symposium proceedings*, 1, 253-260.

Castilla, I., Muñoz, R. and Aguilar, R. M., 2007. Helpdesk Modeling and Simulation with Discrete Event Systems and Fuzzy Logic. *European Modeling and Simulation Symposium (EMSS 2007) proceedings*.

Castilla, I., García, F. and Aguilar, R., 2009. Exploiting concurrency in the implementation of a discrete event simulator. *Simulation Modelling Practice and Theory*, 17, 850-870.

Davis, R., 2001. Business process modelling with ARIS: a practical guide, *Springer-Verlag London*.

Georgakopoulos, D., Hornick, M. and Sheth, A., 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3, 119-153.

Goes, L., Pousa, C., Carvalho, M., Ramos, L. and Martins, C., 2005. JSDESLib: A Library for the Development of Discrete-Event Simulation Tools of Parallel Systems. *19th IEEE International Parallel and Distributed Processing Symposium*.

Hinz, S., Schmidt, K. and Stahl, C., 2005. Business Process Management, Berlin/Heidelberg: Springer-Verlag.

Hlupic, V. and Robinson, S., 1998. Business process modelling and analysis using discrete-event simulation. *Winter Simulation Conference Proceedings*, 25, 534-548.

Kiepuszewski, B., Hofstede, A. and Aalst, W. V., 2003. Fundamentals of control flow in workflows. *Acta Informatica*, 39, 143-209.

Kiepuszewski, B., 2003. *Expressiveness and suitability of languages for control flow modelling in workflows.* Faculty of Information Technology. Queensland University of Technology.

Kim, Y., Kang, S., Kim, D., Bae, J. and Ju, K., 2000. WW-FLOW: Web-based workflow management with runtime encapsulation. *IEEE Internet Computing*, 4, 55–64.

L'ecuyer, P. and Buist, E., 2006. Simulation in java with ssj. *Proceedings of the Winter Simulation Conference*, 611-620.

Medina-Mora, R., Winograd, T., Flores, R. and Flores, F., 1992. The action workflow approach to workflow management technology. *Proceedings of the 1992 ACM conference on Computer-supported cooperative work - CSCW '92*, 281-288.

Muthu, S., Whitman, L. and Cheraghi, S. H., 2006. Business Process Reengineering: A Consolidated Methodology. *Proceedings of the 4th Annual International Conference on Industrial Engineering Theory, Applications, and Practice, 1999 U.S. Department of the Interior - Enterprise Architecture*, 8-13.

Russell, N., Hofstede, A. H. and Mulyar, N., Workflow ControlFlow Patterns: A Revised View, *BPM Center Report BPM-06-22*.

SIGHOS project homepage. Available from: http://sourceforge.net/projects/sighos/. [APR 2010]

Workflow Management Coalition, 1999. Workflow Management Coalition Terminology Glossary. *Document Number WFMC-TC-1011*.

Wynn, M., Dumas, M., Fidge, C., Hofstede, A., and Aalst, W. V., 2008. Business Process Management Workshops, Springer Berlin Heidelberg.

AUTHORS BIOGRAPHY

YERAY CALLERO was born in Haría, Lanzarote and attended the University of La Laguna, where he studied Engineering Computer Science and obtained his degree in 2008. He is currently working on his PhD with the Department of Systems Engineering and Automation at the same university. His research interests include simulation and embedded system software design.

IVÁN CASTILLA was born in La Laguna, Tenerife and attended the University of La Laguna, where he studied Engineering Computer Science and obtained his degree in 2004. He is currently working on his PhD with the Department of Systems Engineering and Automation at the same university. His research interests include parallel discrete event simulation and computer architecture.

ROSA M. AGUILAR received her MS degree in Computer Science in 1993 from the University of Las Palmas de Gran Canaria and her PhD degree in Computer Science in 1998 from the University of La Laguna. She is an associate professor in the Department of Systems Engineering and Automation at the University of La Laguna. Her current research interests are decision making based on discrete event simulation systems and knowledge-based systems, intelligent agents, and intelligent tutorial systems.