SHAMASH. An AI Tool for Modelling and Optimizing Business Processes

David Camacho, Ricardo Aler, Daniel Borrajo
Universidad Carlos III
Avda. Universidad, 30
28911 Leganés (Madrid). Spain

Almudena Sierra-Alonso
Universidad Rey Juan Carlos
Tulipán S/N
Móstoles (Madrid). Spain

Abstract

In this paper we describe SHAMASH*, a tool for modeling and automatically optimizing Business Processes. The main features that differentiate it from most current related tools are its ability to define and use organisation standards, and functional structure, and make automatic model simulation and optimisation of them.* SHAMASH *is a knowledge based system, and we include a discussion on how knowledge acquisition did take place. Furthemore, we introduce a high level description of the architecture, the conceptual model, and other importants modules of the system.*

1. Introduction

Current organisations need a continuous and dynamic reorganisation of their processes to allow them to be more efficient. The principal aim of Business Process Reengineering (BPR) is to design techniques to allow simulate and check differents set of processes that could be improve its own organisaton [6]. This task can be accomplished manually or by using modeling tools. Currently there are many sophisticated modeling tools that help organisations on making their processes more efficient by allowing to graphically design process models and simulate them [7, 8, 12]. However, although these tools are very sophisticated, current technology can be pushed even further by automatically optimising and simulating the processes [13] and allowing to explicitly represent the standards that constrain processes [11].

Artificial Intelligence (AI) has been very successful on both, representing knowledge, which is needed for defining and using organisation standards, and optimising models. There have been already some approaches to apply AI to BPR such as ontology definitions [4, 16, 18], planning [9], multi-agent systems [5]. With respect to representing organisation standards, there is a lot of related work on computer systems for legal support, that require to represent laws using different techniques like Case-Based Reasoning (CBR) [1], ontologies [15], and also reason automatically about them [2, 17].

In this article we present SHAMASH, a tool for modeling, simulating and optimising business processes. SHAMASH shares some of its capabilities with other BPR tools, like offering an interface for process modeling, simulating these processes, and exporting processes to WPDL (Workflow Process Description Language). But it also has some characteristics which are not found in other existing tools. In particular, SHAMASH is able to automatically improve an existing model by using AI optimisation techniques. It also permits to define organisations and process standards, which are used by SHAMASH to automatically validate user process models. Another remarkable characteristic of SHAMASH is that it offers a powerful language to describe rules for the system, and also a specially built inference engine to manage them. Most of the knowledge required in the system can be represented by means of such rules. For instance, the knowledge required for optimising, for describing the behaviour of activities during simulation, and to define the standards, can all be defined by using rules. This makes SHAMASH an extensible and customisable tool. Finally, the tool allows to export the graphical representation of processes and standards into a text version (actually, html code is produced). In some organizations, processes are delivered in a mixture of graphical and text representations, resulting in consistency problems. The text generator allows to maintain the coherence between the graphical and text versions.

This article has been structured as follows. First, the general architecture of SHAMASH will be described in Section 2. One of the central components of SHAMASH (the inference engine) is explained in Section 3. The rest of the article will be illustrated by using an example related to a university maintenance process that is presented in Section 4. SHAMASH has been built by using two methodologies (IDEAL and UML), which are described in Section 5. Then, the main SHAMASH subsystems (Author, Simulation and Optimisation) are detailed in Sections 6, 7. Finally, Sections 8 summarize the conclusions.

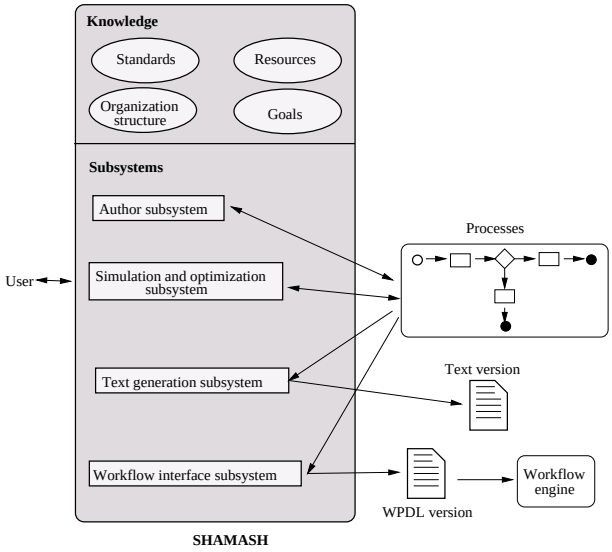

Figure 1. Architecture of the SHAMASH tool.

2. SHAMASH architecture

The general architecture of the SHAMASH tool appears in Figure 1. It is composed of four subsystems:

- **Author subsystem:** through a user-friendly interface, the user can define two types of knowledge to the system: knowledge on standards, and knowledge on processes. Standards, or norms, are statements on any organization that define how processes should behave, be created, achieve business rules, or maximize organization goals. In most cases, this type of knowledge can be easily translated into a rules formalism, so SHAMASH allows the user to interactively create these rules in a language that is easy to understand by the user. We believe that current information technology users are no longer unaware of technology, and the concept of a rule is a very close one to humans. In any case, we contemplate the idea of a programmer profile to help the process modeling user.

  Processes[1] are "computation" units within organizations. They are able to generate an output from an input, using organization resources. For SHAMASH purposes, processes are not constrained to business processes. Therefore, the tool has to be general enough to allow defining all types of behaviour to represent all types of processes, from chemical plant processes to marketing ones. See section 6 for more details on this subsystem.

[1] We will not differentiate in this article between the words processes, procedures, tasks or activities.

- **Simulation and optimization subsystem:** the tool allows to perform simulations with historical or predicted data. Results are analyzed by the system, and misbehaviours reported to the user. Also, the tool can automatically perform an optimization phase by which new optimized models are generated. The user can then decide whether to adopt the new models, or to continue with the old ones. Section 7 describes in more detail these two components.

- **Text generation subsystem:** in most organizations, processes are delivered to their end-users (human resources of the organization) in plain text. Sometimes, they are delivered using a graphical representation without the details that for obvious space restrictions can not appear in the graphical representation. And, in some organizations, processes are delivered in a mixture of graphical and text representations. A common consistency problem appears when any one of the representations, or both are updated. In those cases, the other one has to be changed, and this does not always happens. In SHAMASH this subsystem is responsible of maintaining a coherence between the graphical and text versions. When the user performs any change in the graphical representation of a process, this subsystem automatically generates a new text version of this process.

- **Workflow interface subsystem:** SHAMASH is not to be used directly as a workflow engine. Therefore, it needs to have an interface that automatically translates the defined process models into the input of a workflow engine. As for the output language, the goal would be to generate a process representation complying with the intended standard WfMC (Workflow Management Coalition) WPDL (Worflow Process Description Language). However, given that there is still no general consensus on how this language is, we have adopted a practical approach generating the output in the current version of WPDL.

Also, the tool allows the user to create and maintain knowledge about the organization that will be used when defining, simulating, and optimizing the processes. Types of knowledge that can be defined within the system are knowledge about standards, processes, organization structures, resources (human and material), or goals of the processes. Now, we will describe in more detail these modules.

3. Inference Engine

Given that SHAMASH is a knowledge-based tool using rules and objects, we had first to devise an inference engine that would take a representation of classes and instances in

C++, a set of rules, and build an efficient inference engine based on the RETE algorithm [3]. In order to do so, we designed a language for describing the rules that is based on a classical structure of *if* and *then* parts. *If* parts are composed of conditions that refer to the existence (or not) of instances of classes with some properties. The structure is similar to the Frulekit tool, developed in CMU by Peter Shell and Jaime Carbonell [14]. Apart from the fact that their tool was built in Common Lisp and ours is on C++, and the fact that the languages differ in specific aspects, one of the main differences relates to the possibility of SHAMASH users to ask in the *if* part whether a given value (either constant or variable value) belongs to a list that is the value of an attribute.

Suppose, for instance, that there is a class named `signature`, which represents the type of activity of signing a given document. One of the attributes of that class might be `allowed-signatures` which refers to organization people that can sign the corresponding document, and is represented by a list of references to instances of the class `person`. Then, one might have a rule as in Figure 2 which says that if a document can be signed by two people *person1* and *person2*, and *person1* knows more about the document than *person2*, then *person1* is the one that should sign the document (organization agent that should be the responsible of the activity). Names in italics correspond to variables.

If *signature* is a `signature` such that
person1 is in the list `allowed-signatures`,
person2 is in the list `allowed-signatures`,
assigning to *document1* its `input-document` and
document1 is a `document` such that
assigning to *keyword* its `buy-computer` and
person1 is a `person` such that
`buy-computer` is in the list `keywords` and
person2 is a `person` such that
`buy-computer` is not in the list `keywords`
Then modify object *signature*
with `responsible-agent` is *person1*

Figure 2. Example of rule in SHAMASH.

The general architecture of the RETE module is shown in Figure 3. The RETE net is created at the start of SHAMASH application with the objects and rules that configure base level SHAMASH. Afterwards, the definition of each rule triggers the RETE generator component, which creates the corresponding RETE net structure to that rule. The definition or modification of any object triggers the generation of tokens that traverse the RETE structure in order to generate the next conflict set. Whenever any SHAMASH subsystem wants to execute the rules, it should call the rules execution module, which selects one rule from the conflict set at that moment, executes the actions in the *then* part of the rule that usually cause modifications in the KB. We have also defined several ways in which rules can be executed depending on the type of ruleset they belong to. Behaviour of activities will be executed by the simulator for each activity instance in every simulation cycle. On the other hand, validation rules will be executed until no more rules of that ruleset appear in the conflict set.

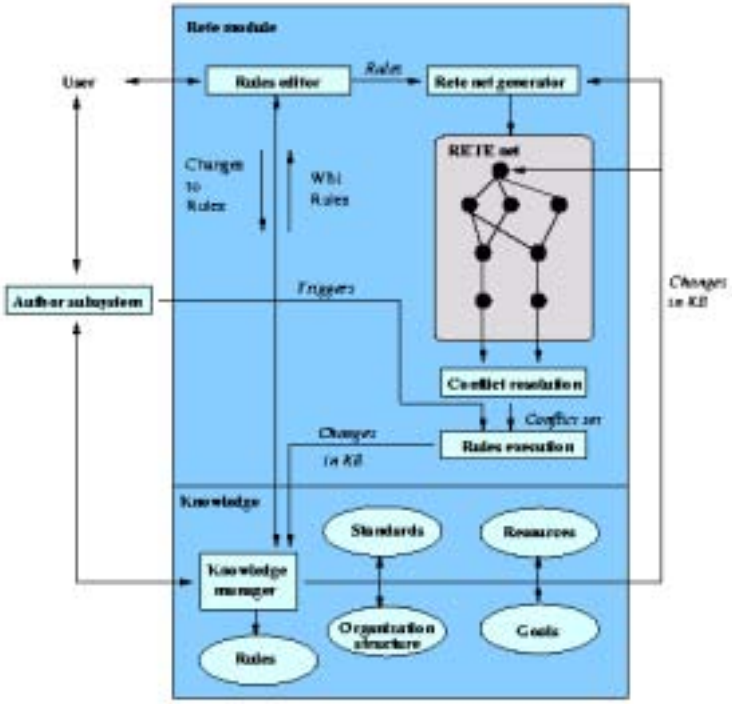

Figure 3. Graphical representation of the RETE module and its connections with the Author and Knowledge subsystems.

4. An Example

In this section we describe an example of a simplified organisation that will be used to ilustrate each one of SHAMASH modules.

In our university there is a maintenance department which offers services such as fixing furniture, producing keys, attaching blackboards to walls, etc. The goal is to represent, analise and improve the management process that is followed by this department when a request arrives. The first step is to determine whether the request can be managed by the maintenance department. Then, according to the type of request, the petition is sent to the manager for signature. Basically, each request has an importance level and if this level is very important, the manager has to sign it. Finally, it is decided whether the service will be carried out by the university staff or it will be performed by external contractors. This final decision depends on the estimated cost of the request. If it is too expensive, the service is con-

tracted. Figure 4 shows a graphical representation of this process.

In addition, this department has a manager and an assistant manager for each of the two university campuses. The maintenance staff depends directly on each of the assistant managers.

Figure 4. Maintenance Service Management

5. Knowledge Acquisition and Modeling Tasks

SHAMASH combines features of both KBS and OO systems. So we have decided to integrate both technologies for its development. From the KBS area we used knowledge acquisition techniques and knowledge representation formalisms, as production rules. From the object oriented area we have used UML notation and use cases. It has to be remarked that these methodologies will be used for requirement analysis, knowledge acquisition and system design; it is not required that the user knows any of them to use the tool. Here, we will explain the conceptual model and how we have used and integrated these technologies to build the tool. Working through the use cases has been the first task. We have develop both Use Case diagrams and Class diagrams from the knowledge acquisition process. The next task has been the elaboration of the Sequence Diagrams to model the interactions in the system. Then, the classes were fully designed with their methods, extra attributes, a more detailed set of relationships including aggregation types, cardinality, and even relationship classes that needed to be defined. From here, the design process has proceeded as a standard object-oriented one and completed with State, Activity, Components and Deployment Diagrams from UML.

5.1. Knowledge Acquisition Process

To build SHAMASH, we needed knowledge about processes, standards, validation for standards and processes, and the behaviour of processes for simulation and the optimisation of models. To address all these matters, expert knowledge was required. It has been often stated that knowledge acquisition is a bottleneck. For this reason, it is very important to plan this stage. Next, we intend to describe the two main expert acquisition issues:

- The knowledge sources: In the SHAMASH project two types of knowledge sources were used: semi-public documentation and expert knowledge. As semi-public knowledge source we used the standards of Unión Fenosa. We analysed the contents of these standards and extracted basic concepts to understand the domain. But the most important knowledge is the expert knowledge. We have extracted knowledge from two kinds of experts: BPR experts and domain experts. From the former kind, we obtained general knowledge about processes, standards, optimisation, etc. From the latter kind of experts, we obtained knowledge for building libraries for particular domains. For instance, from purchasing experts, we obtained knowledge about what processes, standards make up a typical purchasing domains, how to detect bottlenecks in such processes, etc.

- Acquisition Meeting Planning: Knowledge acquisition took place in all phases of the project. In this phase, knowledge acquisition has been split in two parts: knowledge elicitation to build the conceptual model and validation of that model. It is necessary to meet with two different kinds of experts: BPR experts and domain specific experts. BPR have been interviewed for knowledge elicitation and experts in purchasing processes have been asked for conceptual model validation. Afterwards, several meetings took place with the BPR experts taking into account the extracted knowledge. Each of the meetings was focused in each of SHAMASH subsystems (Author, Simulation and Optimisation, Text Generator and Workflow Interface). The knowledge acquisition techniqes we have used have been: open interview, structured interview, questionnaires, protocol analysis, etc. The result of this effort has been the conceptual model that represents the main concepts of the domain, the attributes of those concepts, the relationships among the concepts and the function of each concept in the solution of the problem. This model was validated by the purchasing experts of several companies.

6. Author subsystem

The author subsystem has most of the usual functions in the process modeling tools. Its main function refers to the definition of processes, and their related knowledge. Some of its characteristics are:

- **Definition of standards:** none of the analyzed current tools allows to define an important type of knowledge of any organization: its norms. They constrain how processes should be defined. A typical example are authorization levels for performing certain operations (e.g. signing documents, or approving purchases). They have different shapes depending on the organization, so the interface allows to easily create their structure, to fill them and to link them to other types of knowledge, such as the organization structure, related standards or processes, or resources to be used. If the user wants some code to represent the way in which a constrain of the organization works with respect to the processes, the tool allows to define rules to model that type of knowledge.

- **Definition of processes:** this function is common to all modeling tools, and allows the user to graphically define how processes are combined, how they relate to other processes, or how they can be decomposed or grouped into others in a hierarchical way. Since SHAMASH is a knowledge-based tool, the user can define specific behaviour of processes, or define new types of activities, processes links, or decision steps. This allows to define more simulation and optimisation knowledge into the processes than the usual one, that refers to cost or time associated to processes. In order for the user to design new processes a domain-independent ontology has been defined. That is, the ontology is generic for all workflow process models and each graphical representation of a user model instantiates this generic ontology.

  One application of this type of knowledge could be to define how individual processes provide more or less quality according to the resources employed. Another application could be to use SHAMASH as a tool for performing competencies management, by defining specific knowledge-oriented information that is needed to perform a given process, and selecting resources according to that information.

- **Validation of standards and processes:** since standards define restrictions on how processes should operate, a validation should be performed on the consistency among them. The system incorporates a set of generic validation rules, and the user can define new rules. For instance, when a standard says that approving any purchase above 1,000,000 pts. should be performed by a department head or a higher role in the organization, the validation will check whether this is so in the user created process.

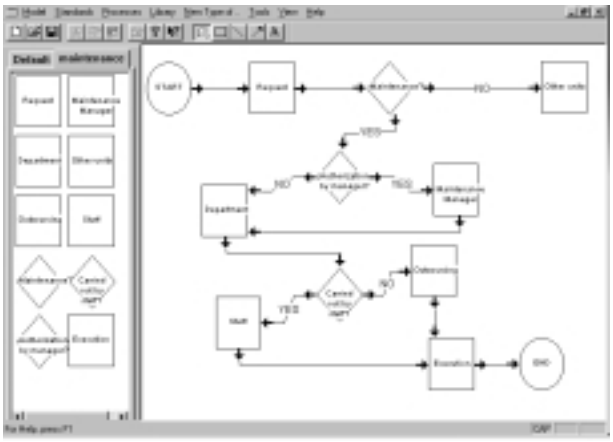

Figure 5. Maintenance Model

- **Connection among organization processes and standards:** processes in organizations are not separated from each other. Therefore, tools modeling processes need first to allow the user to create processes models independent of the rest of processes of the organization (for the sake of modularity). Then, and very importantly, such tools should allow the user to connect the related processes, such that they can be simulated and optimized in an integrated way. SHAMASH allows to do so by means of defining interconnections among processes.

- **Creation of libraries:** given that users are able to define new activities and processes with a particular behaviour, this new knowledge-based processes could be re-used in other related modeling episodes. The tool allows the user to define libraries of processes to be used in other processes modeling applications.

Figure 5 shows our maintenance model, after having been designed by the user with SHAMASH author subsystem.

7. Simulation and Optimization

The aim of this section is to explain two important modules of SHAMASH: the simulator and the optimiser. Both modules work together to optimise a model automatically, so their integration will also be explained in detail.

The simulator interface allows the user to select the process to be simulated, and to define the user goals. These user

goals are numeric values that measure how well the model did after the simulation, according to the organisation criteria. SHAMASH includes two standard user goals -time and cost- but the user can define new ones that take into account any of the simulation indicators, like queue lengths, percentage of use of a given resource, quality of the process, etc. At the end of the simulation, SHAMASH outputs a trace displaying the user goals, the aforementioned indicators, and other useful information. Besides detecting model inefficiencies and errors by watching the user goals and the indicators, the user can define rules to verify those problems automatically.

Figure 6. Example of authorisation Rule. It checks whether the document that arrives to an authorisation activity is important enough (element-level). In that case, it will be sent to the YES branch.

There are many tools that let the user simulate a model. However, SHAMASH has a feature that is rarely found in other simulators: the behaviour of activities can be defined by means of rules too. For instance, the behaviour of the authorisation activity -a decision type activity- is defined by the rule of Figure 6. Basically, this rule says that if some conditions are fulfilled, then the petition should go to the *yes* branch of the activity. Those conditions are expressed by the rule grammar which is common to any rule that can be defined in SHAMASH. If the decision level (an attribute from the petition document) is higher than the level defined in the activity, then the document will travel through the *yes* branch. It should be noted that changing the activity level might influence the model efficiency.

The simulator is an important part of SHAMASH not only because it allows the user to check the behaviour of his/her processes, but for allowing automatic optimization. Of course, the user can optimize his/her model by simulating the model, anoting where the model seems to be inefficient, changing and improving the model by hand and trying again (that is, the user can carry out what is usually called what-if analysis). However, there is no reason why this process can not be automated, and this is where SHAMASH optimisation module enters the picture, as it is explained below.

One of SHAMASH most important features is its ability to automatically optimise user process models. Optimisation is not intended just as an automatisation of what-if analysis (which is quite useful by itself), but as a first step towards adaptive workflow systems [9, 10]. Adaptive workflow aims to provide support for quickly adapting to changes both in company processes and when the process model is being enacted. Changes in company processes can occur because of new laws, standards, norms, business goals, resources, etc. Once those changes have taken place, workflow experts can redesign company processes to adapt to them. But even with the help of the simulator, this is a slow method. Automatic optimisation coupled with other features of SHAMASH (such as the ability to handle standards) can help here. All that is required from the user is to make those changes (standards, resources, etc) to the model. At this point, the model will be inefficient because some other modifications should take place in order to take full advantage of, for instance, more resources, relaxed standards, etc. The user could perform these modifications by hand, but obviously automatic optimisation would be more effective and exhaustive.

Automatic optimisation could also help in the second point addressed by adaptive workflow systems: adapt to changes when the process is being run or enacted. Such changes involve staff coming and going, hardware unexpectedly breaking down, activities taking much more time than expected, etc. Some of these changes could easily be accomodated by the existing model, but at some point it might be worthwile to dynamically modify the process model. Automatic optimisation can also help here, by automatically adapting the model to the new conditions. However, optimisation is a computer intensive process and in quickly changing environments, the optimisation algorithms might not be able to cope.

SHAMASH optimisation is based on a generate and test approach. The generate part will be achieved by using expert heuristics for generating new process models from a given one. Test or evaluation of a newly generated process model will be taken care of by means of a user supplied evaluation function. This function evaluates the model by combining different indicators obtained after simulating the process model.

The underlying paradigm for optimization in SHAMASH is search in a process space. Each node of the search space is a different user process model. This process model in-

cludes the process diagram as well as the organisation structure associated to it and the inputs to the process. Also, an evaluation function must be defined. This evaluation function calls the simulator and measures how well this particular model does. This is measured by a user goal, that can be any arithmetic expression including simulation indicators. Finally, there must be a way to move within this space of possible models. This is provided by the search operators. A search operator takes a model, transforms it, and generates a new model. Optimisation operators propose possible modifications of the initial model. Those new models are generated and simulated. The user goal obtained from the simulation is used as the worth of the model. This process continues until a timelimit or until one of the generated models has a "good enough" worth (this has been specified by the user in advance).

In a model, there are many things that could be changed: increase the number of resources, remove agents, change the process diagram, increase/decrease decision-making parameters, etc. However, the number of possible models that can be obtained by performing such changes at random would be enormous and would make the search process very time consuming. SHAMASH answer to this problem is to use knowledge acquisition to elicit from the expert how to make changes that produce benefit to the model under study. This knowledge will be formalized later into search operators. It would seem that acquiring perfect search operators (that is, those that always generate better models) would be the best option. However, such perfect operators need not exist, and in any case, they would be very difficult to elicit. Therefore, search operators that are likely to generate better models should suffice, although in some cases they might degenerate the model. Such operators would be enough to constrain the search enough to make it efficient. The evaluation function would be used to focus the search further, by choosing the best generated alternative models.

SHAMASH allows to define the search operators by means of rules too, in the very same language used in other parts of the tool. The left hand side of the rules will access the knowledge base and match static features of the model (model diagram, resources, etc) and dynamic ones (bottlenecks, idle resources, etc) and determine how the model should be changed so that it is likely that it will be improved. A simple search operator is shown in Figure 7. This rule changes (increases) the authorisation level of the authorisation activity which was mentioned in previous paragraphs. There is another rule to decrease the same attribute. Therefore, in this simple example, the optimiser can fine-tune automatically one of the free parameters of the model.

The kind of heuristic search described above fits into many different search algorithms, such as hill climbing, simulated annealing, beam search, heuristically augmented genetic programming, etc. In the current version of the tool, best first search has been used. This search method always focuses on the best model (according to the user goal), and generates all possible modifications of this model, according to the applicable search operators. In the future, SHAMASH might use a beam search technique.

Figure 7. Optimisation Rule. It says that if the model has a `Authorisation` activity and its authorisation level is not already too large ($<$50), then a new model can be generated by increasing it.

At this point, the user can either accept the new model proposed by the optimiser or maintain the initial model.

8. Conclusions

In this paper we have presented an overview of a process modeling tool named SHAMASH. SHAMASH allows users to define, simulate, and optimise BPR models. There are in the market many other tools that provide functionalities for modeling processes, but we have identified two areas where technology could be pushed further:

- **Definition of standards:** from the user point of view, it is very interesting that BPR tools allow to define and use knowledge about organisation standards, as SHAMASH does.

- **Automatic optimisation:** Currently, BPR tools simulate the models and allow the user to change them if s/he spots any problem in the simulation results. SHAMASH goes further and automatically suggest changes that improve the model.

Another important feature of SHAMASH is that some of the knowledge in the system can be represented inside the tool by means of rules, which users usually understand better than other formalisms like computer programs. This makes SHAMASH a very extensible tool. In SHAMASH, the behaviour of activities, validation rules, organisation standards, and optimisation operators, can all be defined by rules. In order to handle them efficiently, SHAMASH includes a RETE algorithm that has been completely integrated with the rest of the tool.

We intend SHAMASH to be an adaptable tool by both providing basic process libraries and by making its architecture modular. However, in order to provide the two important features mentioned before (standards and optimisation), expert knowledge is required. To acquire this knowledge, it is necessary to use a knowledge based methodology. We have used knowledge acquisition techniques to obtain the expert knowledge and knowledge representation formalisms, such as production rules. For instance, they have been used to represent optimisation and validation rules. Knowledge for this kind of systems comes from two different sources. The knowledge required for handling standards and optimisation needs a BPR expert, whereas the knowledge for building libraries needs an expert in the specific domain of that library (i.e. the purchasing domain).

Acknowledgements

The research reported here was carried out as part of the research project funded by CICYT TAP-99-0535-C02 (http://decsai.ugr.es/∼lcv/SEPIA/tap99-0535-c02-01.html).

References

[1] K. D. Ashley. Case-based reasoning and its implications for legal expert systems. *Artificial Intelligence and Law*, 1992.

[2] J. Breuker and N. d. Haan. Separating world and regulation knowledge: where is the logic? In M. Sergot, editor, *Proceedings of the third international conference on AI and Law*, pages 41–51, New York, 1991. ACM.

[3] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern matching problem. *Artificial Intelligence*, 19:17–37, 1982.

[4] G. M. Fox, M.S. Enterprise modelling. *AI Magazine*, pages 109–121, Fall 1998 1998.

[5] T. Gray, E. Prez, D. Pinard, S. Abu-hakima, A. Daz, and I. Ferguson. A multi-agent architecture for enterprise applications. In W. Hamscher, editor, *Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering*, August 1994.

[6] M. Hammer and J. Champy. *Reengineering the Corporation*. Harper Business Press, New York, 1993.

[7] T. T. S. C. HPS. Ithink. www hps-inc.com/bus_solu/ithink/ithink htm, 2000.

[8] IDS-Scheer. Aris. www.ids-scheer.com/aristoolset.htm, 2000.

[9] P. Jarvis, J. Moore, J. Stader, A. Macintosh, A. C. du Mont, and P. Chung. Exploiting AI technologies to realise adaptive workflow systems. In *Agent-Based Systems in the Business Context. AAAI'99 Workshop*, 1999. Submitted.

[10] M. Klein. Workshop towards adaptive workflow system. In *Proceedings of the 1998 Conference on Computer-Supported Cooperative Work*, 1998.

[11] U. Reimer, A. Margelisch, B. Novotny, and T. Vetterli. Eule2: A knowledge-based system for supporting office work, 1998.

[12] Rockwell-Software. Arena. www.sm.com, 2000.

[13] W. J. Salter. Organizational designs cannot be optimised. In W. Hamscher, editor, *Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering*, August 1994.

[14] P. Shell and J. G. Carbonell. FRuleKit: A frame-based production system. User's manual. Internal paper, 1989.

[15] J. B.-C. Trevor and R. V. Pepijn. Ontologies in legal information systems; the need for explicit specifications of domain conceptualisations. In *Sixth International Conference on Artificial Intelligence and Law*, page 132. ACM Press, 1997.

[16] M. Uschold and M. Gruninger. Ontologies: Principals, methods and applications. *Knowledge Engineering Review*, 11(2), 1996.

[17] R. Winkels and H. d. Bruijn. Making a case for case frames. *Information & Communications Technology Law*, 7(2):117–133, 1998.

[18] E. Yu and J. Mylopoulus. Organization modelling for business processes reengineering. In W. Hamscher, editor, *Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering*, August 1994.