Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 389–406
Expert Systems with Applications
www.elsevier.com/locate/eswa

Integrating planning and scheduling in workflow domains

María Dolores R-Moreno [a,*], Daniel Borrajo [b], Amedeo Cesta [c], Angelo Oddi [c]

[a] Departamento de Automática, Universidad de Alcalá, Carretera Madrid-Barcelona, Km. 33,600, 28871 Alcalá de Henares, Madrid, Spain
[b] Departamento de Informática, Universidad Carlos III de Madrid, Avda. de la Universidad, 30, 28911 Leganés, Madrid, Spain
[c] ISTC-CNR-Italian National Research Council, Viale Marx 15, I-00137 Rome, Italy

Abstract

One of the main obstacles in applying AI planning techniques to real problems is the difficulty to model the domains. Usually, this requires that people that have developed the planning system carry out the modeling phase since the representation depends very much on a deep knowledge of the internal working of the planning tools. On some domains such as business process reengineering (BPR), there has already been work on the definition of languages that allow non-experts entering knowledge on processes into the tools. We propose here the use of one of such BPR languages to enter knowledge on the organisation processes to be used by planning tools. Then, planning tools can be used to semi-automatically generate business process models.

As instances of this domain, we will use the workflow modeling tool SHAMASH, where we have exploded its object oriented structure to introduce the knowledge through its user-friendly interface and, using a translator transform it into predicate logic terms. After this conversion, real models can be automatically generated using a planner that integrates planning and scheduling, IPSS. We present results in a real workflow domain, the telephone installation (TI) domain.

Keywords: Planning and scheduling; Workflow management systems; Business process reengineering; Makespan minimisation; Time windows

1. Introduction

In the last years, companies and markets have quickly grown in complexity so they are looking for flexible and dynamic ways to efficient manage their resources and processes. Due to the competitive nature of businesses and the strong pressure of the market, quality initiatives and the gradual improvement of processes are not enough for the continuous and dynamic changes that current organisations need. Levels of changes so radical need new and powerful tools that can allow the efficient redesign and management of the organisation. Also improving customer service and increasing customer retention is gaining importance. This is the main objective of BPR (Hammer & Champy, 1993).

Over the past few decades, BPR has become fashionable among medium and big enterprises due to its capability to present solutions that improve this type of management. Although there have already been many approaches to the computer-aided design of processes, very few have focused on the automatic generation of process models that have in mind the organisation resources as well as their capabilities and availability.

Once the organisation has been studied in depth from a process and resources perspective, corresponding models are modeled in order to handle processes and resources computationally. Business processes are usually represented as workflow, that is, computerised models within which all the parameters needed for the completion of the processes can be defined: resources involved, orders, tasks, conditions, goals, quality criteria, information flow, etc. Workflow management systems (WFMSs) (Leymann & Roller, 1994; Medina-Mora, Winograd, & Flores, 1993; Mohan, 1997) have been deployed in sectors like insurance, banking, accounting, manufacturing, telecommunications,

* Corresponding author. Tel.: +34 918856607.
E-mail addresses: mdolores@aut.uah.es (M.D. R-Moreno), dborrajo@ia.uc3m.es (D. Borrajo), a.cesta@istc.cnr.it (A. Cesta), a.oddi@istc.cnr.it (A. Oddi).

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.05.027

administration and customer service (Lydiard, Jarvis, & Drabble, 1999) but it does not have significant commercial impact yet, because of problems related to the cost of process modeling and the inflexibility in their execution.

We can see WFMSs as a set of methods and technologies that allows modeling and managing some of the processes that happen in the company (there are some processes that will likely never be modeled in workflow systems because, they are either too difficult to formalize, or too dynamic). It can provide active support to a business process by controlling the routing of work around the organisation automatically. This is done based on the business processes models describing the flow, the decisions, the exceptions, the resources to be used, etc.

WFMS co-ordinates user and system participants, together with the appropriate data resources, which may be accessible directly by the system or off-line to achieve defined goals by setting deadlines. The co-ordination involves passing tasks to participants' agents, and ensuring that all complete their tasks successfully. In case of exceptions, actions to solve the problem can be triggered, or human operators alerted.

Optimising the organisation procedures, routines and resource management, several aspects must be considered. Examples are the activities or tasks that should be performed; the organisation model that describes the roles of each agent (software or human), who can perform what in the organisation; and the information model that describes which information is needed to perform an activity.

In the last few years an increasing development of documentation tools and/or process modeling techniques have emerged that represent and reason computationally about the knowledge of the current processes and resources. Usually, the task of defining those models is performed with the aid of a set of tools that provide a graphical representation of them, together with the relations among the activities that occur within the processes. The human is usually an expert in the processes that take place in the organisation. Many issues need to be considered on this task and implemented like the reusability of past processes, accessibility to the models by the different agents in the organisation, consistency of usage, or selection of the right model.

Prior to WFMS, many enterprises created special-purpose tailored applications to support their processes. The advantage of WFMS-based solutions is that the workflow representation is explicit, and separated from the application code. This means that a WFMS can be customised quickly to support a new business or process, and that workflows are relatively easy to modify when a process changes. Current WFMS do not address all aspects of the problem, however. In general, they do not deal with scheduling, that is, resources management/allocation. Similarly, while they provide means of generating exception events when things go wrong, they do not have a built-in re-planning function or automatic methods for adapting the workflow model according to those exceptions. They do, however, provide interfaces so that application-specific modules performing these functions can be integrated.

Workflow systems hold the promise of facilitating the everyday operation of many enterprises and work environments. Despite the popularity of these products, there is still a lack of maturity in some respect, i.e., a lack of a semantic associated to the models or an easy way to reason about that semantic, that could be overcome using techniques coming from other fields such as artificial intelligence (AI).

Without any doubt, the application of AI techniques to Workflow Management systems has created a big expectation. The AI community and in particular the planning and scheduling field, has been applying successful techniques in different and complex domains like robotics, satellites or military logistics. In these domains, there are activities that must be performed (planning) in a temporal horizon that consume or produce resources (scheduling). During execution, completion of activities, and delays and other problems are detected to take the appropriate measures (rectify the situation, or in more drastic cases, a new plan) to satisfy the goals. In order to represent this information, rich representation models are needed, the majority of them based on predicate logic as is the case of the planning standard language, PDDL2.2 (Edelkamp & Hoffmann, 2004). There is also the HTN representation where planning problems and operators are organised into a set of tasks. High level tasks are reduced into a set of lower levels and the way to do it can be done in several ways as in Yang (1997) and Kuter et al. (2005).

In the past, some researchers saw the advantages of the integration of AI planning and scheduling for workflow generation, as shown by the existence of a Technical Co-ordination Unit of the European research network on planning and scheduling, PLANET (PLANET), on applications of planning and scheduling to workflow. This has lead to some exploratory work reflected in a Roadmap PLANET and some related work (Hannebaeur, 1999; Kearney & Borrajo, 2000; Myers & Berry, 1999; R-Moreno & Kearney, 2002).

The aim that we want to achieve with this integration is double. From one hand, given that the majority of BP tools are based on objects and rules, we propose to translate this knowledge into first order predicate logic, in particular into the planning domain definition language PDDL2.2.

On the other hand, we propose an integrated P&S framkework to automatically solve BRP problems. In the literature there are basically two approaches to solve this type of problems. The *component based approach* (Cesta, Pecora, & Rasconi, 2004), where the two subproblems of planning and scheduling are just solved one after the other, and the *integrated approach* (Ghallab & Laruelle, 1994; Tate, Drabble, & Kirby, 1994) where there is an uniform representation without the decomposition over two sequential subproblems. We believe that systems that integrate

P&S, as IPSS (R-Moreno, 2003), are the best suitable candidates and we will explain why IPSS is one of the best planners suited for the automatic solving of BPR problems.

The paper is structured as follows: Section 2 describes the main concepts that Workflow and AI planning systems share. Next, Sections 3 and 5 introduces instances of both domains, SHAMASH and IPSS, explaining on Section 4 how the integration can be possible with a simple example of installing a new telephone line in a telecom company that is explained in detail on Section 6. Then, Section 7 shows some results of IPSS against state of the art planners. Finally, Sections 8 and 9 outlines the future work and conclusions.

2. Workflow management and AI planning and scheduling

The first step for the integration of workflow management and AI planning and scheduling Systems is to identify points in common by understanding the way both work.

2.1. WFMS

The main module in WFMS is the workflow engine that is in charge of coordinating the model execution and determining the agents (human or software), the data, the business constraints and other applications implied in the process. The Workflow Management Coalition (WFMC) Management Coalition provides a general architecture framework where five interfaces are identified with relation to the workflow engine:

- Tools to model, define and analyse the flow of the control in the processes.
- Administration module.
- Monitoring module.
- Simulation module.
- Re-engineering workflow module.

The workflow engine must include: an assignation task module in charge of assigning the tasks among the different agents that participate in the processes, a knowledge baseline of the organisation structure and a module to handle the exceptions and failures recover.

When modeling workflow systems, we need to identify and use definitions of the different elements that will be used, such as:

- Tasks: each task is an activity or a set of actions handled as a whole. They require a set of resources, and when these resources include humans they usually play a role for each task.
- Persons (users): the tasks are performed in a defined order by specific human resources (or agents playing the human roles) based on rules or business constraints.
- Roles: each role defines a set of potential competences that exist in the organisation. They are defined independently of the persons to whom the roles will be assigned.
- Routing: each routing defines the sequences of steps that the documents (or data) must follow within the workflow system. It is very important that the WFMS has the capability of routing the tasks to remote or occasional users.
- Transitions: are logical rules that determine the flow of the document within the system. They express what action is going to be executed depending on the value of logical expressions. The rules can have varying complexity with multiple options, variations and exceptions.
- Data: are the documents, files, images, registers in the data base and other type of information used to perform the work.
- Event: is an interruption that contains information. The events can be triggered by the user or appear during a process according to the state of the data or decisions made by the user.
- Deadline: is the maximum time that is assigned to a task to be concluded, the maximum time to execute a path, or to finish a task before a given date, etc. We can assign events to deadlines so when the time limit is reached some events can be triggered.
- Process: is a set of linked activities within an organisation.
- Policy: is a characterization of a set of constraints or a recommendation on how tasks should be performed.

Fig. 1 shows the workflow architecture proposed by the WFMC. The Process Model is the first stage in the adoption of a Workflow solution and involves the crucial task of revealing and recording all of the manual and automatic internal business processes of an organisation. From there, the user designs, models, optimises and simulates the organisation's processes through user friendly interfaces. We include in this stage the design of the process templates that can be instantiated and enacted by a workflow system. Then, it comes the process planning where the activities required to achieve some user goals are instantiated, resources assigned, and a preliminary scheduling performed. These two stages are included in the process definition interface of Fig. 1.

Next, the enactment/execution stage is defined, where agents (software or humans) carry out the activities, with the workflow system co-ordinating execution. The monitoring stage runs concurrently with Enactment/Execution, so status information is made available to human operators. Exceptions, such as deviation from the plan, and subsidiary processes are initiated to solve problems.

2.2. AI planning and scheduling

AI planning and scheduling (AI P&S) consists of a set of techniques on modeling, searching for solutions of problems with time and resources, execution, monitoring and repair. Although they have been two very related fields they have evolved separately. Planning generates a plan (sequence or parallelization of activities) such that it

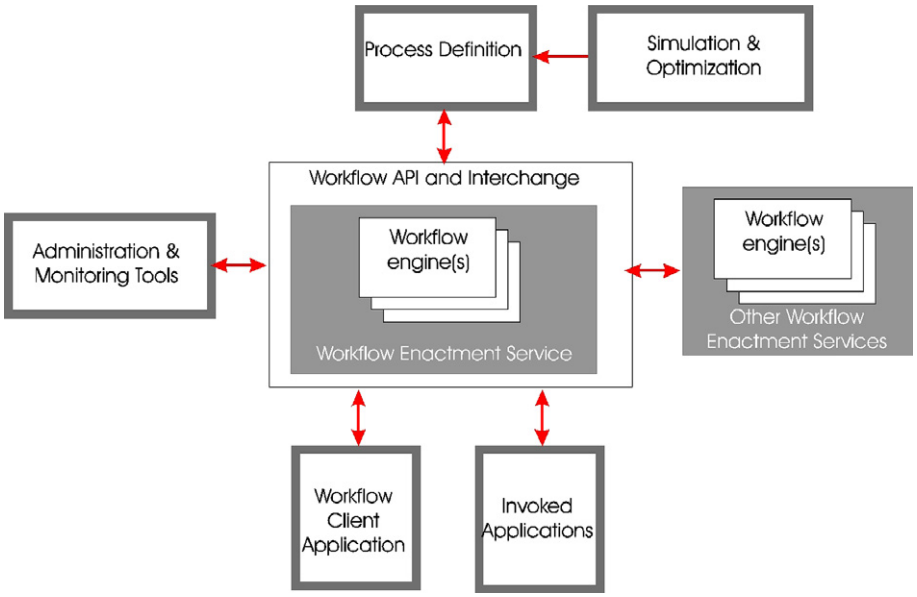

Fig. 1. Architecture proposal by WFMC.

achieves a set of goals given an initial state and satisfying a set of domain constraints represented in operators schemas. In scheduling systems, activities are organised along the time line having in mind the resources available. These systems can perfectly handle temporal reasoning and resource consumption, together with some quality criteria (usually centred around time or resource consumption) but they cannot produce the needed activities and their precedence relations given that they lack an expressive language to represent the activities. Traditionally, planning was first performed and the solution was given as an input to the scheduling systems.

So in these systems, the user should first supply a domain description that is composed of a set of operators. An instance of a business process, (e.g., the accounting process) is analogous to a plan in AI. In business process management (BPM) terminology, a plan also includes allocation of resources and target start and end times, while in AI terminology this task is usually performed by scheduling techniques. Currently, there is an increasing interest to integrate these two fields because of real domains needs as for example, workflow domains. Thus, systems as IPSS (R-Moreno, 2003; R-Moreno, Oddi, Borrajo, Cesta, & Meziat, 2004b), SAPA (Do & Kambhampati, 2003) or IXTET (Ghallab & Laruelle, 1994) are suitable candidates to solve this type of problems.

At a high level, the inputs of a planner are:

- Domain theory: the STRIPS representation originally proposed by Fikes and Nilsson is one of the most widely used alternatives (Fikes & Nilsson, 1971). It was introduced to overcome what were seen as computational difficulties in using states to build plans. In the STRIPS representation, a world state is represented by a set of logical formulae, the conjunction of which is intended to describe the given state. Actions are represented by so-called operators. An operator consists of pre-conditions (conditions that must be true to allow the action execution), and post-conditions or effects (usually constituted of an add list and a delete list). The add list specifies the set of formulae that become true in the resulting state while the delete list specifies the set of formulae that are no longer true and must be deleted from the description of the state.
- Problem: is described in terms of an initial state and goals. Those states are represented by a logical formula that specifies a situation for which one is looking for a solution.
  - Initial state: in planning, one has to specify the starting situation of the posed problem.
  - Goals: are often viewed as specifications for a plan. They describe the successful behaviours that execution of the plan should produce; they specify what one would like to be true at the end of the solution of the problem for a given time.

AI planners generate as an output a plan or set of plans if a solution exists. A plan can be seen as a sequence of operator applications that can lead from the initial state to a state in which the goals are reached.

Once we have described the main elements in AI P&S systems, we can identify the following phases when using them for solving problems:

- Domain modeling: in this phase the user introduces the knowledge to the system, that is, the operators, the initial conditions and goals. Each planner has its own syntax although lately there has been an effort to unify the

syntax in a common language: the Planning Domain Definition Language, now in the 2.2 version (PDDL2.2) (Edelkamp & Hoffmann, 2004).
- Planning and scheduling: the plan is outlined as a set of instantiated actions in a given order. Commonly, plans do not contain information about resources, so in some problems planning and scheduling can be separated. In other cases, this idea has to be abandoned and mechanisms to handle resources through constraining equations must be integrated to solve the problem.
- Execution: this stage relates to the actions' execution.
- Monitoring: the results of the actions execution can differ from the actions expected results, so monitoring must take place to anticipate events or re-plan if the initial plan cannot be achieved.

2.3. Common phases in both systems

AI P&S and BPM are complementary disciplines with much to gain from collaboration. The ability to invoke AI components flexibly and dynamically from within the workflow framework would considerably enhance business productivity. Table 1 outlines at a high level the concepts that AI P&S share with the Workflow community (for a more detailed description we refer to the Workflow Management PLANET TCU PLANET roadmap).

Each BPR domain (e.g., the accounting process in an organisation) can be defined in terms of a set of activities (they are also called tasks or even processes) that are performed by organisation agents (either human or software). Therefore, there is a strong relation between operators in planning and activities in BPR.

For BPR, a problem might be described as a process that has to be designed (modeled) for a particular task to be performed within the organisation. For instance, modeling the purchasing of an organisation would be the problem that one has to solve. In the initial state would be all knowledge that the organisation has about itself and has to be for a specific process within the organisation. Examples of information that should appear in the initial state would be: the hierarchical and/or functional representation of the organisation, the resources that it uses and when they are available, the documents that are generated, etc.

Table 1
Concepts mapping between AI P&S and workflow

AI P&S	Workflow
Modeling + planning and scheduling	Modeling and Scheduling
Execution	Enactment
Re-planning	Exceptions
Monitoring	Monitoring
Operators	Activities, tasks,...
Initial State	Organisation, resources
Goals	Business goals, service provision,...

The goals might be represented by the business goal of the organisation with respect to that process. For instance, a purchase process has to be completed, having in mind a set of time or cost constraints, or even some user satisfaction criteria.

As a result, BPR processes can be sequences of activities, adding conditional branches or activities occurring simultaneously (that is, plans in P&S).

Another common approach to model workflow system is the Petri Nets formalism (Purvis, Purvis, & Lemalu, 2001). It is a formal and graphical language which is appropriate for modeling concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic systems. It is a particular case of directed graph with an initial state called the initial marking. There are two kinds of nodes: places and transitions. Arcs are either from a place to a transition or a transition to a place. The basic problem with Petri Nets is that they are so simple that even a small system needs many states and transitions (known as the *state explosion*). Instead, we will use the AI planning formalism to model processes and also validate them avoiding inconsistencies and it can give other solutions in case exceptions occur (re-planning).

3. The SHAMASH tool

Once the different stages between both areas have been identified, in this section we describe the features of a workflow modeling tool, SHAMASH. SHAMASH (Aler, Borrajo, Camacho, & Sierra-Alonso, 2002), a R&D project funded by the IV Esprit Program, generated a process modeling tool that allows simulation, modeling and optimisation of business processes taking into account a realistic model of the organisation. This aspect combined with the features that the tool embodies, make SHAMASH a powerful tool for BPR. The richer modeling capabilities allow among others, modeling of organisation standards and rules of procedures, business goals, automatic validation and optimisation of the models, or generation of HTML output, such that people from the organisation can freely browse the processes in which they participate.

The SHAMASH tool consists of four subsystems as Fig. 2 shows:

- Author subsystem: this module provides a user-friendly interface that enables definitions of three types of knowledge: on standards (policies), on processes and on the organisation. The standards, or norms, constrain how processes should behave according to the internal company criteria. The interface allows the user to create their structure, related standards or processes, organisation resources to be used, business goals, flow elements and its attributes. Processes are computational *units* within the organisations. The user can describe the sequence of steps or conditionals to be completed to accomplish some goal. Each step or activity has pre- and post-conditions that define its behaviour. Once the

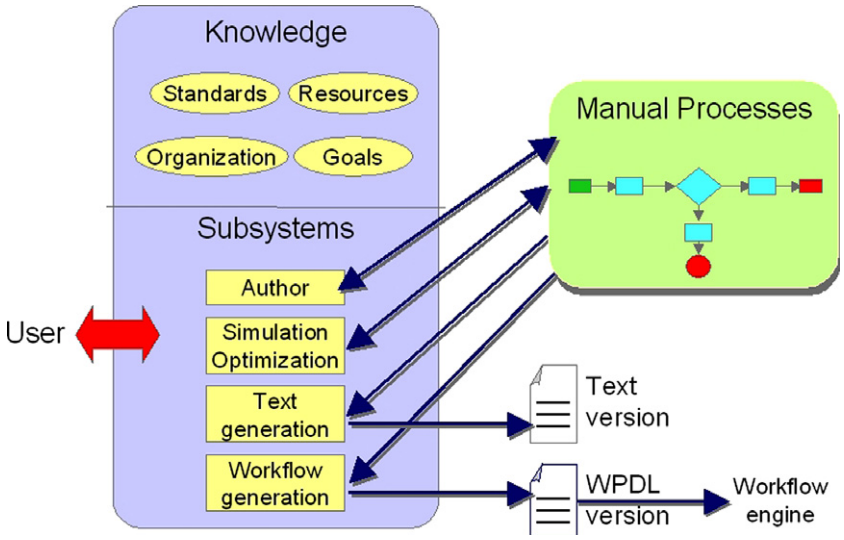

Fig. 2. SHAMASH architecture.

user has defined his/her processeses, the tool allows connection of the related process so they can be simulated and optimised. SHAMASH also allows the user to define libraries of processes to be used in other modeling applications.

- Simulation and optimisation subsystem: this is an important part of the modeling tool. The simulator can check the behaviour of the process that the user has created. The simulation is a semantic-based one, given that it has into account the behaviour defined for each activity using a rule-based representation and historical data or different probabilities distributions. SHAMASH also helps in detecting static problems (using validation expert heuristics) and spotting problems that can only be detected when the model is running (as underused resources or bottlenecks). Also, the tool is able to automatically perform an optimisation phase by which new better models are generated by searching the space of process models and using business goals as search metrics. The user can decide whether to adopt the new models, or to continue with the old ones.
- Text generation subsystem: this subsystem is responsible for maintaining coherence between the graphical (defined by specific people in the organisation) and the text versions (the ones seen by most people in the organisation). SHAMASH allows to automatically generate a new HTML version in accordance with the changes made.
- Workflow interface subsystem: SHAMASH cannot directly be used as a workflow engine. An interface is needed to automatically translate the defined process models into the input of a workflow engine. As a result, a WPDL (Workflow Process Description Language), the standard language generated by the WFMC, can be generated as output of the process representation.

The tool also allows the user to create and maintain knowledge about the organisation through an interface. The behaviour of the model is defined through objects and rules with a well-defined and easy-to-use syntax. This allows detailed specification of how activities are performed, what inputs they take and in what format, how the outputs are generated from the inputs, etc.

3.1. SHAMASH as a knowledge based system

The Knowledge Base of SHAMASH and the applications developed with it contain all the knowledge about the domain: the objects, their methods, and rules to define the processes and validation behaviour. The object-orientation helps in two ways: first, it is a good way to structure the knowledge on processes and organisations; second, it can be user extendible by using object inheritance.

When defining a domain, the user starts by selecting and adding, through the interface, the new objects and attributes that belong to the process. In SHAMASH the main classes that represent the organisation are:

- Activity: defines the types of actions involved in the process. For instance, in a process of installing a new line, receiving the customer order, building a cable or sending a invoice.
- Element: defines the elements and their features (attributes such as name or the capacity type: binary multicapacity) that flows through a process. That is, the inputs and outputs of the tasks. For instance, an invoice, or a new line.
- Resource: defines the resources available in the organisation, such as people or software and their features (attributes). There are several types of resources such as human or material resources, and the last one can be consumable or available.
- Role: defines the roles that the people on the organisation can play in the mentioned activities, as Engineer or Technician.

- Unit: defines the units in an organisation that can be either individuals or groups of people, such as Engineer Department, or Tom Miller.

3.2. Rule based approach

SHAMASH follows a rule-based approach for defining the behaviour of some elements in the Knowledge Base. The main rationality for having used rules for defining behaviour was that the user companies selected them as the knowledge formalism that was closer for them to represent the knowledge about the behaviour of the elements of the processes. Rules can access any represented information about the organisation, its structure, its processes, as well as its business rules (or standards). Therefore, very rich semantic behaviour can be defined within the processes, which account for a more realistic representation of them.

In order to help the user on entering the rules, a syntax-driven Rule Editor was defined. In the Rule Editor, the user can easily define, modify or remove rules in order to describe the behaviour of standards and processes, establish validation functions and optimisation criteria, etc.

For the integration of the IPSS with SHAMASH we had to translate from the BPR based language of SHAMASH into the IPSS language (Carbonell et al., 1992), as it is described below.

The SHAMASH ontology on processes adopts an activity-centred modeling viewpoint (Myers & Berry, 1999). The activities use resources to satisfy the process requirements defined by the user. Every activity has a set of preconditions that must be true to execute an activity or a process.

Rules can be used to define these conditions and to verify if an activity can be executed or not, or if it has achieved what it was supposed to (process goals). In SHAMASH there are three types of rules that can be triggered in an activity: pre-condition, behaviour and post-condition rules. The first type are represented as rules that have to be fired before the activity execution. The post-conditions rules have to be true after the activity execution and the behaviour activities define what the activity should do. Any of the mentioned rules is composed of two parts:

- The left part of the rule: defines the preconditions of the rule, i.e., when the right part of the rule can be executed.
- The right part of the rule: defines the postconditions.

An example of a behaviour rule syntax in pseudocode is shown in Fig. 3. This rule says that if there is a Spare Line with attributes *IsCable* equal to *No* and *Available* equal to *Yes*, and there is an Employee with *hours-left* greater or equal to three, then after the execution of the activity the field *IsCable* will be modified to *Yes* and the employee will have in its activities list to build the cable.

The activities need objects (e.g., data or information) that are created and modified within a process, through a series of activities. In Fig. 3, *BuildCable* and *Available* are attributes of the *SpareLine* object. The *SpareLine* will be modified by different activities depending on its initial values. The user will specify in each activity the input and output data. The activities also need resources to be performed and those are assigned according to the organisation structure and the activity rules of behaviour. In the example, we need an employee with at least 3 h available; if there is any, the Spare Line task will be assigned to the list of tasks that the resource (employee) must perform. SHAMASH allows the user to define the type of resource that will be associated to each activity during the definition stage.

Once the user has defined all the data, activities and organisation structure that will be part of his/her process, he/she should specify the flow of control, i.e., the order in which activities are executed, and how they are linked. In the case of big processes, this stage is usually quite tedious for the user: draw all the activities, decision points and all the control connectors. The focus of our work is to automatically generate the model, avoiding going through all the drawing process, and making sure that the established connections among activities conform a valid sequence of activities. After the model has been automatically generated, the user can simulate and optimise the process using SHAMASH, as it had been defined by him/her. If

```
<Rule Build-Cable-Activity>
   <Left-part>
      Check if there is a SpareLine with the values
         IsCable = No
         Available = Yes
         Resource = (Exists Employee with hours-left >= 3)
   <Right-part>
      Assign the new values to the SpareLine
         IsCable = Yes
      Assign the new values to the resource Employee
         assigning to list to-do-activities = Build-Cable
```

Fig. 3. A pseudocode example of a SHAMASH behaviour rule.

the model does not satisfy his/her expectations, he/she could modify it as required.

4. The automatic model generation

In this section we make the connection between AI planning and workflow tools more precise, by describing first how to translate an organisation model described in terms of SHAMASH into a planning domain model for IPSS, how IPSS can produce the desired plan (model), and how this is translated back into SHAMASH (R-Moreno, Oddi, Borrajo, Cesta, & Meziat, 2004a).

To automate the process models generation in workflow modeling tools, and in particular, in SHAMASH, we need to translate the rich semantic representation of SHAMASH into AI planning languages. This is not an easy tasks as these languages do not use ontology representation (attributes) neither rules. The translation must be done not only at the syntactic level but also at the semantic level.

The general translation process presented in this section could serve to any planner that uses logical formulae. We have used IPSS (R-Moreno et al., 2004b) because it can reason about time (deadlines, time windows, etc.) and it can separate the resource reasoning from the logical reasoning what makes it a good candidate for workflow domains (not all the state of the arts planners can do this reasoning although they are based on logical formulae). A first step for the automatic process generation was to use the PRODIGY planner (Veloso et al., 1995) but the disadvantages of using this planner were: the needs of domain dependent coded functions for the time reasoning, not explicit language for time and resources, impossibility of imposing deadlines at specific goals or providing as output a Total Order plan.

To overcome the drawbacks of the PRODIGY planner, we have used IPSS because it can reason about parallel actions, time and resources, or minimise the makespan as these domains require. The automatic process generation using IPSS is as follows: first, the system translates the SHAMASH objects and rules into the IPSS language (Carbonell et al., 1992). Then, IPSS produces a parallel plan of activities. And later, this sequence is translated back into SHAMASH to be presented graphically to the user. This process is shown in Fig. 4. The translation has in mind the different semantics of BPR and AI planning concepts as well as their similarities:

Predicates: IPSS domain theory is an augmentation of the STRIPS representation. In this representation, a world state is represented by a set of grounded literals, the conjunction of which is intended to describe the given state. The states in SHAMASH are represented as C++ objects (that is, instances, attributes and their values). To represent them in IPSS we have generated, for each tuple ⟨instance, attribute, value⟩ in SHAMASH, a binary predicate whose name is the attribute name. Its two arguments correspond to the identifier of the instance that it belongs to, and the value of the attribute. For instance, one of the attributes of the order document element in the installing a line process of any telecom company is the type of payment method (visa, check or cash). If there is an instance of an order document, *order-Client123*, whose payment method attribute has *check* as value, then this is translated into the predicate:

```
(Payment order-Client123 Check)
```

Operators: task dependency is represented in BPR as activities with pre- and post-conditions, following the WFMC standard. As mentioned before, in SHAMASH there are three types of rules that can be triggered in an activity: pre-condition, behaviour and post-condition rules and each rule has pre-conditions and post-conditions. On the other hand, IPSS has operators with preconditions and effects. Therefore, each activity name is translated into an operator name, the pre-condition rules and the left-hand side of the behaviour rule in an activity into the operator preconditions and the right-hand side of the behaviour rules and post-conditions rules into the operator effects. Pre and post-conditions refer to questions about the contents of the process at a given moment. Most of these questions

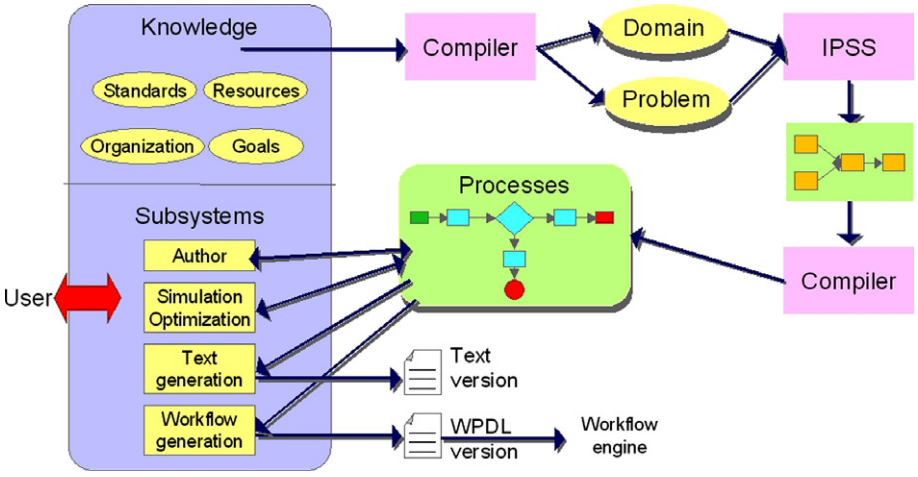

Fig. 4. SHAMASH and IPSS integration.

```
Operator-Name                  Start-Time    End-Time
(Set-Spare-Available Spare-3)          0            4
(Build-Cable Spare-3)                  4            7
(Check-Line Line-3)                    0            5
```

Fig. 5. IPSS example of a plan.

refer to knowing the value of a given attribute of an instance, or setting it. In order to translate those questions and settings, the translation of attribute values into literals in the previous paragraph is taken into account.

Also, to establish the precedence between activities in a process, they must have the same input/output elements. For instance, in a installing a line process (explained in more detail in the next section) the activity *Set-Spare-Available* precedes the activity *Check-Line*, so the output element of the *Set-Spare-Available* activity (Spare Line element) must be the same as the input element of the activity *Check-Line*. In IPSS, input/output elements are operator parameters (*params* in Fig. 9), that is, the variables that will appear with the operator name when it is instantiated.

Types: IPSS requires that each variable in the operators belongs to a user-defined type. Since SHAMASH follows an object-oriented approach, it represents all static knowledge as classes and instances. Therefore, every class in SHAMASH is translated into a IPSS type. Thus, the Element class will be translated as:

```
(ptype-of Element: top-type)
```

and the classes that belong to the Element class (as the Spare Line element) as:

```
(ptype-of SpareLine Element)
```

SHAMASH has also five basic types that are translated into IPSS user-defined types. For example, SHAMASH represents integer numbers as type integer. Given that IPSS has the capability of allowing to define variables with continuous values, called infinite types, this SHAMASH type is defined in IPSS as such.

Problem: apart from the domain theory, IPSS also needs the problem initial state and goals. In SHAMASH the initial state is represented as the instances of the organisation units, roles, resources and elements defined by the user for each BPR domain (activities and organisation). For example, the instance that represents a human resource (e.g., `Tom`) can have its availability as one of its attributes. If it is considered a boolean attribute, this information is translated as:

```
(Availability Tom TRUE)
```

IPSS allows to separate the resource reasoning from the causal reasoning (see Section 5.2). Most of the planners consider discrete resources like agents (human or software) as logical predicates. This causes the search to become intractable when the number of resources increase as the number of causal links increases. These results are shown in Srivastava, Kambhampati, and DO (2001) and R-Moreno (2003).

The resources defined in a hierarchy in SHAMASH are treated in the *resource* field in IPSS (see Fig. 9).

The availability (represented as such as in SHAMASH) as a numeric value can also be represented. This could also be translated into IPSS as:

```
(Availability Tom 0.7)
```

Goals: in relation to IPSS goals, we translate the user goals defined in the SHAMASH process into the IPSS problem goals. As an example, in a Installing a Line process, the aim of the process might be defined as client satisfaction. If we define it as when the client has allocated the line for using the phone line, it could be represented in IPSS as:

```
(Allocated-Line order-Client123 TRUE)
```

Also if the conditions of the problem require to specify a deadline in the plan or in some (all) of the goals, this can be easily specified by a parameter when running IPSS in the first case or using the reserved words *before*, *after*, *at* and *within* in the second case. IPSS is able to handle any of the Allen prmitive (Allen, 1984). For example, we can express that a required Line (the one with identifier number 3) should be allocated before 6 o'clock as follows:

```
((Allocated Line3 TRUE) before 6)
```

Metrics: if there were quality metrics in SHAMASH for the process, they could also be translated into IPSS. The quality metric(s) that want to be considered within the process model generation can be represented in each operator using the *cost* field (Fig. 9 shows an example of the definition of the time quality metric). If any metric is defined, IPSS will try to find a plan minimising that metric.

Plan: the planner generates a sequence of instantiated operators (activities). This sequence or plan represents the activities instances or tasks dependency in a process model. A plan in IPSS for a simplified process would look like the Fig. 5. It shows when each operator starts and finishes considering as the origin time value the 0. Then, the plan is translated back into SHAMASH, where it is graphically presented. Once the activities are translated back to SHAMASH interface, the user can change its order, add/delete new activities if the results are not as expected or improve the model obtained by simulating and optimising it.

5. IPSS: an integrated system

IPSS R-Moreno (2003) and R-Moreno et al. (2004a), based on the non-linear metric planner QPRODIGY (Borrajo, Vegas, & Veloso, 2001), integrates AI planning and scheduling techniques. In IPSS, the planner and the scheduler interleave information during the solving process. It is subdivided in two levels: IPSS-P that corresponds to the planning reasoner, and IPSS-S that corresponds to the scheduling reasoner. The planner focuses on the actions selection (possibly in the optimisation of some quality metric different than

time-resource usage), and the scheduler on the time and resource assignments. IPSS-P is composed of the QPRODIGY planner and the Deordering layer that transforms the total ordered (TO) plan to a parallel plan that the scheduling component needs. IPSS-S is composed of the Ground-CSP and Meta-CSP layers, for the time and resources reasoning. The whole architecture is shown in Fig. 6.

The planning level performs a bi-directional search: it begins by performing backward search from the goals, selecting which goals to achieve, which operator to use to achieve the corresponding goal, and which bindings to use for its variables. As soon as it can apply any selected operator, it decides whether apply it or continue subgoaling. If, at any decision point, it applies one operator, the planner consults the scheduler for the time and resource consistency. If the resource-time reasoner finds the plan inconsistent, then the planner backtracks. If not, the operator gets applied, current state is modified, and search continues. The planner also allows to choose actions under a pre-defined metric: *actions that consume less quantity of a given resource*, *actions that take less time in being executed*, *actions that maximise the capacity*, etc. The scheduler also helps using other optimisation criteria during the solving process as minimising the makespan or maximising resource usage.

5.1. IPSS-P

The planner we are using generates TO plans. This type of plans is not appropriate if we are looking for time and resource optimisation. That is, some orderings can be removed from the incomplete plan being generated by the planner, allowing parallel executions of operators. This is referred as *deordering* of the solution, and IPSS has a module that transforms the incomplete TO plan into an incomplete partial ordered (PO) plan. Given that the deordering process can be generally NP-HARD (Bäckström, 1998), we have followed a greedy incremental heuristic approach to avoid searching in the space of all possible deorderings. The *Minimal Link Deordered* heuristic tries to place the operators as near to the origin and between them as possible, that is to minimise the makespan. This heuristic makes the *Deorder-layer* not complete, but we obtain good results as the experimental section shows.

The deordering algorithm starts computing the link that satisfies that the preconditions of the operator that is going to be added, is supported by the effects of an operator in the list of applied operators. The links search starts from the origin until the last operator applied. Links and threats solving techniques have been applied in order to obtain a valid incomplete order solution.

Once the new links are computed, this layer provides the new links to the scheduler module. Given that there could be new precedence constraints computed by the scheduler in case of resource conflicts, the scheduler could add new links.

The reasons to choose QPRODIGY are manyfold. Among them, we can highlight the possibility of defining and handling different quality metrics (Borrajo et al., 2001), reasoning about multiple criteria (Aler & Borrajo, 2002), flexibility to easily define new behaviours, capability to represent and reason about numeric variables, definition of constrains on variable values in preconditions of operators, explicit definition of control knowledge as well as its automatic acquisition through different machine learning techniques (Veloso et al., 1995), and explicit rationale of each problem solving episode through the search tree.

5.2. IPSS-S

IPSS-S is composed of the Ground-CSP and the Meta-CSP layers. The Ground-CSP layer represents the set of temporal constraints as a Simple Temporal Problem (Dechter, Meiri, & Pearl, 1991). In particular, significant events, as start/end time of operators are represented as temporal variable tp_i called time points.

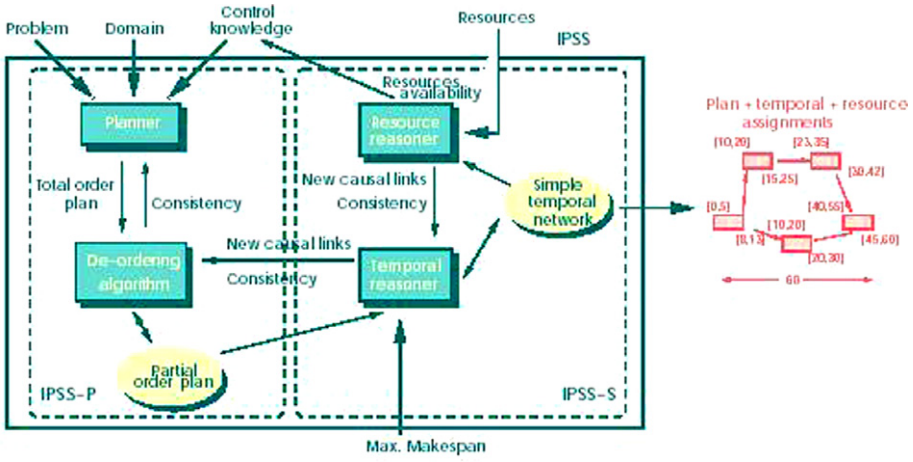

Fig. 6. IPSS architecture.

The time complexity for add (included the consistency checking) or remove a constraint is O(ne), where n is the current number of time points and e is the number of constraints. In fact we have used a Bellman-Ford based implementation.

In the Meta-CSP layer, resource constraints are superimposed, giving rise to a set of resource conflicts that are solved with a number of sequencing decisions (Cesta, Oddi, & Smith, 1998; Cesta, Oddi, & Smith, 2002).

To handle cumulative resources, special attention has to be paid to the Meta-CSP due to exponential time to complete the computation. To address this problem two main ideas have been integrated:

1. After propagation in the Ground-CSP, the earliest start time of all temporal variables are extracted; this is used as a basis for computing the Minimal Critical Set (MCS) (i.e., the Meta-CSP).
2. A polynomial MCS sampling method is introduced to extract a subset of MCSs and overcomes the exponential worst case of complete enumeration.

If the Meta-CSP has no assigned variables, there are no remaining resource conflicts and a solution has been found. But if the set of possible orderings of a conflict is empty then there is no feasible solution.

One of our goals is to minimise the makespan, at the same time that we compute a valid plan. If we use the feedback from the Meta-CSP, we can aim at using efficiently all the available resources. As a first approach to using such feedback, we have implemented a resource (load) balancing heuristic in the form of *domain dependent* control rules to choose resource bindings that minimise the makespan and at the same time avoid resource conflicts. When IPSS has to make a decision on which bindings should assign to an operator variables, it can use a set of control rules for making this decision. These control rules consult the Meta-CSP module through a condition in their left-hand side to know which is the resource less used in that time point.

Fig. 7 shows an example of a control rule for the *Build-Cable* operator in the TI domain. The **resource-less-used-p** meta-predicate implements the consult to the Meta-CSP. It binds the worker (⟨ww⟩ variable) that should build the cable with the one that the Meta-CSP has decided is less used. As the results on Section 7 show, these control rules can greatly improve the makespan.

6. An example: installing a new telephone line

A domestic or a company customer contacts a telecom company customer service (over the phone, in a shop or via the Internet). Let us say Mary Thompson contacts that company to ask for a new telephone line (ADSL, normal line or enterprise). At this point the business process starts (if the user agrees to the terms and conditions). The customer details are needed to see if she is an existing customer and already has a line with them (in that case, a discount will be applied to the second line). If the customer is asking a line for the first time, a spare pair of wires must be available from the house to make a connection from the distribution point (DP, e.g., telegraph pole). If no pair is available, then a new cable must be built (and the customer must be notified that the delivery date will be delayed).

Afterwards, it is necessary to check that there is a spare line card available in the exchange (in that case it is reserved/allocated). If none is available, an installation must be arranged. Installation involves making the connection at the DP (connecting a drop wire to the pair of wires that lead back to the exchange). Then, someone must:

- Contact the customer to arrange a visit to the house to fit new network terminating equipment (NTE, that is, the box on the wall that the phone is plugged into);
- Arrange for an engineer to turn up on the right day/time to test the line end to end and install the NTE;
- Allocate a telephone number to the new line and configure the exchange;
- Update the exchange, line plant and customer records;
- And of course, check with customer that he/she is happy with the service.

In this process there are several activities to perform: build a cable (BUILD-CABLE), set the spare pair of wires available (SET-SPARE-AVAILABLE), check if there is a spare line card available in the exchange (CHECK-LINE), contact the customer to fit the NTE (FIT-NTE), test the line (TEST-LINE), allocate a telephone number (ALLOCATE-NUMBER) and update customer data (UPDATE-DATA). These activities can be performed by any of the workers in the company, each one with different levels of knowledge: more expertise they are less time is required to execute the activity.

As an example of a possible behaviour of the BUILD-CABLE activity, the rule describing its behaviour is shown in Fig. 8. Fig. 9 shows its corresponding IPSS operator.

```
(Control-Rule Bindings-BUILD_CABLE
(IF (and (current-goal (built-cable-for <spare>))
         (current-operator  BUILD_CABLE)
         (resource-less-used-p <ww>)))
(THEN select bindings ((<w0> .  <ww>) (<sp0> . <spare>) )))
```

Fig. 7. A control rule to bind resources in the TI domain.

```
Rule Bill with properties ruleset behaviour
   If Exists a SpareLine of type MetaClassElement named var sp0
      with IsCable = ( FALSE )
      with Available = ( TRUE )
      with At-spare = ( Exists Zone of type MetaClassElement named var z0 )
   and
      Exists a Worker of type resource named var w0
         assigning to var z0 its At-worker
   Then Modify object sp0
      with IsCable is ( TRUE )
```

Fig. 8. Example of a BUILD-CABLE activity rule in SHAMASH.

```
(OPERATOR BUILD_CABLE
  (params <sp0>)
   (preconds
     ((<w0> WORKER)
      (<sp0> SPARELINE)
      (<z0> ZONE))
     (and (IsCable <sp0> FALSE)
          (Available <sp0> TRUE)
          (At-spare <sp0> <z0>)
          (At-worker <w0> <z0>)))
   (effects
      ()
      ((del (IsCable <sp0> FALSE))
       (add (IsCable <sp0> TRUE))))
   (resources
      ((<req-from-worker> (and CAPACITY (resource-from-pred
                                          (Build_Cable-from-worker
                                                    <w0> <req-from-worker>)))))
      ((<w0> <req-from-worker>)))
   (constraints
      ((<duration> (and DURATION (constraint-from-pred
                                       (Build_Cable-duration  <w0> <duration>)))))
      ((DURATION <duration>)))
   (costs
      ((<unit> LEVEL (cost-from-pred (Expert_Level <w0> <unit>))))
      ((EXPERT_LEVEL <unit>))))
```

Fig. 9. IPSS operator corresponding to the BUILD-CABLE rule of Fig. 8.

From the rule, the system will automatically translate the *params, preconds* and *effects* fields.

The precondition of the BUILD_CABLE activity checks if there is an element of the type Element class (represented by the variable sp0) with attribute values such that: there is no cable built for the corresponding line `(IsCable)`, the spare line card is available `(Available)` and the spare and worker must be in the same zone `(At-spare)`, `(At-worker)`. As a post-condition, the execution of the activity will cause that the cable has been built.

The variable ⟨sp0⟩ is used to represent the element instantiated by IPSS among the spare cards that the problem might have (spare-3, in the plan instantiated in the last section).

The *resource-from-pred* function binds the capacity of the resource required to perform that activity. The function checks that the value is not higher than that global capacity of the resource. That information is obtained from the resource hierarchy of SHAMASH, where we have the information of the workers assigned to each unit. In the example of Fig. 10 the Engineer Unit with id 0 has just one resource assigned (capacity 1) and the Engineer Unit with id 1 has two workers assigned (total capacity is equal to 2).

The *cost-from-pred* function generates a list of values to be possible bindings for the corresponding variable by using the information of the current state. This function is used with infinite-type (numbers), as is the case of the

```
(state
    (and (Availability Tom TRUE)
         (Cost Engineer-Unit-Expert-0 798Euro)
         (Cost Engineer-Unit-Expert-1 548Euro)
         (Id Engineer-Unit-Expert-0 123A)
         (Id Engineer-Unit-Expert-1 123B)
         (Assignedto Engineer-Unit-0 Tom)
         (Assignedto Engineer-Unit-1 Mike)
         (Assignedto Engineer-Unit-1 Rose)
         (ClientName order-Client123 SmithD.)
         (Build_Cable-duration Engineer-Unit-Expert-1 (5 9))
         (Expert_Level Engineer-Unit-Expert-1 5)
         (Build_Cable-duration Engineer-Unit-Expert-0 (2 4))
         (Expert_Level Engineer-Unit-Expert-0 7)
         (Allocated Line3 FALSE)
         (Available Spare3 FALSE)
                              ...  ))
    (goal (Allocated Line3 TRUE))
```

Fig. 10. IPSS problem for the Installing a Line example.

integer type, and in this case generates all the numeric values of the predicate `Expert_Level` in the problem. Depending on the level of expertise (being 10 the higher) the duration of the activity will vary and so does the cost.

The duration value is specified in the *constraints* field by the *constraint-from-pred* function. This function allows specifying the minimum and maximum value of the duration and let the schedule module of IPSS reason about it. Also, through this function we can specify temporal windows in the operator start time.

With respect to the problem definition, Fig. 10 shows the initial conditions generated automatically from the organisation information. This includes issues such as who works in each unit (`Assigned to Engineer-Unit-Expert-0 Tom`), the cost per hour of the employees belonging to a given category (`Cost Engineer-Unit-Expert-0 798Euro`) and other information related to the details of the line. The Figure also shows the goals that IPSS needs to accomplish. In this case, the only goal is to allocate the line with identifier number 3. With these conditions IPSS generates the plan described before, with resources, units and roles in that process.

7. Experiments

In this section the different IPSS configurations are explained and the comparison against some state of the art planners will be presented. We have defined two sets of experiments:

- Quantitative experiments for makespan minimisation, without time windows.
- Experiments with time windows.

The problems represented in the second type sets of experiments play an important role in BPR. Most of the processes that are defined in a company have a deadline: activities have to be performed before or inside a temporal window. We show in the next subsection how IPSS can handle it when most of the state of the art cannot.

7.1. IPSS configurations used

Table 2 shows the name of each IPSS configuration used and a brief description.

- IPSS bases its search in the QPRODIGY search integrated with the algorithm that converts incrementally the partial TO plan into a partial PO plan. Then, the *Ground*-CSP layer checks its consistency and provides the temporal information back to the search. Finally, the *Meta*-CSP layer checks resource conflicts.
- IPSS-R bases its search in the QPRODIGY search integrated with the three layers working together and a load balancing heuristic that controls the resource usage. First, the TO plan generated into a PO plan by the *Deorder-layer*. Then, the other two layers work in parallel: the Temporal Network that checks the temporal consistency and the resource layer that checks resource consistency, provides temporal and resource information back to the search that helps on next decisions to be made. The heuristic helps on deciding the resource to perform each activity.

Table 2
The different IPSS configurations used

Name	Description
IPSS	Temporal planner with three layers: *Meta*-CSP, *Ground*-CSP and *Deorder-layer*
IPSS-R	Temporal planner with a load balancing heuristic and three separate layers: *Meta*-CSP, *Ground*-CSP and *Deorder-layer*

The following section describes the results obtained by IPSS and other state of the art planners.

7.2. Telephone installation (TI) domain

In order to exploit parallelism, we will consider that allocating a line can be done in parallel to any other activity. Also, to reduce the number of activities to be accomplished, we have only considered the operators BUILD-CABLE, SET-SPARE-AVAILABLE and CHECK-LINE. The rest of operations must be performed in sequence.

The modeling of the domain does not consider if the worker is or not occupied doing something else. So, apart from planners that provide serial solutions, as, for example, SERISTAR (that belongs to the HSP family Bonet & Geffner, 2001), FF (Hoffmann, 2002) or QPRODIGY (Borrajo et al., 2001), the solution given by other planners as LPG (Gerevini & Dimopoulos, 2002) or MIPS (Edelkamp & Helmert, 2000; Wah & Chen, 2003) is not correct because in the modelization we do not consider the resource occupability so they can assign different actions to the same worker at the same time.

In order to compare not only TO planners against our approach, we have also coded this domain having in mind the availability of the chosen worker for performing the corresponding action. In this case, each action (operator) requires the worker not to be busy on performing some actions. After the execution of the action, the agent (worker) is freed by a dummy action (the *free-worker* action). This forces the planners that reason about parallel actions not to consider parallelizing two actions that require the same worker. We have called this domain the TI_OCCU domain.

We have randomly generated 11 sets of problems, with 3 problems in each set, increasing the number of goals (1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 50). Then, for each set we have increased the number of workers (2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50). The total time given to solve each problem (time bound) was 120 s.

The set of problems for the TI_OCCU domain is exactly the same as in the TI domain with the particularity that in all the problems the availability of the worker must be explicitly set to *not-occupied* (that is, in the initial conditions we add the predicate (*not-occupied* ⟨*worker*⟩) for each ⟨worker⟩ in the problem), so we have added an extra operator to free the worker with duration 0.

Given that LPG bases its search in a non-deterministic local search, we have run five times each problem, and considered the best solution, the median solution and the worst solution. This is represented in the tables by LPG-MIN, LPG-MED (it will be considered the baseline to compare the results) and LPG-MAX, respectively. Tables 3 and 4 show the number of solved problems by each planner considering the number of workers[1] and goals.[2] As it can be seen, IPSS-R, FF and LPG are fast and efficient planners that solve all problems. Then, IPSS almost solves all the problems and the worst performance is by the MIPS planner.

Fig. 11 shows the makespan obtained for all the planners in problems solved by all. We have not included MIPS in the graphs due to the low percentage of problems solved, but if considered, it obtains the worst results in makespan and time although it gets the plans with less number of operators than LPG. The best results are obtained by IPSS-R and LPG-MIN. Then, LPG-MED, LPG-MAX, IPSS and the worse makespan is, as expected, by FF because it is a TO planner.

With regards to the number of operators in the solutions, FF finds the shortest solutions, followed by the IPSS configurations and then LPG runs (note that in this configuration we have considered the *free-worker* operator so the number of operators must be higher than in the others). Fig. 12 shows these results.

With respect to the execution time, FF is faster than any of the LPG runs or the IPSS configurations as Fig. 13 shows. Also, IPSS runs in COMMONLISP while the rest of planners run in C, that usually makes the execution much faster. But, time in workflow system it is not critical as it could be in others domains as satellites or robotics where obtaining a plan fast is needed in order to make decisions.

An important feature of our system that the rest of the IPC state of the art planners do not incorporate, is the ability to impose temporal windows to the goals. This is a very critical problem in *workflow system* because in most of the cases the activities that have to be performed would not be valid if they are not executed in a specific temporal horizon. For example, it is common in telecommunication companies to agree on installing a telephone line in 2 days, so plans whose execution takes more time are not valid.

To test this type of problems we have not generated any set of problems but we show how IPSS works through an example. Because we could set the temporal windows in the operators or in the goals, we have decided to impose the temporal constraints on the goals for clarity.

Let us consider a problem with 6 workers (W1, W2, ..., W6) that have to perform 5 goals: checking five spare line cards (SP0, ..., SP4). The star-shaped spare lines also need to set the spare pair of wires. Fig. 14 shows the disposition of the spare lines, the workers and the communication among the different zones.

For simplicity all the activities have a unit duration. We will impose temporal windows in four of the five goals as Fig. 15 shows. The first goal has to start after the time instant 3, the second goal should start before the time instant 4, the third goal should be contained in the interval (2 5) and the forth goal should start at the time instant 1. The `(check_line spare4)` goal can start at anytime.

The solution that IPSS finds to the problem of Fig. 14 is shown in Fig. 16, that achieves a solution fulfilling all goal constraints.

[1] W2 with two workers, W3 with 3 workers and so on.

[2] G1 stands for problems with one goal, G2 with two goals and so on.

Table 3
Number of problems solved by each planner in the TI domain from a total of 429 problems given 120 s considering the no. of workers

System	W2	W3	W4	W5	W6	W7	W8	W9	W10	W20	W30	W40	W50	Solved	%
LPG	33	33	33	33	33	33	33	33	33	33	33	33	33	429	100
FF	33	33	33	33	33	33	33	33	33	33	33	33	33	429	100
IPSS-R	33	33	33	33	33	33	33	33	33	33	33	33	33	429	100
IPSS	29	32	33	33	33	33	33	33	33	33	33	33	33	424	98
MIPS	24	27	26	25	25	25	24	24	23	9	0	0	0	232	54

Table 4
Number of problems solved by each planner in the TI domain from a total of 429 problems given 120 s considering the no. of goals

System	G1	G2	G3	G4	G5	G10	G15	G20	G30	G40	G50	Solved	%
LPG	39	39	39	39	39	39	39	39	39	39	39	429	100
FF	39	39	39	39	39	39	39	39	39	39	39	429	100
IPSS-R	39	39	39	39	39	39	39	39	39	39	39	429	100
IPSS	39	39	39	39	39	39	39	39	39	39	35	424	98
MIPS	30	29	29	30	29	27	23	25	10	0	0	232	54

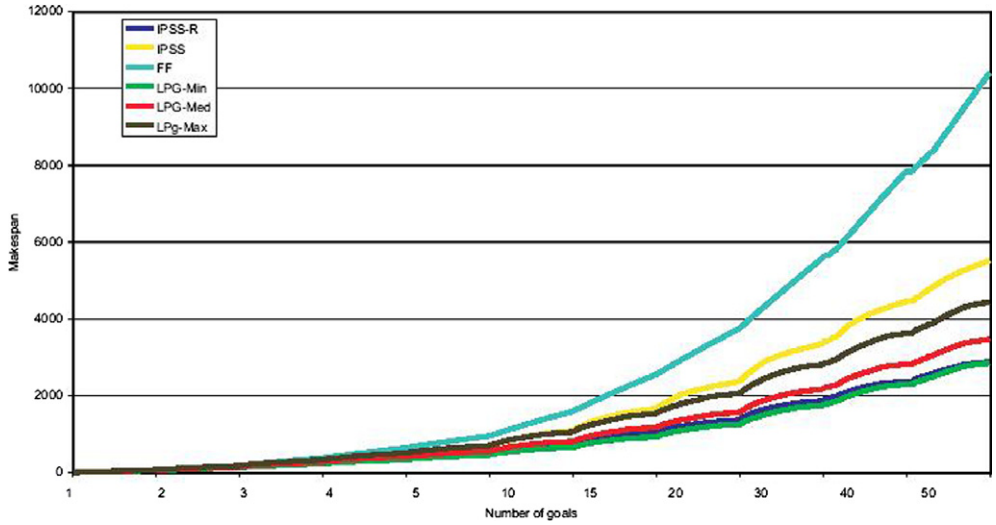

Fig. 11. Makespan for FF, LPG, IPSS and IPSS-R in TI.

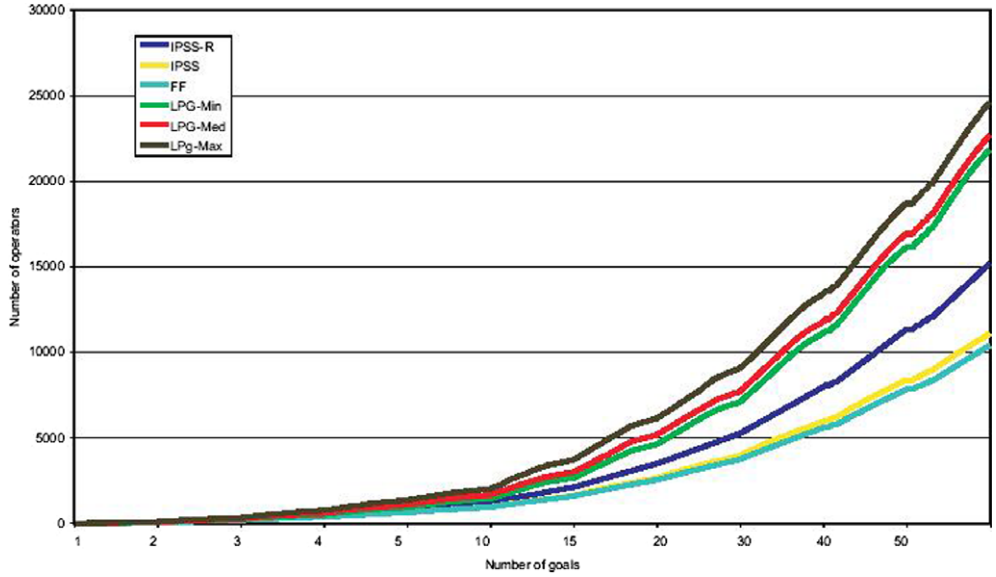

Fig. 12. Number of operators by FF, LPG, IPSS and IPSS-R in TI.

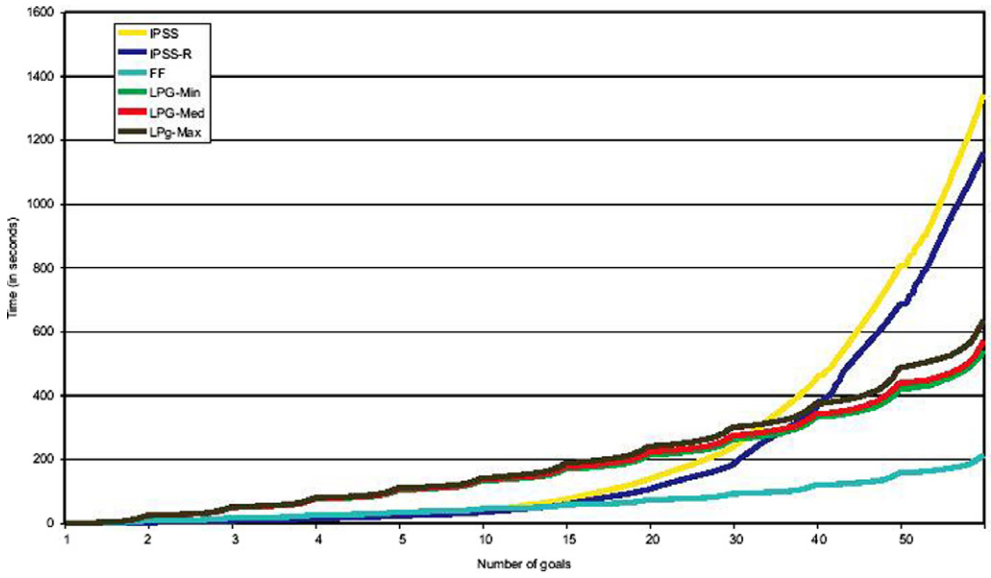

Fig. 13. Time for FF, LPG, IPSS and IPSS-R in TI.

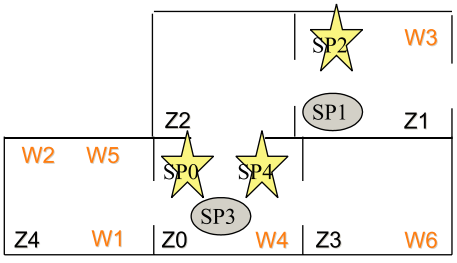

Fig. 14. Initial conditions for the temporal windows example.

```
((check_line spare0) after 3)
((check_line spare1) before 4)
((check_line spare2) within (2 5))
((check_line spare3) at 1)
```

Fig. 15. Temporal windows imposed to the goals.

8. Related work

Several research groups have integrated AI techniques with workflow management systems (support the modeling and the enactment phase of the software engineering process). Some have applied planning tools in the enactment

```
Start-Time    End-Time      Id   Operador
         0           0       1   (s-op)
         0           1      13   (goto_zone worker1 zone4 zone0)
         0           1       8   (goto_zone worker0 zone3 zone0)
         0           1       4   (goto_zone worker5 zone4 zone0)
         1           2      16   (set_spare_available worker1 spare0 zone0)
         1           2      14   (set_spare_available worker1 spare4 zone0)
         1           2       9   (goto_zone worker0 zone0 zone2)
         1           2       5   (goto_zone worker5 zone0 zone2)
         1           2       3   (check_line worker4 spare3 zone0)
         2           3      15   (check_line worker1 spare4 zone0)
         2           3      10   (goto_zone worker0 zone2 zone1)
         2           3       6   (goto_zone worker5 zone2 zone1)
         3           4      11   (set_spare_available worker0 spare2 zone1)
         3           4       7   (check_line worker5 spare1 zone1)
         4           5      17   (check_line worker1 spare0 zone0)
         4           5      12   (check_line worker0 spare2 zone1)
         5           5       2   (f-op)

#<THIS result: 0.493925 secs, 64 nodes, 1 sol, 15 this-cost, 5 this-makespan>
```

Fig. 16. The solution to the temporal windows example.

phase and very few have integrated planning techniques with BPR tools during the modeling phase.

The SWIM system (Berry & Drabble, 2000) integrates process instantiation, task allocation, execution, monitoring and optimisation for improvement in the schedule. There is a Process Library that provides the correct process definition to the Dynamic Process Manager. New processes can also be added to this library.

The MILOS project (Goldmann, Münch, & Holz, 2000) is a process modeling and enactment approach that provides project planning and management support like resource allocation and time scheduling for tasks in the project. The MILOS workflow engine allows the model and plan to be changed during project enactment and provides support for process restarts whenever necessary. The library of process models contains descriptions of best practices in software development for a given company. The project manager selects processes and methods from the library creating an initial project plan. The plan is uploaded to standard project management tools as MS-Project.

The TBPM (Jarvis et al., 2000) project is based on work carried out in the Enterprise project (Stader, 1996) and centres around an intelligent workflow engine that includes an interactive process planner. The planner uses AI techniques based on O-Plan (Currie & Tate, 1991) to assist in task planning, while permitting the user to participate in planning decisions. An agent-based architecture supports the execution and co-ordination of the planned process among multiple participants and distributes processes across computer networks. The user is able to plan a task by assembling fragments and then to refine it, generating a hierarchical model of the process to be performed. For more flexibility, the user is able to edit the process model before and during its enactment, in order to specialise it.

In all these systems the modeling phase is based on a process library, but there is no automatic generation as we have outlined in this integration of SHAMASH-IPSS. Each time new model/templates are created, there is no need to create a library; the planner will generate the correct one automatically.

9. Conclusions and future work

In this paper, the representation problem for workflow domains has been faced. We have used the SHAMASH workflow modeling tool generated in an EU Esprit Project. To provide a frame of reference between Workflow Systems and AI planners, the stages that both areas have in common has been described following the classification given by some researchers (Kearney & Borrajo, 2000; PLANET). The translation between both fields has had in mind the different semantics of BPR and AI planning concepts as well as their similarities.

We have described the advantages of using IPSS for modeling processes in SHAMASH, a Knowledge Based System that uses an object oriented and rule-based approach. The SHAMASH rules and objects are translated into types and operators to produce a plan that correspond to a process instance (tasks dependency). Each plan will vary depending on the resources available, elements that flow through the process and the different tasks that the organisations can introduce to adapt their processes to the market changes. With this approach the models generated are automatically validated avoiding inconsistencies in linking activities and saving time to the user.

Finally, we also wanted to study the issues that AI planners can gain with this approach. Generally, to specify the domain theory, a deep understanding of the way AI planners work and its terminology is needed. However, if we use a tool like SHAMASH, the description language is closer to the user and allows an automatic verification of the syntax through a friendly interface. In this integration we could have also used the PDDL2.2 language instead. Then, any planner that supports PDDL2.2 could exploit the advantages of using a descriptive language as the one used in SHAMASH. Although some planners of the competition cannot reason, for instance, about temporal windows or deadlines, IPSS does. And this is usually a strong requirement that drove us to build a system with the properties that IPSS presents.

Acknowledgements

The SHAMASH project has being carried out in the course of the R&D project funded by the Esprit Program of the Commission of the European Communities as project number 25491. A complementary grant was given by the Spanish research commission, CICYT, under project number TIC98-1847-CE. We thank the partners of this project, who have originated and contributed to the ideas reported: UF (Unión Fenosa), SAGE (Software AG España), SEMA GROUP sae, UC3M (Universidad Carlos III de Madrid), WIP (Wirstchaft und infrastruktur & Co Planungs KG), and EDP (Electricidade de Portugal). We would specially like to thank all the UC3M team, the PLANET people and Paul Kearney (BT). Through talks with him we have outlined many ideas. This work has also been partially funded by grant MCyT TIC2002-04146-C05-05 and the UAH project PI2005/084.

References

Aler, R., & Borrajo, D. (2002). Learning single-criteria control knowledge for multi-criteria planning. In *Proceedings of the AIPS-02 workshop on planning and scheduling with multiple criteria* (pp. 35–40).

Aler, R., Borrajo, D., Camacho, D., & Sierra-Alonso, A. (2002). A knowledge-based approach for business process reengineering, SHAMASH. *Knowledge Based Systems, 15*(8), 473–483.

Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence, 23*, 123–154.

Bäckström, C. (1998). Computational aspects of reordering plans. *Journal of Artificial Intelligence Research, 9*, 99–137.

Berry, P. M., & Drabble, B. (2000). SWIM: an AI-based system for organizational management. In *Proceedings of the 2nd NASA international workshop on planning and scheduling for space*, San Francisco, California.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence, 129*(1–2), 5–33.

Borrajo, D., Vegas, S., & Veloso, M. (2001). Quality-based learning for planning. In *Working notes of the IJCAI'01 workshop on planning with resources*.

Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., et al., 1992. *PRODIGY4.0: The manual and tutorial*. Tech. Rep. CMU-CS-92-150, Department of Computer Science.

Cesta, A., Oddi, A., & Smith, S. (1998). Profile based algorithms to solve multiple capacitated metric scheduling problems. In *Proceedings of the fourth international conference on artificial intelligence planning systems* (*AIPS-98*).

Cesta, A., Oddi, A., & Smith, S. (2002). A constrained-based method for project scheduling with time windows. *Journal of Heuristics, 8*, 109–136.

Cesta, A., Pecora, F., & Rasconi, R. (2004). Biasing the structure of scheduling problems through classical planners. In *Proceedings of the workshop on integrating planning into scheduling*.

Currie, K., & Tate, A. (1991). O-plan: the open planning architecture. *Artificial Intelligence, 52*, 49–86.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence, 49*, 61–95.

Do, M. B., & Kambhampati, S. (2003). SAPA: a scalable multi-objective metric temporal planner. *Journal of Artificial Intelligence Research, 20*, 155–194.

Edelkamp, S., & Helmert, M. (2000). On the Implementation of MIPS. In *AIPS workshop on model-theoretic approaches to planning*.

Edelkamp, S., & Hoffmann, J. (2004). PDDL2.2: *the language for the classical part of the 4th international planning competition*. Tech. Rep. Technical Report No. 195, Institut fur Informatik.

Fikes, R., & Nilsson, N. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence, 2*, 189–208.

Gerevini, A., & Dimopoulos, Y. (2002). Temporal planning through mixed integer programming. In *Proceedings of the AIPS'02 workshop on planning for temporal domains*.

Ghallab, M., & Laruelle, H. (1994). Representation and control in IxTeT, a temporal planner. In *Proceedings of the second international conference on AI planning systems* (*AIPS-94*).

Goldmann, S., Münch, J., & Holz, H. (2000). Distributed process planning support with MILOS. *International Journal of Software Engineering and Knowledge Engineering, 10*(4), 511–525.

Hammer, M., & Champy, J. (1993). Reengineering the corporation. In *Reengineering the corporation*. New York: Harper Business Press.

Hannebaeur, M. (1999). From formal workflow models to intelligent agents. In *AAAI-99 workshop on agent-based systems in the business context*. Brian Drabble and Peter Jarvis Cochairs.

Hoffmann, J. (2002). The metric-FF planning system: translating ignoring delete lists to numerical state variables. *Journal of Artificial Intelligence Research*.

Jarvis, P., Moore, J., Chung, P., Mcbriar, I., Stader, J., Ravinranathan, M. et al. (2000). Applying intelligent workflow management in the chemicals industries. In L. Fisher (Ed.), *The workflow handbook 2001*. Published in association with the Workflow Management Coalition (WfMC).

Kearney, P., & Borrajo, D. (2000). An R&D agenda for AI planning applied to workflow. In *Proceedings of the eBusiness and eWork conference 2000*.

Kuter, U., Nau, D., Pistore, M., & Traverso, P. (2005). The Yoyo planner: going where hierarchical task-network planning meets with symbolic model checking. In *Proceedings of the 15th international conference on automated planning and scheduling* (*ICAPS'05*).

Leymann, F., & Roller, D. (1994). Business process management with FlowMark. In *Proceedings of the IEEE computer society international conference*.

Lydiard, T., Jarvis, P., & Drabble, B. (1999). Realizing real commercial benefits from workflow: a report from the trenches. In *AAAI-99 workshop on agent-based systems in the business context*. Brian Drabble and Peter Jarvis Cochairs.

Medina-Mora, R., Winograd, T., & Flores, P. (1993). Action workflow as the enterprise integration technology. *Bulletin of the Technical Committee on Data Engineering. IEEE Computer Society, 6*, 2.

Mohan, C. (1997). Recent trends in workflow management products, standards and research. In *Proceedings NATO advanced study institute* (ASI) *on workflow management systems and interoperability*.

Myers, K. L. & Berry, P. M. (1999). At the boundary of workflow and AI. In *AAAI-99 workshop on agent-based systems in the business context*. Brian Drabble and Peter Jarvis Cochairs.

PLANET. Network home page: European Network of Excellence in AI planning. Available from http://planet.dfki.de/index.html.

PLANET. Workflow management TCU road map. Available from http://scalab.uc3m.es/dborrajo/planet/wm-tcu/.

Purvis, M., Purvis, M., & Lemalu, S. (2001). A framework for distributed workflow systems. In *Proceedings of the 34th annual Hawaii international conference on system sciences* (*HICSS-34*). IEEE Computer Society.

R-Moreno, M. D. (2003). *Representing and Planning tasks with time and resources*. PhD thesis, Universidad de Alcala, Spain.

R-Moreno, M. D., & Kearney, P. (2002). Integrating AI planning with workflow management system. *International Journal of Knowledge-Based Systems, 15*, 285–291.

R-Moreno, M. D., Oddi, A., Borrajo, D., Cesta, A., & Meziat, D. (2004a). Planning and scheduling for workflow domains. In *The ECAI'04 ''planning and scheduling: Bridging theory to practice'' workshop*.

R-Moreno, M. D., Oddi, A., Borrajo, D., Cesta, A., & Meziat, D. (2004b). IPSS: integrating hybrid reasoners for planning and scheduling. In *Proceedings of the 16th European conference on artificial intelligence*, *ECAI04*.

Srivastava, B., Kambhampati, R., & DO, M. B. (2001). Planning the project management way: efficient planning by effective integration of causal and resource reasoning in RealPlan. *Artificial Intelligence, 131*, 73–134.

Stader, J. (1996). Results of the enterprise project. In *Proceedings 16th international conference of the British computer society specialist group on expert systems*, Cambridge, UK.

Tate, A., Drabble, B., & Kirby, R. K. (1994). O-Plan2: An open architecture for command, planning, and control. Morgan Kaufman.

Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: the PRODIGY architecture. *Journal of Experimental and Theoretical, AI 7*, 81–120.

Wah, B. W. & Chen, Y. (2003). Partitioning of temporal planning problems in mixed space using the theory of extended saddle points. In *15th IEEE international conference on tools with artificial intelligence* (*ICTAI'03*).

Workflow Management Coalition. Available from http://www.wfmc.org/.

Yang, Q. (1997). Intelligent in planning. *A decomposition and abstraction based approach*. Springer.