

WORKFLOW SUPPORT FOR SCHEDULING IN SURGICAL CARE PROCESSES¹

Ouyang, Chun, Faculty of Science and Technology, Queensland University of Technology, 2 George St, Brisbane QLD 4000, Australia, c.ouyang@qut.edu.au

Wynn, Moe Thandar, Faculty of Science and Technology, Queensland University of Technology, 2 George St, Brisbane QLD 4000, Australia, m.wynn@qut.edu.au

Kuhr, Jan-Christian, GECKO mbH, Deutsche-Med-Platz 2, 18057 Rostock, Germany, jan-christian.kuhr@gecko.de

Adams, Michael, Faculty of Science and Technology, Queensland University of Technology, 2 George St, Brisbane QLD 4000, Australia, mj.adams@qut.edu.au

Becker, Thomas, GECKO mbH, Deutsche-Med-Platz 2, 18057 Rostock, Germany, thomas.becker@gecko.de

ter Hofstede, Arthur H.M., ¹Faculty of Science and Technology, Queensland University of Technology, 2 George St, Brisbane QLD 4000, Australia, a.terhofstede@qut.edu.au, and ²Eindhoven University of Technology, Eindhoven, The Netherlands

Fidge, Colin J., Faculty of Science and Technology, Queensland University of Technology, 2 George St, Brisbane QLD 4000, Australia, c.fidge@qut.edu.au

Abstract

Real-world business processes rely on the availability of scarce, shared resources, both human and non-human. Current workflow management systems support allocation of individual human resources to tasks but lack support for the full range of resource types used in practice, and the inevitable constraints on their availability and applicability. Based on past experience with resource-intensive workflow applications, we derive generic requirements for a workflow system which can use its knowledge of resource capabilities and availability to help create feasible task schedules. We then define the necessary architecture for implementing such a system and demonstrate its practicality through a proof-of-concept implementation. This work is presented in the context of a real-life surgical care process observed in a number of German hospitals.

Keywords: Surgical care processes, resource coordination, scheduling, schedule-driven processes, workflow management systems, business process management.

¹ This research was funded in part by Australian Research Council grant DP0773012, *Rapidly Locating Items in Distribution Networks with Process-Driven Nodes*, and the German Federal Ministry of Education and Research under grant 01IS099009. The research was carried out in partnership with Gecko as part of the PERIKLES project (www.perikles.org). We are grateful to the many clinicians who have supported us in investigating and evaluating the domain of perioperative processes. We also wish to thank Erik Nieswandt of Gecko for his substantial contribution to the project.

Introduction

Apart from their humanitarian role, in order to ensure their ongoing viability hospitals must generate revenue through surgical procedures which serve as key value-adding processes. Surgical careflows are usually well-structured, resource-centric and strongly depend on accurate coordination and efficient scheduling. They typically comprise a *pre-admission* and a *peri-operative* stage, where the latter refers to the part of the patient's clinical pathway that begins with admission to the hospital and ends with post-operative transfer to a ward. From a cost- and resource-oriented perspective, the peri-operative part of the process plays a key role. In many hospitals this phase is carried out in a central operating facility, characterised by a high density of both *human resources* (e.g., surgeons, scrub nurses) and *non-human resources* (e.g., operating rooms, surgical instrument sets).

An operating facility usually runs multiple operating rooms concurrently to enable a high throughput of surgical cases. Technical resources that are expensive and scarce are also shared among different cases. In order to use these resources as efficiently as possible, high-quality *scheduling* of all resource-centric activities that are part of the peri-operative stage is crucial. Ideally, scheduling takes place well in advance of the day of surgery, i.e., during the pre-admission part of the process, while *re-scheduling* involving *coordination of resources* on-the-fly often becomes necessary as circumstances change. The quality of schedules thus becomes a key factor in determining overall process performance, and involves knowledge of resource availability, an understanding of how resources may be deployed in real time, and an up-to-date view of the status of each process.

Scheduling and resource coordination are important challenges in the healthcare domain and the issues they raise have been studied for many years (Cardoen et al., 2008; Cardoen et al., 2009; Cayirli and Veral, 2003; Plasters et al., 2003). More broadly, the literature on scheduling and resource planning can be found in operational management, organisation science (Crown, 1997), artificial intelligence, etc and the techniques vary from GANTT charts, operations research, to agent-based simulation (Hutzschenreuter et al., 2008). However, most of the existing approaches focus on resource dependencies in terms of individual activities, or improvement to scheduling algorithms alone. This has led to the lack of a systematic support throughout an entire process lifecycle. Scheduling and coordination of resources is usually carried out manually, or using tools that are not integrated with the organisation's workflow processes.

A Workflow Management System (WfMS) supports process execution by controlling the flow of work so that individual tasks are done in the right sequence and by appropriately qualified individuals (van der Aalst and van Hee, 2003). Furthermore, workflow systems can be used to perform real-time monitoring of a process' status and resource utilisation during process enactment. Hence, we observe that this information could be shared with a scheduling engine so that a valid initial schedule could be generated before a surgical process begins and then be maintained as the process is performed.

Recently, business process management researchers have started to explore the scheduling aspects of workflow modelling. There are two main research streams. One is related to time management in workflows, for example, Eder et al. (1999) discuss how to enforce timing constraints during workflow execution, and Combi and Pozzi (2004) propose a WfMS which can manage timing aspects of workflows via the adoption of a temporal database as part of the workflow system. The second stream of research focuses on the problem of scheduling tasks in workflows, e.g., several authors have presented algorithms for scheduling tasks or jobs (Bettini et al., 2002; Tramontina et al., 2004; Senkul and Toroshu, 2005), and R-Moreno et al. (2007) propose applying planning techniques from Artificial Intelligence to the scheduling of tasks in workflow domains.

Unlike this previous work, however, our research concerns extending current WfMSs with scheduling and resource coordination facilities, rather than making improvements to scheduling algorithms or integration of time management schemes. To this end, the most relevant work to our own is that by Mans et al. (2009) who propose an extension to a WfMS for calendar-based scheduling functionalities.

However, their research is limited to scheduling human resources in the context of the specific healthcare domain of medical consultation processes.

Our research aims *to extend current WfMSs with capabilities to support resource-intensive and schedule-driven business processes*. In this particular paper, we focus on surgical care processes in the healthcare domain. The main contributions of this paper can be summarised as follows:

- A detailed analysis of key requirements for a WfMS to support rich resource coordination (human and non-human) and scheduling (Section 3);
- An architecture design for extensions to a WfMS to support these requirements (Section 4);
- A proof-of-concept implementation of various system components (Section 5); and
- An illustration of how scheduling requirements in surgical care processes can be (partially) automated by workflow systems (Section 6).

A demonstrable implementation of the design is developed in the framework of a state-of-the-art WfMS called YAWL².

1 A Surgical Care Process

A surgical care process typically comprises a *pre-admission* and a *peri-operative* stage, where the latter refers to the part of the patient's clinical pathway that begins with admission to the hospital and ends with post-operative transfer to a ward. The scenario described below is based on information we collected from interviews with clinicians and on-site work shadowing in several German hospitals, and supplemented by a literature review of relevant clinical conferences.

Figure 1 shows a simplified process model that applies to patients undergoing a planned surgery³. The model covers the pre-admission (*Do Surgical Assessment to Admit Patient*) and peri-operative (*Register Procedure to Re-Admit at Ward*) part of the entire process. A task may be annotated with the resources *required* for carrying out the task and/or the resources *scheduled* for surgery, or the resources being *utilised* during the surgery. Below, we describe these in more detail.

Prior to admission to the hospital, in the outpatient's department the patient is assessed by a surgeon. During the assessment, it is possible to schedule certain resources that are required for surgery even at this initial stage. Schedule-relevant data input may include resources such as an operating room (OR) and surgeon, and the estimated duration of the resource allocation. After the surgical treatment has been agreed on, the patient is seen by an anaesthetist who will determine the proper anaesthesiological procedure. The anaesthetist may specify further resources that are needed (e.g., anaesthetist, nurse, post-operative capacity), thereby supplementing the scheduling input made by the surgeon.

The *perioperative* part of an inpatient's process also contains administrative tasks such as registering the procedure with the OR coordinator and re-admitting the patient to a ward after surgery, whereas the majority of the activities, e.g., *induction* and *operation*, do not require a human user to interact with a workflow system. However, these activities are resource-intensive and, as a result, not only do they require scheduling in advance but also their resource utilisation status can affect surgery scheduling and thus needs to be monitored.

Depending on the individual urgency level, the end-to-end surgical process may well extend over days and weeks where the duration of the *pre-admission* phase is typically long compared to the *perioperative* part. Multiple professional teams are involved in the careflow and each of them may enter schedule-relevant data to the patient's case. Therefore, resource requirements for a surgery evolve over time to become fully specified close to the date of the planned surgery.

² <http://www.yawlfoundation.org>

³ In reality, process models are much more complex and may depend on, e.g., the type of the patient case (in-patient, outpatient, emergency case, etc.) or the operating department (general surgery, traumatology, etc).

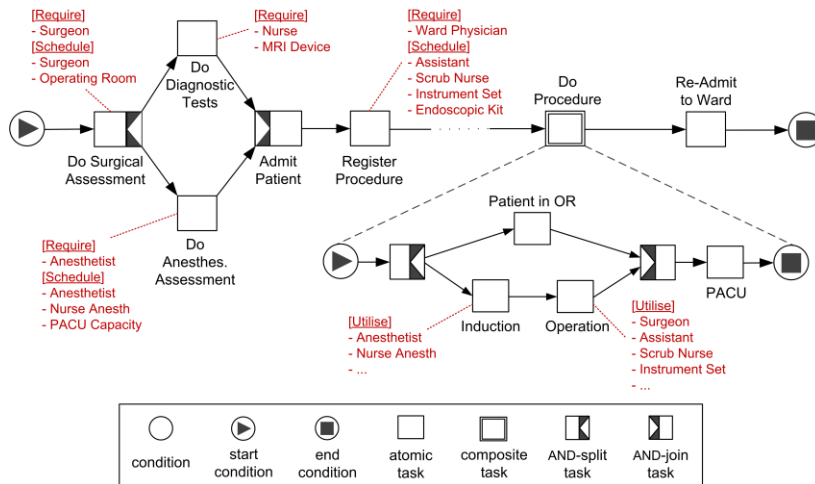


Figure 1. A simplified model of a surgical care process (using YAWL notation).

2 Requirements Analysis

The simplified surgical care scenario described in the previous section highlights many requirements for schedule-aware WfMSs. For instance, to produce efficient and optimised schedules, one must have knowledge of *all* the different types of resources that are required to perform the various activities within a business process. In addition, one must be aware of resource availability (e.g., whether a resource is available or unavailable for different reasons, such as being reserved, in use, broken, under repair, or absent), their associated resource lifecycle (e.g., after use certain tools may require their own maintenance, such as sterilisation), and any other related constraints (such as certain medical staff cannot work continuously for more than the pre-defined maximum number of hours). One must also have real-time access to resource information to make appropriate resource assignments according to the schedule. A WfMS should also support the need for adjustments to be made to scheduled activities and should provide well-managed resource utilisation data for post-execution process analysis.

In our earlier work (Ouyang et al., 2010) we proposed some preliminary data requirements for workflow-based resource coordination. In this paper, we extend that work by defining functional requirements for a schedule-aware WfMS. These requirements are defined at a conceptual level, i.e., independent of any specific WfMS. This allows us to apply the designed system, although motivated by surgical care processes, to any problem domain. It should be noted that our focus is on the necessary interactions between a WfMS and a scheduling system, whereas the actual operations within the scheduling system are beyond the scope of this paper.

Terms and concepts. A *resource* is an entity that is assigned to a task in a process and is required to perform work in order to complete the objective of the task. A *resource specification* contains the definition of human and non-human resources that are involved in the execution of each task in a process. *Resource constraints* are business rules that a resource should obey when being used, e.g., the number of hours a human resource can work without taking a break. A *schedule* is a collection of (schedulable) *activities* each associated with a certain *time period*, a set of allocated *resources*, and possibly a specific *location*. A *resource utilisation plan* is a simplified process specification containing only (schedulable) activities of the original process specification and annotated with individual case information. It serves as a basis for calculating schedules related to the process. Finally, we clarify the term *activity* used in the rest of the paper. An *activity* is a schedulable unit of work that is part of a process and usually requires one or more resources and needs to be scheduled before it is carried out. An activity comprises more than one task. A task is an executable unit of work in a process. Executing a task requires an interaction with the workflow engine and an instance of a task in a case is called a *work item* at runtime.

Requirement 1 (Rich Resource Specification) *A schedule-aware WfMS must support a comprehensive specification of resources that are involved in business processes, including a rich taxonomy of resources, resource data, resource availability, resource locations, and resource use constraints.*

A suitable taxonomy of resources includes, for example, human vs. non-human, consumable vs. non-consumable, or application vs. non-application. Resource data includes, for example, role and position (for humans), usage type (for non-human resources), quantity, capacity, size, or cost. Resource availability and location information is usually represented in resource calendars. Resource constraints include the resource's usage lifecycle and related rules for its application. Detailed data requirements can be found elsewhere (Ouyang et al., 2010).

Requirement 2 (Support for Tasks with Multiple Resources) *A schedule-aware WfMS must allow for the specification of one or more resources (both human and non-human) that are required by a task and how they are involved in carrying out the task. A schedule-aware WfMS must allow for the deployment of such tasks.*

Multiple resources may be involved in different ways in carrying out the task. A *primary resource* is the main human resource responsible for performing the task, and is usually required to interact with the workflow system. A *secondary resource* is an additional resource that is required to perform the task, and can be a human or a non-human resource. For example, a nurse conducting diagnostic tests for a patient may be referred to as the primary resource, while the MRI device used for the tests is a secondary resource (see Figure 1). When multiple primary resources are involved, a team approach (van der Aalst and Kumar, 2001) to the task's execution may be required.

Requirement 3 (Real-time Access to Resource Availability Status) *A schedule-aware WfMS must be able to determine each resource's availability during assignment of resources to tasks. It should also be able to update the resource's availability status upon task completion.*

Existing WfMSs assume that resources are always available. In reality, a task cannot be carried out if the resources assigned to it are not available. Hence, a schedule-aware WfMS must be able to verify the availability of a certain resource when assigning that resource to a task. That is, the WfMS should be aware of the current state of resources as well as the availability of resources in the future (e.g., from a resource calendar). Also, when tasks are executed, they utilise the assigned resources which then became unavailable. When the task is completed, the non-consumable resources are released, as are any consumables that remained unused. Thus, task execution changes the availability status of resources, and a schedule-aware WfMS must be able to interact with a resource calendar to update the resources' status upon task completion.

Requirement 4 (Real-time Access to Process Specification and Process Execution Status) *A schedule-aware WfMS must allow a scheduling system access to process specification (used to compose a resource utilisation plan) and process execution status (used to update the status of each activity in the resource utilisation plan) for calculating or maintaining an up-to-date schedule.*

It is necessary to access process specification in order to generate resource utilisation plans. Also, a utilisation plan needs to be updated during process execution. For example, if a (scheduled) activity has already started, it cannot be re-scheduled. To this end, it is necessary that the scheduling system can access process execution status at runtime.

Requirement 5 (Support for Tasks dedicated for Scheduling and for Resource Utilisation) *A schedule-aware WfMS must support the modelling and execution of tasks dedicated for scheduling of activities (namely scheduling tasks or S-Tasks for short) and tasks for allocation or de-allocation of resources upon the occurrences of scheduled activities (namely utilisation tasks or U-Tasks for short).*

Existing WfMSs support (normal) tasks that are allocated the required resources and are carried out by those resources at runtime (namely *R-Tasks*). In resource-centric and schedule-driven processes, there are activities (e.g. *induction, operation*) which are carried out without human users' interacting with a WfMS but require scheduling (via *S-Tasks*) in advance as well as monitoring of their resource

utilisation (via *U-Tasks*). Unlike R-Tasks, S-Tasks need to interact with resource data, resource calendar, resource constraints and rules to obtain the required information, and then the scheduling system to generate the schedule. U-Tasks need to interact with the scheduling system to update the resource utilisation plan, and then the WfMS for resource utilisation logging (see Requirement 8).

Requirement 6 (Support for Real-time User Input for Scheduling) *A schedule-aware WfMS must allow for real-time user input when carrying out a scheduling task, including the activity to be scheduled, start time, duration, resources required, and relation to other activities.*

Task-level resourcing requirements, i.e., the resources required by a task to be carried out, are defined in the process specification at design time. However, a schedule-aware WfMS must support resourcing requirements as part of the scheduling of activities. Apart from resourcing requirements, it must also allow the user to enter resource deployment information at runtime and provide up-to-date resource availability status. For example, in a surgical care process, this allows professional teams to enter data for the schedule of a planned surgery progressively until close to the date of the surgery.

Requirement 7 (Schedule-aware Resource Allocation) *A schedule-aware WfMS must support resource allocation strategies based on the resource availability information in a schedule. It should be able to indicate to the user the reasons for the unavailability of a resource.*

A resource may be unavailable for various reasons, among which are sickness, scheduled absence, scheduled downtime and allocation to another activity. A scheduling agent (i.e., the user that performs a scheduling task) may decide to treat each of these situations differently. For instance, if a resource has a scheduled absence then it cannot be allocated to any other activity, however if a resource is allocated to another activity, it may be reallocated to an activity with a higher priority. Thus, precise information about the reason for unavailability may substantially assist the decision making of stakeholders who are in charge of scheduling tasks.

Requirement 8 (Resource Utilisation Logging) *A schedule-aware WfMS must maintain detailed resource utilisation logs during process execution.*

To check real-time resource status, resource utilisation needs to be carefully monitored, and to enable post-execution analysis, resource utilisation needs to be recorded. A schedule-aware WfMS must capture all resource-related information as well as recording histories of resource utilisation (e.g., start and completion times of resource activities, equipment downtimes, etc) for analysis purposes.

3 System Design

Figure 2 shows a schematic of a WfMS with a service-oriented architecture capable of supporting real-time resource coordination and scheduling, required to facilitate integration of the resourcing and scheduling requirements detailed in the previous section. The *Workflow Engine* and *Process Designer* components are core modules of a typical workflow system. The Process Designer provides a user-facing design environment for the creation of process models (specifications), including their control-flow, data and resourcing requirements. Process specifications are stored in a *Process Repository* from where they may be loaded into the Workflow Engine and instantiated to produce cases (i.e. process instances). The Workflow Engine manages the execution of cases by progressing with each case according to its current state and control-flow description, and by performing the specified data mappings between the case and its tasks. Each task in a process instance is associated at design time with a specific service that will be responsible for processing the instantiated work items of that task (which includes the check-out/check-in of work items to/from the Engine). At each stage of a process instance, the Engine determines which tasks are enabled and thus should be offered to specified services for processing. Most importantly, to support schedule-awareness, two additional component services are needed. These interact directly with the Workflow Engine and each other, and perform the scheduling and allocation of resources required for work execution.

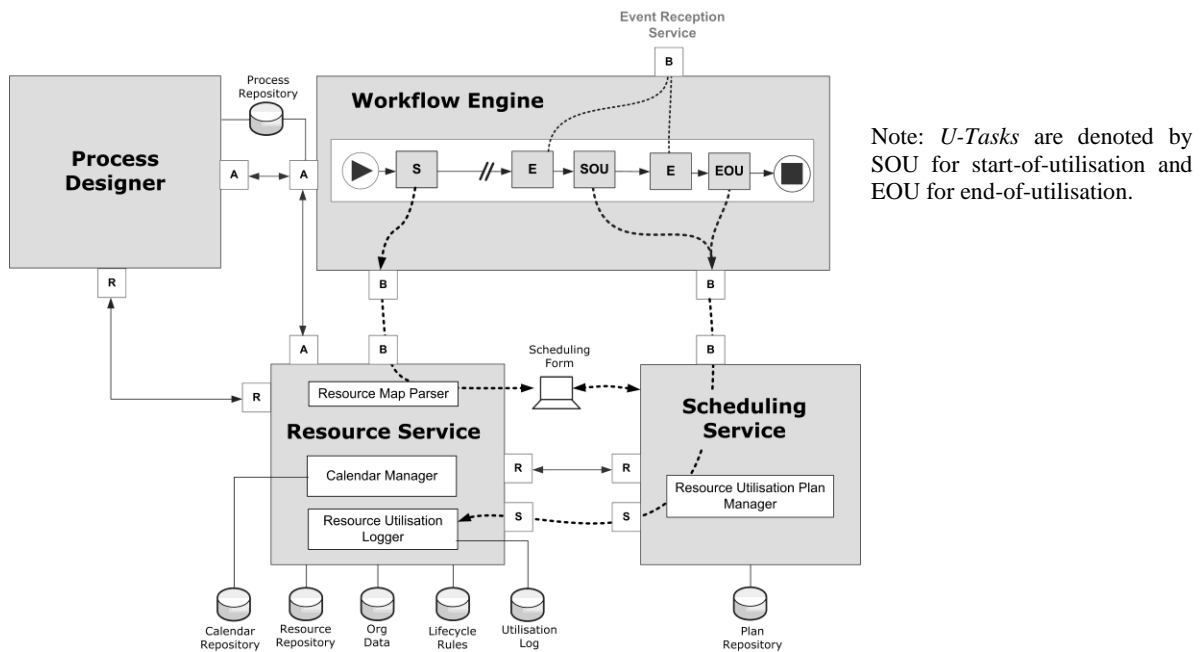


Figure 2. Architecture of a WfMS that supports resource coordination and interactions with a scheduling service.

The *Resource Service* is responsible for managing all resources (both human and non-human) of an organisation, and for the proper allocation of required resources to each enabled work item (*R-Tasks*) and activity at runtime. Each work item may have allocated to it one *primary* resource (a human resource that will directly interact with a work item via a worklist) and/or zero or more *secondary* resources (human or non-human resources that are deployed during the performance of an activity or work item, but are not required to interact directly with the workflow system). This service also provides the set of available resources to the Process Designer, and interacts with the Scheduling Service to resolve planned resource allocations (see below). This service incorporates a *calendar* that is used to maintain a record of current and planned resource utilisation or availability via updates that reflect an individual resource's lifecycle, including the states 'requested', 'available', and 'unavailable' ('busy/in use', 'on leave', 'sick', 'in recovery', 'in maintenance', etc). Also incorporated in this service is a *utilisation logging* module, which records a log of all resource utilisations and updates, and may be used for retrospective process reporting, analysis and improvement.

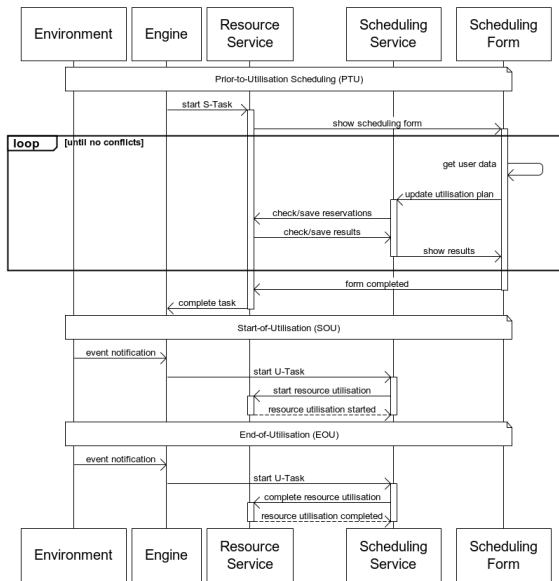
The *Scheduling Service* handles scheduling requests involving active process instances (which are triggered by *S-Task* enablements through the Resource Service and are resolved via interactions between the two services), and is directly responsible for the resolution of planning problems that are initiated by coordination and re-coordination events in an operative environment. This service is directly responsible for the handling of allocation/de-allocation tasks (i.e. *U-Tasks*) by mapping real world events (triggering *E-Tasks*) to their corresponding activities and then allocating the required resources to them. The service is based on the notion of *resource utilisation plan* that belongs to each case containing schedulable activities. Since any allocation and/or de-allocation may affect a case's resource utilisation plan (e.g., completed activities are no longer schedulable and must be withdrawn from the utilisation plan), the service incorporates the *Resource Utilisation Plan Manager*, which transforms individual case specifications into utilisation plans that contain only schedulable activities.

The Engine, Process Designer and services communicate with each other through a number of well-defined interfaces, of which the notable ones are as follows.

- *Interface A* is an Engine-oriented interface which provides for the uploading and unloading process specifications between the Process Repository, the Engine, and for registering, querying or removing references to external services;

- *Interface B* is an Engine-oriented interface which provides notifications of task enablements to the appropriate services, and for the general runtime task-related interactions required between the Engine and services. It is through this interface that responsibility for S-Tasks is delegated to the Resource Service, and for U-Tasks to the Scheduling Service;
- *Interface R* is a Resource Service-oriented interface which is used to provide resource availability data to the Process Designer and to the Scheduling Service; and
- *Interface S* is an interface that provides for communication between the Resource and Scheduling Services, such as notifications of scheduling and utilisation task enablements, the checking of resource availabilities, the updating of calendar records, and resource status change events.

Figure 3 is a message sequence chart (MSC) showing the various high-level interactions between the engine, services and environment that occur at particular stages of a schedule-aware process instance.



Note: the Prior-to-Utilisation phase may be executed one or more times before the Start-of-Utilisation phase for each process instance.

Figure 3. MSC specifying interactions between a WfMS and a scheduling service.

4 Implementation

Based on the system design described in Section 3 we have implemented all the requirements in Section 2 as a prototype extension in the YAWL environment. YAWL is a highly expressive open-source WfMS with a formal semantics based on insights gained from an extensive analysis of workflow patterns⁴. In particular, it explicitly supports patterns pertaining to the resource-perspective such as capability-based work item distribution, reallocation, and escalation. YAWL has a service-oriented architecture, allowing the scheduling extensions to be added independent of the operation of the core engine. This functionality makes YAWL well suited for schedule-aware processes.

We adopted a generic design to make the implementation applicable in principal to any problem domain including surgical care processes (see Section 6). The implementation is based on the concept of schedulable activities whose start and end are triggered by events from the environment that may be created, e.g. by sensor applications⁵. This enables us to *automatically* log the real utilisation of resources. To realise the architecture of Figure 2 we have extended an existing Resource Service and

⁴ <http://www.workflowpatterns.com>

⁵ This required an environment-aware process execution engine which are realised by implementing an even processing layer and a message-based communication service that integrates with YAWL. The details are beyond the scope of this paper.

implemented a Scheduling Service, both of which enhance the functionality of the workflow engine. In addition to the requirements of Section 3, the Scheduling Service also makes available methods that may be used for designing a scheduling GUI that is completely decoupled from the task perspective.

4.1 Resource Service

The YAWL Resource Service is a large and complex service that consists of a number of sub-components that together support a comprehensive set of workflow resource patterns and functionalities. The core service manages the allocation of resources to tasks, both through the design time supply of available resources to the Process Designer for selection, and for runtime allocation, distribution and manipulation. To support these runtime functions, a dedicated UI worklist is incorporated. The editing of task data is supported through the use of web forms, which may be user-specified or dynamically generated. An administration toolset is provided to maintain organisational data, monitor process instances and manage other services. All functionality is exposed through a series of programming interfaces, and several *pluggable* interfaces allow end-user organisations to add their own customisations.

To support schedule-aware processes, the Resource Service was extended for this implementation in several ways. Firstly, the notion of *non-human resources* was introduced to include management of resources such as rooms, tools and equipment, and their grouping into disparate categories. The ability to set a period of down-time (or recovery time) for a resource after its use to denote temporary unavailability based on business rules was also added. Secondly, the ability to distinguish between primary and secondary resources was introduced, so that a task or an activity could have assigned to it a number of required resources that will not directly interact with the workflow system. Thirdly, the concept of allocating resources was extended from a per-task basis to allow allocation *per-activity*, which marks the specified set of resources as “in-use” for a duration demarcated by a pair of events occurring within a process instance. Fourthly, a *Calendar Manager* was added to allow resources to be granted statuses such as “available”, “requested”, “reserved”, “in-use”, and so on, for set periods of time, so that available resources for future events may be selected. Finally, a utilisation logging component was introduced that maintains an archive of all resource scheduling requests and status updates. All scheduling functionalities are exposed through a programming interface that allows interaction with other services, such as the Scheduling Service.

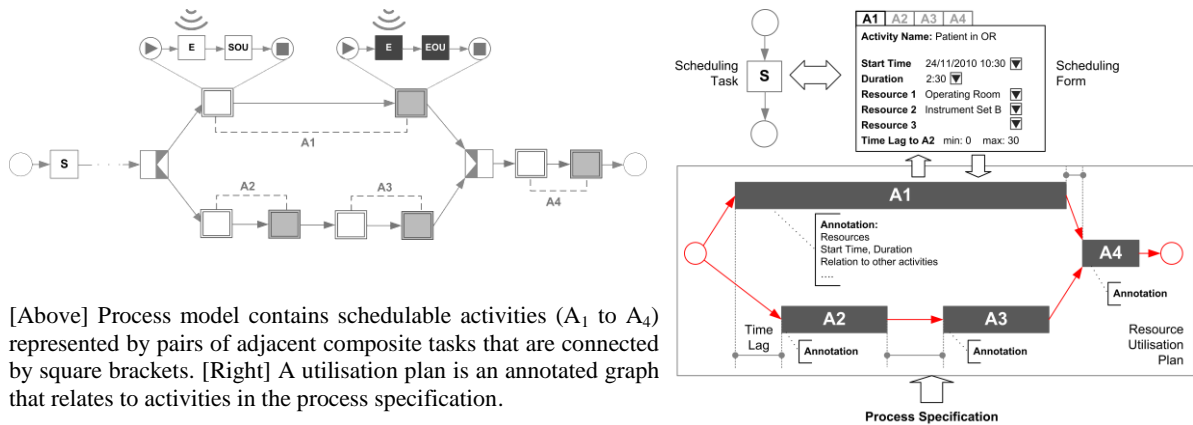
4.2 Scheduling Service

In accordance with Figure 2, the Scheduling Service communicates with the workflow engine and the Resource Service using the following interfaces:

- *Interface B* parses the process specification that is executed in the engine for schedulable activities.
- *Interface R* gets from the Resource Service a listing of known resources (including roles, capabilities, categories, and subcategories) and to make them selectable by the user.
- *Interface S* checks the resource utilisation plan based on the availability of the associated resources.

The Scheduling Service has been built around the concept of a resource utilisation plan. Figure 4 shows that a utilisation plan can be thought of an annotated graph that relates to activities in the process specification. To this end, the Scheduling Service maps the process specification of an active case onto an instance of a utilisation plan represented in an XML format.

When starting a work item of an S-Task, the user interacts with a specially designed scheduling form that allows for the specification of scheduling requirements. While the implementation of scheduling forms naturally depends on the concrete problem domain, it uses only generic functionalities of the involved services. For each activity the user can specify a start time, expected duration, and a set of resources that are required to perform the activity. Inter-activity relations (e.g., control flow and, if applicable, minimum/maximum time lag) are currently also specified.



[Above] Process model contains schedulable activities (A₁ to A₄) represented by pairs of adjacent composite tasks that are connected by square brackets. [Right] A utilisation plan is an annotated graph that relates to activities in the process specification.

Figure 4. Resource utilisation plan and its relation to activities and scheduling tasks.

Figure 5 shows a screenshot of a scheduling form that has been generated by the Scheduling Service. Activities of the current process that are still schedulable are represented as tabs (*Surgical Procedure*, *Induction*, *Extubation*). For each activity the user may assign multiple resources, which can be human or non-human. Each resource requirement is mapped to a reservation request against the Resource Service. For each reservation request the user may specify the desired outcome (*tentatively booked*, *reserved*, or *cancelled*). Resources may be selected by their role and/or capability (human resources) and category and/or subcategory (non-human resources), respectively.

Figure 5 Web form that enables scheduling of activities via S-Tasks.

Upon *complete* or *save*, the underlying utilisation plan is processed by the Scheduling Service. This includes the following steps:

1. Checking whether the time lag relations between activities are valid.
2. Passing the XML representation of the utilisation plan to the Resource Service to perform the appropriate requests. This involves updates on the resource's calendar including status changes – such as ‘requested’ or ‘cancelled’ – in accordance with the underlying resources lifecycle model.

If the utilisation plan is invalid, e.g., due to unavailability, a description of the problem is shown to the user, who may then alter the schedule accordingly and resubmit the form for re-validation. Both calendar reservations and the utilisation plan are saved to persistent storage.

When an activity becomes enacted, the Scheduling Service checks out the corresponding U-Tasks (i.e., SOU and EOU) and resolves them to the associated resources by looking up the plan repository. The start and end of the real utilisation are then logged by calling the *Utilisation Logger* in the Resource Service. Furthermore, the Scheduling Service may receive resource status change notifications from the Resource Service at any time, and acts on them by updating the relevant utilisation plan. Since the latter is capable of determining scheduling conflicts regarding instantiated processes, this functionality may be used to support re-coordination challenges that could be caused, for example, by short-term changes at the calendar level or by emergency situations.

5 Application to Surgical Care Processes

Based on an in-depth analysis of surgery-related business processes in several German hospitals, and a study of the literature, we have used our approach to demonstrate the proposed system's ability to support key scheduling requirements for surgical care processes. While the implementation has yet to “go live”, we present preliminary findings here.

Recall the surgical process model in Figure 1. We now apply our approach to this process, as shown in Figure 6. S-Tasks are labelled with **S** and R-Tasks with **R**. They allow for the specification of resources, and are checked-out by the Resource Service and distributed to a human resource to be worked on. Therefore, they must have a primary resource assigned to them.

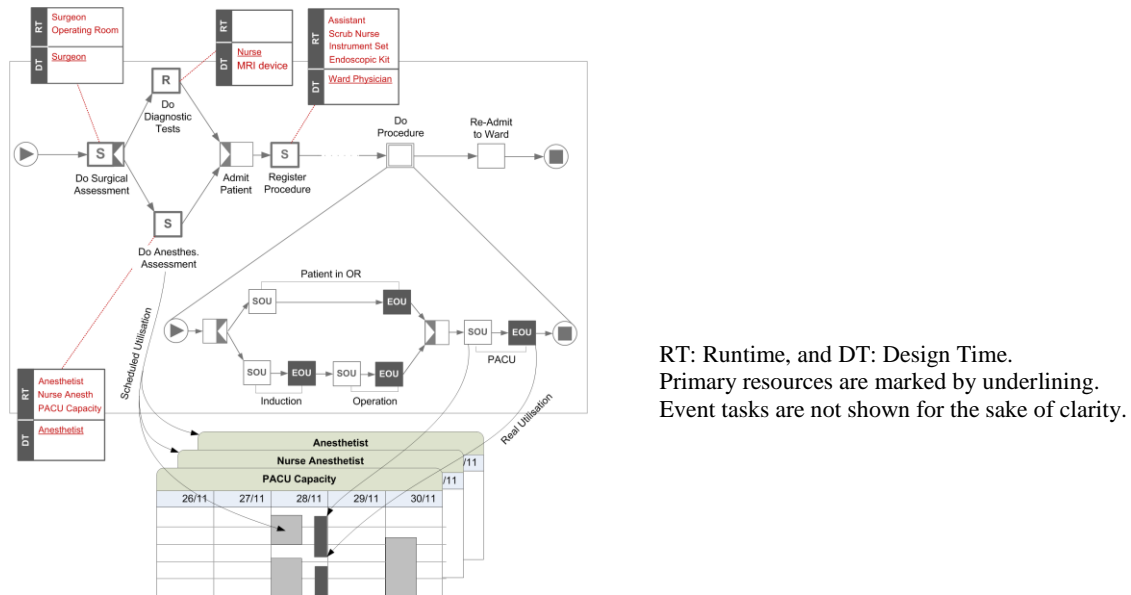


Figure 6 Model of a hospital's business process for patients that undergo planned surgery.

Task annotations indicate the role or type of the required resources as well as the point in time when these resources may be specified. R-Tasks may have a number of *secondary* resources specified, as in the case of *Do Diagnostic Tests* which upon starting allocates a nurse and an MRI device. S-Tasks differ from R-Tasks in that they allow for *scheduling* of resource-centric activities at runtime. The activities within *Do Procedure* are represented by SOU-EOU task pairs where each task will be checked out by the Scheduling Service which subsequently calls the Resource Service to write the utilisation log. The resource assignments have been defined earlier in the process by means of S-tasks.

The bottom part of the diagram shows a set of individual resource calendars. Blocks shaded in light/dark grey refer to scheduled/real utilisation, respectively. By way of reservation requests via S-Tasks, Operating Room coordinators and other stakeholders can pro-actively assign resources to each individual case, thereby potentially identifying availability issues at an early stage of the process. The real utilisation of any scheduled activity is tracked by U-tasks (SOU and EOU tasks). This information can support short-term re-coordination on the day of surgery and, furthermore, offers interesting process mining possibilities. In particular, comparing scheduled and real use of resources may be used for process improvement and retrospective process analysis.

Following the method of Clinical-Proof-of-Concepts (CPoC) proposed by Bardram (2008), the system will be evaluated in a real hospital setting by April 2011. A CPoC is a short-term test of a system carried out by stakeholders in a real clinical setting using simulated scenarios. This test will allow us to get feedback from the users based on their own experience and may be used to evaluate the approach as a whole. Furthermore, a CPoC may uncover implementation issues that would otherwise have not been observed under laboratory conditions.

6 Conclusions and Future Work

We have presented an overview of our research into the requirements for, case studies in, and design and implementation of a schedule-aware WfMS. This included identifying functional requirements for resource coordination and scheduling services together with an architecture design. We have extended the YAWL system with such a resource management service and certain scheduling capabilities to support resource-intensive and schedule-driven processes. These services serve as core components for extending current WfMSs to support such business processes in various domains.

In this paper, we showed how scheduling and resource coordination can be carried out for a surgical care process. In future work, we will extend the system by providing support for automatic rescheduling in certain cases (e.g., shifting reserved time blocks within the permitted min/max time lag) and for notification of key users in cases of scheduling conflicts. We also plan to support automatic parsing of the control flow information of a process model by the scheduling service thereby eliminating the need for a custom form to specify control flow relations. To evaluate the generality of the architecture, we will also use the system to model resource-intensive and schedule-driven processes in domains such as construction management and manufacturing.

References

- van der Aalst, W.M.P. and van Hee, K.M. (2002). *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, Massachusetts, 2002.
- van der Aalst, W.M.P. and Kumar, A. (2001). Team-enabled Workflow Management Systems. *Data and Knowledge Engineering*, 38(3):335–363.
- Bardram, J.E. (2008). Clinical Proof-of-Concept – An Evaluation Method for Pervasive Healthcare Systems. In *Proc. of the Tenth International Conference on Ubiquitous Computing*.
- Bettini, C., Wang, X.S., and Jajodia, S. (2002). Temporal Reasoning in Workflow Systems. *Distributed and Parallel Databases*, 11(3):269–306.
- Cardoen, B., Demeulemeester, E., and Belien, J. (2008). Operating Room Planning and Scheduling: A Literature Review. Technical Report FEB Research Report KBI 0807, Katholieke Universiteit Leuven, Leuven.
- Cardoen, B., Demeulemeester, E., and Belien, J. (2009). Optimizing a multiple objective surgical case scheduling problem. *International Journal of Production Economics*, 119(2):354–366, 2009.
- Cayirli, T. and Veral, E. (2003). Outpatient Scheduling in Health Care: A Review of Literature. *Product Operations Management*, 12(4):519–549.
- Combi, C. and Pozzi, G. (2004). Architectures for a Temporal Workflow Management System. In *Proc. of the 2004 ACM Symposium on Applied Computing*, pages 659–666.
- Crowston, K. (1997). A Coordination Theory Approach to Organizational Process Design. *Organization Science*, 8(2):157–175.
- Eder, J., Panagos, E., and Rabinovich, M. (1999). Time Constraints in Workflow Systems. In *Proc. of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 286–300. Springer-Verlag.
- Hutzschenreuter, A. K., Bosman, P.A.N., Blonk-Altena, I., van Aarle, J., and La Poutré, H. (2008). Agent-based patient admission scheduling in hospitals. In *Proc. of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track (AAMAS '08)*, pages 45–52.
- Mans, R.S., Russell, N.C., van der Aalst, W.M.P., Moleman, A.J., and Bakker, P.J.M. (2010). Schedule-Aware Workflow Management Systems. *Transactions on Petri Nets and Other Models of Concurrency*, volume 6550 of *Lecture Notes in Computer Science*, pages 121–143.
- Plasters, C.L., Seagull, F.J., and Xiao, Y. (2003). Coordination Challenges in Operating-Room Management: An In-Depth Field Study. In *AMIA Annual Symposium Proc. 2003*, pages 524–528.
- R-Moreno, M.D., Borrajo, D., Cesta, A., and Oddi, A. (2007). Integrating Planning and Scheduling in Workflow Domains. *Expert Systems with Applications*, 33(2):389–406.
- Senkul, P. and Toroslu, I.H. (2005). An Architecture for Workflow Scheduling under Resource Allocation constraints. *Information Systems*, 30(5):399–422.
- Tramontina, G.B., Wainer, J. and Ellis, C.A. (2004). Applying Scheduling Techniques to Minimize the Number of Late Jobs in Workflow Systems. In *Proc. of the 2004 ACM Symposium on Applied Computing*, pages 1396–1403.