



ELSEVIER

Knowledge-Based Systems 15 (2002) 473–483

Knowledge-Based
SYSTEMS

www.elsevier.com/locate/knosys

A knowledge-based approach for business process reengineering, SHAMASH

Ricardo Aler^{a,*}, Daniel Borrajo^a, David Camacho^a, Almudena Sierra-Alonso^b

^aUniversidad Carlos III, Avda. Universidad, 30, 28911 Leganés Madrid Spain

^bUniversidad Rey Juan Carlos, Tulipán S/N, 28933 Móstoles Madrid Spain

Received 13 June 2001; accepted 7 January 2002

Abstract

In this paper we present an overview of SHAMASH, a process modelling tool for business process reengineering. 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. Furthermore, we introduce a high level description of the architecture, the conceptual model, and other important modules of the system. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Business process reengineering; Automatic optimisation; Knowledge-based tool

1. Introduction

Current organisations need a continuous and dynamic re-organisation 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 different sets of processes that could improve its own organisation [9]. This task can be accomplished manually or by using modelling tools. Currently there are many sophisticated modelling tools that help organisations on making their processes more efficient by allowing to graphically design process models and simulate them [11, 12,20]. However, although these tools are very sophisticated, current technology can be pushed even further by automatically optimising and simulating the processes [21] and allowing to explicitly represent the standards that constrain processes [19].

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 [7,24,28], planning [13], multi-agent systems [8]. 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) [3], ontologies [23], and also automatically reason them [5,27].

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 workflow process description language (WPD). 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

* Corresponding author. Fax: +34-91-624-9129.

E-mail addresses: aler@inf.uc3m.es (R. Aler), dborrajo@ia.uc3m.es (D. Borrajo), dcamacho@ia.uc3m.es (D. Camacho), asierra@escet.urjc.es (A. Sierra-Alonso).

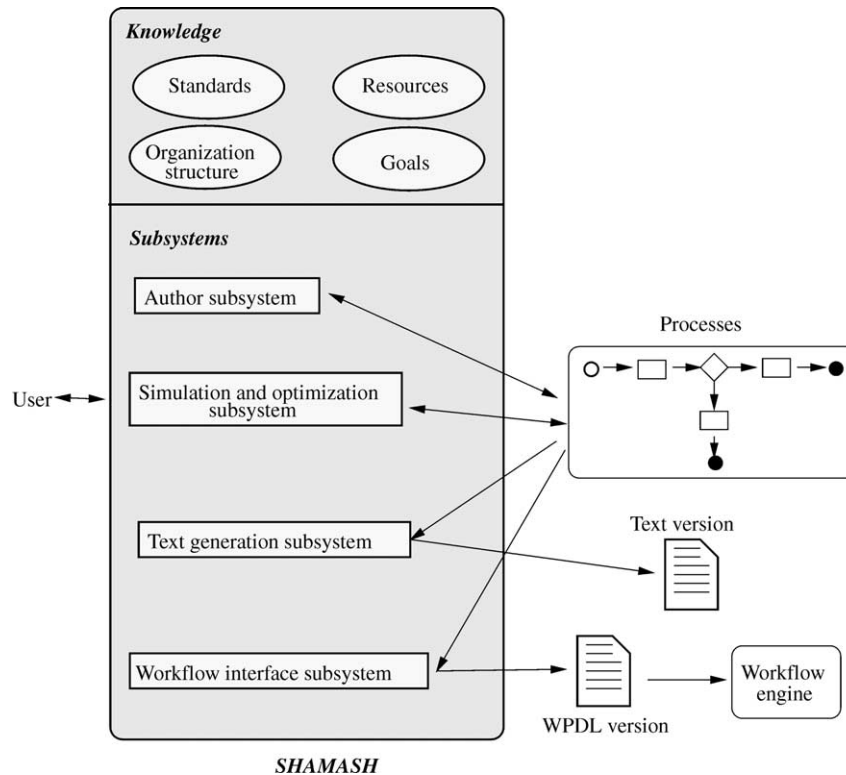


Fig. 1. Architecture of the SHAMASH tool.

produced). In some organisations, 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, and text generator) are detailed in Sections 6–8. Finally, Sections 9 and 10 summarise the conclusions and the future lines of work.

2. SHAMASH architecture

The general architecture of the SHAMASH tool appears in Fig. (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 organisation that define how processes should behave, be created, achieve business rules, or maximise organisation goals. In most cases, this type of knowledge can be easily translated into

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.

Processes¹ are ‘computation’ units within organisations. They are able to generate an output from an input, using organisation 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.

Simulation and optimisation 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 optimisation phase by which new optimised 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 organisations, processes are delivered to their end-users (human resources of the organisation) in plain text. Sometimes, they are delivered using a graphical representation without the

¹ We will not differentiate in this article between the words—processes, procedures, tasks and activities.

```

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

```

Fig. 2. Example of rule in SHAMASH.

details that for obvious space restrictions cannot appear in the graphical representation. And, in some organisations, 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 happen. In SHAMASH this subsystem is responsible for maintaining 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 workflow management coalition (WfMC) workflow process description language (WPDL). 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 organisation that will be used when defining, simulating, and optimising the processes. Types of knowledge that can be defined within the system are knowledge about standards, processes, organisation 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 to first 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 [6]. 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 [22]. 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 organisation 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 Fig. (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 (organisation agent that should be the responsible of the activity). Names in italics correspond to variables.

The general architecture of the RETE module is shown in Fig. (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. For instance, 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.

4. An example

In this section we describe an example of a simplified organisation that will be used to illustrate each one of the 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, analyse 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

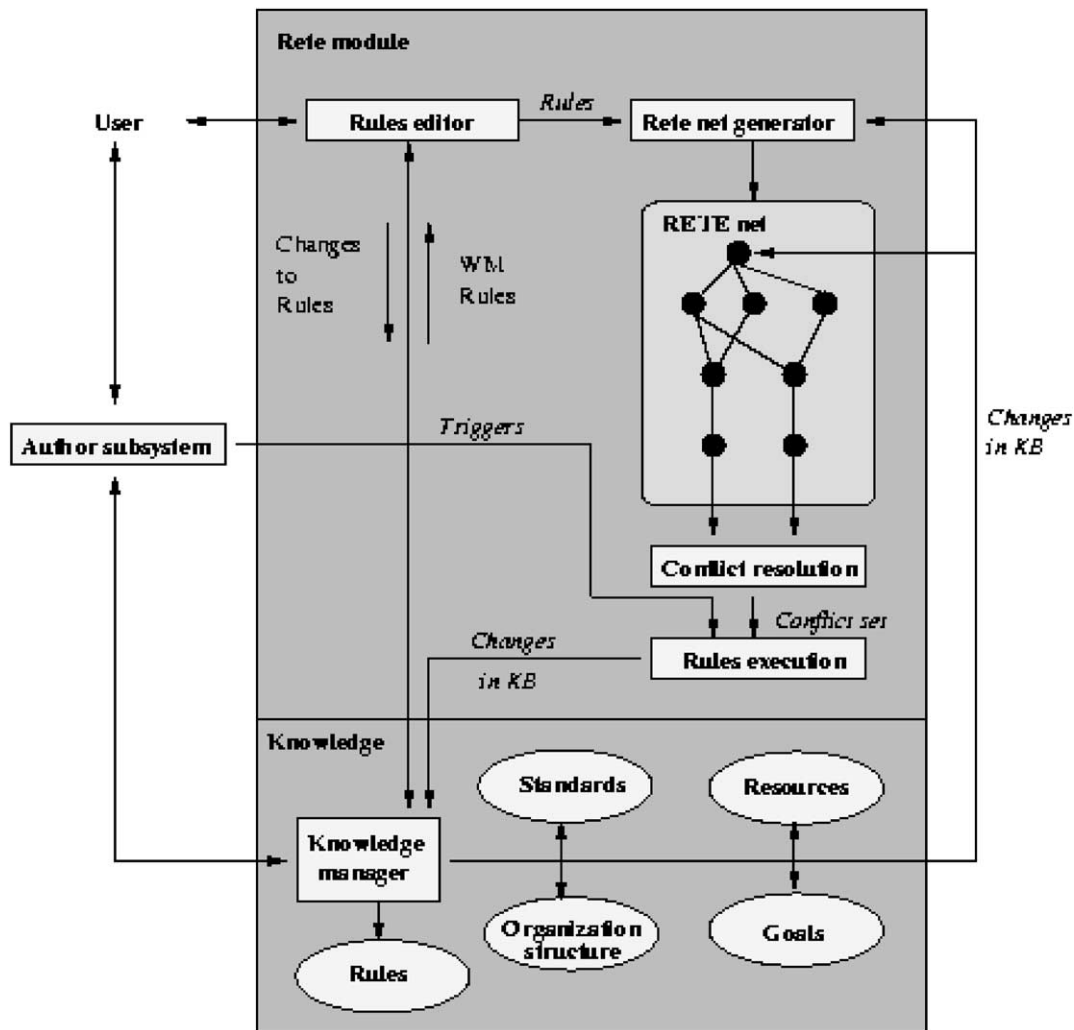


Fig. 3. Graphical representation of the RETE module and its connections with the author and knowledge subsystems.

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 contracted. Fig. (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.

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 developed 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

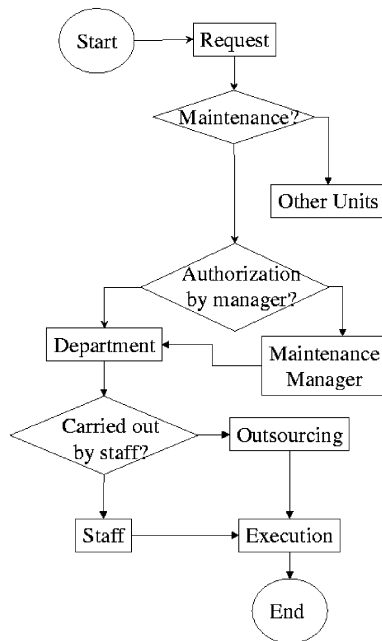


Fig. 4. Maintenance service management.

knowledge was required. Therefore, an intensive knowledge acquisition task has been carried out. It has been often stated that knowledge acquisition is a bottleneck. For this reason, it is very important to plan this stage. When we began the knowledge acquisition we wondered about the following questions:

- What are the knowledge sources?
- Which techniques and methodology we were going to use to acquire the knowledge to simulate the process behaviour, optimise the processes and validate processes vs. standards?
- How to carry out the acquisition meetings?

Sections 5.1.1 and 5.1.2 intends to answer the questions above in the light of our experience within the project.

5.1.1. 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 and domain experts. From the former, 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.

5.1.2. 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. As we said in Section 5.1.1, it is necessary to meet with two different kinds of experts: BPR and domain specific experts. BPR have been interviewed for knowledge elicitation and experts in purchasing processes have been asked for conceptual model validation.

In order to obtain the basic concepts of the BPR domain, we have analysed semi-public documentation available from Unión Fenosa, EDP, and WIP. Afterwards, several meetings took place with the BPR experts taking into account the extracted knowledge from the documentation mentioned before. Each of the meetings was focused in each of SHAMASH subsystems (author, simulation and optimisation, text generator and workflow interface). The knowledge acquisition techniques 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 Unión Fenosa, EDP, and WIP.

6. Author subsystem

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

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

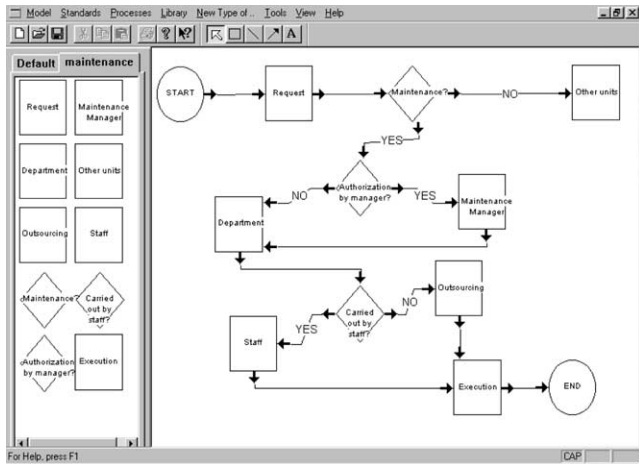


Fig. 5. Maintenance model.

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 by a higher role in the organisation, the validation will check whether this is so in the user created process.

Connection among organisation processes and standards. Processes in organisations are not separated from each other. Therefore, tools modelling processes need first to allow the user to create processes models independent of the rest of processes of the organisation (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 optimised 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 modelling episodes. The tool allows the user to define libraries of processes to be used in other processes modelling applications.

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

7. Simulation and optimisation

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.

Once a process model has been either designed anew or modelled after an already existing company process, the user is usually interested in detecting problems this model might have. SHAMASH's answer to this requirement is twofold:

- Detecting problems in a static manner: this is achieved by using validation rules, as it has already been described in Section 6.
- Spotting problems that can only be detected after the model has been run: for instance, underused resources or bottlenecks. Running the model in reality to check for its faults would be both too slow and too risky. SHAMASH provides a simulator module to allow an off-line analysis.

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

Address: C:\shamash\Simulator\SimulationTrace.html

- Process Finishing Time: 72
- Predefined Indicators:
 - Time Indicator: 220.0
 - Cost Indicator: 0.0
- Human Resources:

	Number	Identifiers List	Cost
Possibles (Before Simulation)	0	[]	---
Possibles (After Simulation)	0	[]	---
Used	0	[]	0.0
Idle	0	[]	0.0

END SIMULATION OF PROCESS: MAINTENANCE2

Fig. 6. Simulation trace.

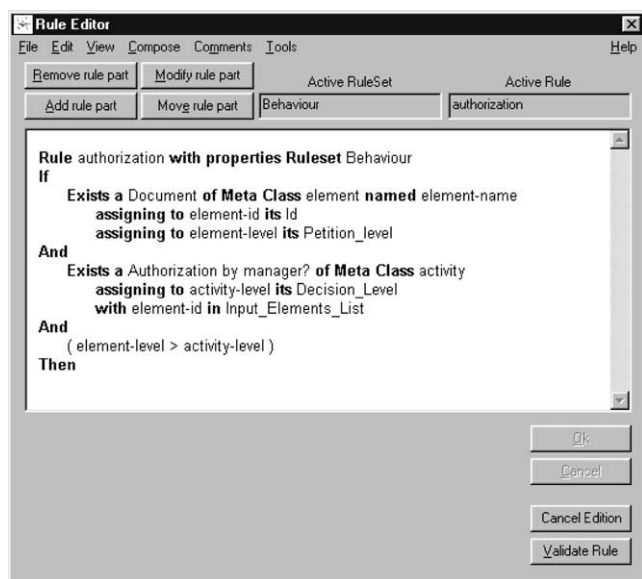


Fig. 7. Example of authorisation rule: it checks whether the document that arrives to an authorisation activity is important enough (element-level); in that case, it will be sent to the YES branch.

indicators, and other useful information. Part of the simulation trace can be seen in Fig. (6). Besides detecting model inefficiencies and errors by watching the user goals and the indicators, the user can define rules to verify those problems automatically.

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 Fig. (7). 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 optimisation. Of course, the user can optimise his/her model by simulating the model, making note 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 cannot be automated, and this is where SHAMASH optimisation module enters the picture as explained below.

One of SHAMASH's most important features is its ability to automatically optimise user process models. Optimisation is not intended just as an automatization of what-if analysis (which is quite useful by itself), but as a first

step towards adaptive workflow systems [13,15]. 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 accommodated by the existing model, but at some point it might be worthwhile 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. This basic behaviour of the optimiser can be seen in Fig. (8).

The underlying paradigm for optimisation in SHAMASH is search in a process space. Each node of the search space is a different user process model. This process model includes 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. See Fig. (9) for a graphic visualisation of the search space and the optimisation process. Optimisation starts with the process model at the bottom of Fig. (9). Optimisation operators (the arrows that

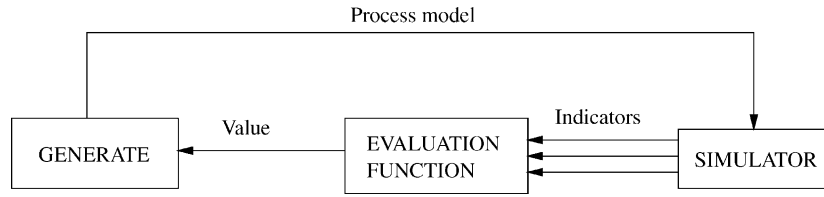


Fig. 8. Basic behaviour of the optimiser: new process models are generated from old ones. Then, they are simulated and performance indicators are obtained, which are subsequently used to guide the generation of new models.

leave the initial model) 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. In Fig. (9), two new models are proposed. The first one has a worth of 3 and the second one, of 7. The best model (that with a worth of 7) is selected and used as the new seed to generate new models. 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 formalised 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.

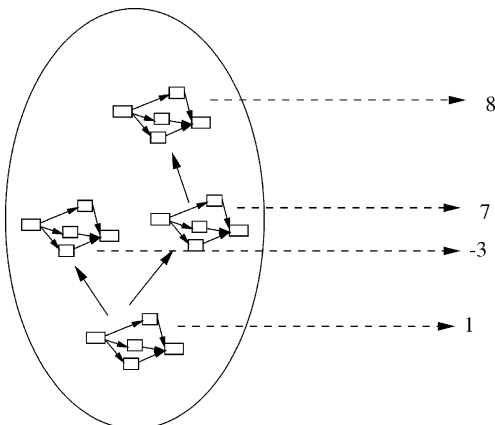


Fig. 9. SHAMASH optimisation problem.

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 (bottle-necks, 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 Fig. (10). 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.

Once the optimiser has been run, it returns an explanation of why the model obtained is better than the old one. More specifically, SHAMASH returns the sequence of search

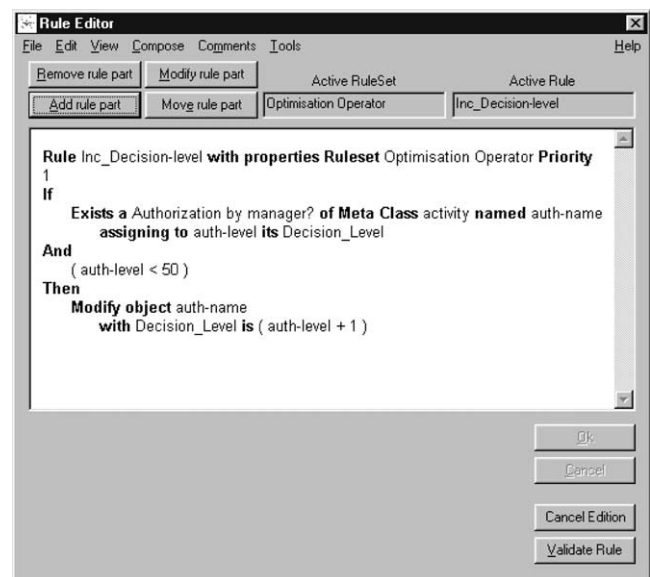


Fig. 10. Optimisation rule: it says that if the model has an Authorisation activity and its authorisation level is not already too large (<50), then a new model can be generated by increasing it.

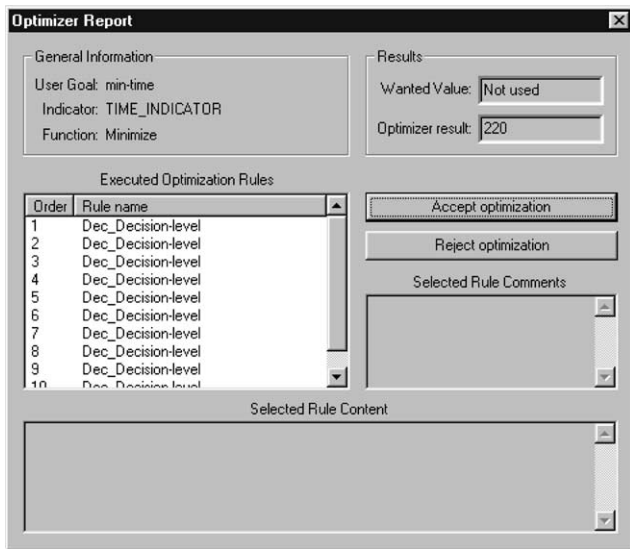


Fig. 11. Optimiser result: this window shows the user goal used for optimisation (top left), the user goal value obtained by the most optimised model (top right), and the list of operators that were applied to the initial model to obtain the optimised model (middle left).

operators that were applied to the initial model to obtain the better model. It also displays the user goal value for the new model. This can be seen in Fig. (11).

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

8. Text generator

In most organisations, processes are delivered to their end-users (human resources of the organisation) in plain text. Sometimes, they are delivered using a graphical representation without the details that for obvious space restrictions cannot appear in the graphical representation. And, in some organisations, 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 for maintaining coherence between the graphical and text versions. When the user performs any change in the graphical representation of a process, this subsystem will automatically generate a new text version of this process.

This module produces HTML files that describe the process diagrams and their relationships with other components of SHAMASH. The user needs to follow the following steps to use the TG:

1. If the user wants to build a text version for a process, (s)he first selects the A/Ps wanted to be translated.
2. Then, the TG method would walk through the set of related standards and the process diagram (by following the connections between processes) and will produce the HTML file.
3. If the user wants to modify an old text version, (s)he should select the A/Ps to be modified.
4. The textual contents of the process or standard can be modified using the text editor.
5. SHAMASH has implemented two main functions for the text generator process: generate and modify.

When the HTML file is obtained, it is possible to browse related information to the set of related standards and the process diagram. Fig. (12) shows an example of the generated HTML text version of the process in Fig. (4).

Since the TG generates HTML files, they can be modified by the user using any text editor, Netscape Composer, the text editor given by SHAMASH, etc. Therefore, the user will have unlimited freedom to enrich the text version by adding information and statements. Of course, the user is responsible of ensuring the coherence between the text and the graphic version, after (s)he performs any change in the automatically generated HTML file.

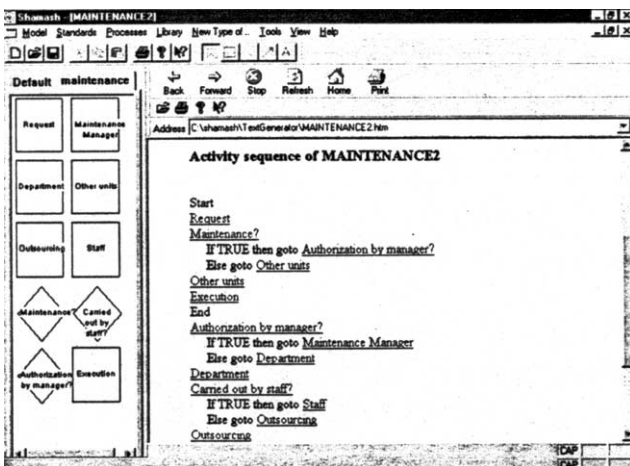


Fig. 12. HTML translation of process in Fig. 4.

9. 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 modelling 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).

10. Future lines of work

This project provides with many interesting opportunities to apply Artificial Intelligence research results. They will not be applied in the first version of SHAMASH but we expect to use the following AI techniques in the future:

- Currently, SHAMASH evaluation function returns just one value. However, the user might want to optimise functions including several user goals. This is called multi-objective optimisation. Multi-objective optimisation can be addressed by using hierarchical evaluation functions [1], or more rigorously by using Pareto algorithms like in Refs. [4,17,18,25].
- SHAMASH allows to introduce stochastic effects in several parts of the model like decision activities, arrival rates, etc. In that case, the model returned by the optimiser would depend on the random effects that happened during the successive simulations. A quick way to solve this problem is to simulate each model several times, to get average results. However, this would make optimisation too time consuming. Further research might solve this problem.
- A related problem is that simulation results depend on the scenario that has been used. A scenario describes the

sequence and rate of arrivals of the inputs to the process. If a model is optimised according to a single scenario, it might not be valid for other scenarios. Several simulations per model might be the answer to this, but again this would be very inefficient.

- Efficient search operators could be learnt by using machine learning techniques such as [16] macro-operators.
- Best practices could be used to both build search operators for optimisation or to carry out case based planning (improving process models by analogy with best practices).
- Using planning techniques to obtain process diagrams. The approach we are currently following consists on the user defining activities through the author subsystem interface, translating them into planning operators in planning description language (PDL), executing a planner to generate a plan (sequence of instantiated activities), and translating back into SHAMASH [14]. We are using a nonlinear planner, PRODIGY4.0, to generate those plans [26]. This scheme allows the user to focus on the requirements of the process to be generated and the organisation structure, and let the planning system build the best model for those requirements.
- Another way of automatically generating those models is through the optimiser. There is a strong similarity between how it works and the planning technique called planning by rewriting as described in Ref. [2]: a plan is supplied (in SHAMASH case the plan is an activity diagram) and then domain dependent search operators are used to rewrite that plan (or other components of the model, like number of resources) into another more efficient plan (optimisation).
- Efficient optimisation procedures could be learnt by using machine learning techniques similar to macro-operators [16], case based reasoning [10] or search control knowledge in case we would use a planner [1].

Both issues are currently being discussed in recent forums such as PLANET (network of excellence in AI planning), where the authors of this paper collaborate.

Acknowledgments

The authors would like to thank the work of all components of the UC3M team that allowed to build this tool: J. David Arias, Nuria Cortijo, Fernando Fernández, José I. Giráldez, Noyda Matos, Carla Salazar and César J. Soto. The research reported here was carried out in the course of the R + D project funded by the Esprit Programme of the Commission of the European Communities as project number 25491, and co-financed by CICYT TIC98-1847-CE. We thank the partners of this project, who have originated and contributed to the ideas reported. They are UF (Unión Fenosa), SAGE (Software AG España),

SEMA GROUP sae, UC3M (Universidad Carlos III de Madrid), WIP (Wirtschaft und infrastruktur & Co Planungs KG), and EDP (Electricidade de Portugal).

References

- [1] R. Aler, D. Borrajo, P. Isasi, Genetic programming and deductive-inductive learning: a multistrategy approach, in: J. Shavlik (Ed.), Proceedings of the Fifteenth International Conference on Machine Learning, ICML'98, 1998, pp. 10–18, Madison, Wisconsin, July.
- [2] J.L. Ambite, C.A. Knoblock, Flexible and scalable query planning in distributed and heterogeneous environments, in: R. Simmons, M. Veloso, S. Smith (Eds.), Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS-98), AAAI Press, 1998, pp. 3–10, Pittsburgh, PA, June.
- [3] K.D. Ashley, Case-based reasoning and its implications for legal expert systems, *Artificial Intelligence and Law* (1992).
- [4] T. Blickle, Theory of evolutionary algorithms and application to system synthesis, PhD thesis, Swiss Federal Institute of Technology, November 1996.
- [5] J.A. Breuker, N. den Haan, Separating world and regulation knowledge: where is the logic? in: M. Sergot (Ed.), Proceedings of the Third International Conference on AI and Law, ACM, New York, 1991, pp. 41–51.
- [6] C.L. Forgy, Rete: a fast algorithm for the many pattern/many object pattern matching problem, *Artificial Intelligence* 19 (1982) 17–37.
- [7] M. Gruninger, M.S. Fox, Enterprise modelling, *AI Magazine* Fall (1998) 109–121.
- [8] T. Gray, E. Prez, D. Pinard, S. Abu-hakima, A. Daz, I. Ferguson, A Multi-agent architecture for enterprise applications, in: W. Hamscher (Ed.), Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering, 1994, August.
- [9] M. Hammer, J. Champy, *Reengineering the Corporation*, Harper Business Press, New York, 1993.
- [10] K.J. Hammond, Case-based planning: an integrated theory of planning, learning and memory. PhD Thesis, Yale University, 1986.
- [11] The Thinking Systems Company: HPS, Ithink, www.hps-inc.com/bus_olu/ithink/ithink.htm, 2000.
- [12] IDS-Scheer, ARIS, www.ids-scheer.com/aristoolset.htm, 2000.
- [13] P. Jarvis, J. Moore, J. Stader, A. Macintosh, A. Casson-du Mont, P. Chung, Exploiting AI technologies to realise adaptive workflow systems, Agent-Based Systems in the Business Context, AAAI'99 Workshop, 1999, submitted for publication, 1999.
- [14] P. Kearney, D. Borrajo, An R & D agenda for AI planning applied to workflow management, in: B. Stanford-Smith, P.T. Kidd (Eds.), *E-Business: Key Issues, Applications and Technologies*, IOS Press, Ohmsha, 2000, pp. 1072–1078, October.
- [15] M. Klein, Workshop Towards Adaptive Workflow System, Proceedings of the Conference on Computer-Supported Cooperative Work (1998).
- [16] R. Korf, Learning to solve problems by searching for macro operators, PhD Thesis, Pitman, 1985.
- [17] W.B. Langdon, Evolving Data Structures Using Genetic Programming, Research Note, RN/95/1, UCL, Gower Street, London, WC1E 6BT, UK, January 1995.
- [18] W.B. Langdon, Pareto, population partitioning, price and genetic programming, research note, RN/95/29, University College London, Gower Street, London WC1E 6BT, UK, April 1995.
- [19] U. Reimer, A. Margelisch, B. Novotny, T. Vetterli, EULE2: a knowledge-based system for supporting office work, 1998.
- [20] Rockwell-Software, ARENA, www.sm.com, 2000.
- [21] W.J. Salter, Organizational designs cannot be optimised, in: W. Hamscher (Ed.), Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering, 1994, August.
- [22] P. Shell, J.G. Carbonell, FRuleKit: a frame-based production system. User's manual, Internal paper, 1989.
- [23] J.M. Bench-Capon Trevor, R.S. Visser Pepijn, Ontologies in legal information systems; the need for explicit specifications of domain conceptualisations, Sixth International Conference on Artificial Intelligence and Law, ACM, New York, 1997, p. 132.
- [24] M. Uschold, M. Gruninger, Ontologies: principals, methods and applications, *Knowledge Engineering Review* 11 (1996) 2.
- [25] D.A. Van Veldhuizen, G.B. Lamont, Evolutionary computation and convergence to a Pareto front, in: J.R. Koza (Ed.), Late Breaking Papers at the Genetic Programming Conference, University of Wisconsin, Stanford University Bookstore, Madison, Wisconsin, USA, 1998, pp. 22–25, July.
- [26] M. Veloso, J. Carbonell, A. Prez, D. Borrajo, E. Fink, J. Blythe, Integrating planning and learning: the prodigy architecture, *Journal of Experimental and Theoretical AI* 7 (1995) 81–120.
- [27] R.G.F. Winkels, H. de Bruijn, Making a case for case frames, *Information and Communications Technology Law* 7 (2) (1998) 117–133.
- [28] E. Yu, J. Mylopoulos, Organization modelling for business processes reengineering, in: W. Hamscher (Ed.), Working Notes of the AAAI-94 Workshop on Artificial Intelligence in Business Process Reengineering, 1994, August.