A Framework for Adaptive Workflow (Model Human Behavior - Don't Constrain It!)

H. Dominic. Covvey, MSc, FACMI, FHIMSS,[1, 5] Donald D. Cowan, PhD, DSc,[2,3,5] Paulo Alencar,[2,3] PhD, William Malyk, MMath,[2] Joel So, BMath,[2] D. Henriques, MEng[4] and Shirley L. Fenton, BES, MA[3,5]

Technical Report CS-2012-06

Abstract

Healthcare processes are complex and highly variable from day to day. Healthcare process execution can be affected by any participant in a process, including clinicians, the patient, and the patient's family, as well as by environmental factors such as clinician, staff, facility and equipment availability, and patient clinical status. However, there are no solutions that enable computer support for a process to address the full complexity and variability of healthcare processes. We have re-conceptualized workflow and developed an innovative process representation and execution framework based on concepts from software engineering, machine learning, complexity, and database management. This new framework frees processes to track human behavior, thereby releasing us from the constraints of past methods.

Introduction

Many of our existing computer-based systems and applications, by their very nature, tightly constrain human behavior. However, for a brief phase, during the set-up and implementation of a system, a degree of adaptation to the local environment is usually possible in most systems. Unfortunately, once this adaptation has been implemented, it effectively freezes the work processes into a state that is virtually invariant, thereby frustrating our need to adapt to changing situations. Over the last few years there has been increasing recognition of the importance of the requirement for the support of flexible, adaptable processes if systems are to be truly successful and address the needs of their users. In particular, it has become clear that systems must synergistically align with and support the constantly varying nature of human workflow. In this article we will use the terms 'workflow' and 'processes' interchangeably, as well as considering clinical and operational processes similarly.

Furthermore, while it is recognized that healthcare processes are complex, little has been written about the fact that health care satisfies the formal definition of a complex system, exhibiting characteristics such as having many interacting agents and objects whose behavior is affected by feedback and that adapt to their histories, a system that is influenced by its environment, that appears to be 'alive' as it evolves, that exhibits emergent phenomena that arise without a central controller, and a 'system' that comprises a complex mix of ordered and disordered behavior, with parts acting at the edge of chaos [1]. Although much has been done on the representation of workflow in business settings, the representation of workflow in highly dynamic settings, like health care remains a challenge.

[1] School of Pharmacy, University of Waterloo
[2] David R. Cheriton School of Computer Science, University of Waterloo
[3] Computer Systems Group, University of Waterloo
[4] Systems Design Engineering, University of Waterloo
[5] National Institutes for Health Informatics

Complex dynamic environments are characteristic of health care. They typically involve considerable human interaction resulting in a high degree of variability. Healthcare settings have many decision makers, kinds of decisions, events and a multitude of reactive, subsidiary workflows that often require a quick revision of the course of action. Operational and treatment protocols attempt to regularize workflow, but the needs of care, the great variety of situations and individuals' decisions, the exigencies of the moment (such as equipment failure or medical emergency), and the nature of human beings frustrate attempts at regularization, often resulting in protocols being labeled as 'rigid' or 'cookbook medicine' and hence are often abandoned. While event sequences in healthcare processes may abide by loose constraints, they are largely non-deterministic. Therefore, it is difficult, if not impossible, to *prescribe* healthcare workflow fully. Instead, workflow must be dynamic, self-adapting and evolving at execution-time to match the dynamicity of the environment.

Traditional workflow representation and execution technology, by its very static nature, supports a finite set of scenarios. In fact, traditional workflow is understood to support, at best, the union of atomic workflow patterns described by van der Aalst [2]. Available workflow platforms that support these patterns are often incomplete, unsatisfactory, or even non-existent. In fact, no single commercial product supports all listed patterns, let alone the dynamicity we describe [2].

There are also adaptive case management systems [26] where, in a healthcare setting, everything that happens to a patient such as history documentation, related documents and test results is collected in a case folder and then managed centrally by one individual. This person then forwards the case to the next person for review, approval and the possible addition of information because of investigations or actions performed. Such a system is dynamic in that the current manager of the case folder can decide the next step in the workflow. The workflow is implicit in that it is represented by and in the time sequence of the documents. However, workflow can be inferred and can be mined or manipulated.

In contrast, we have developed a new architecture and implementation for representing and guiding healthcare processes based on earlier work by Jackson and Twaddle [4]. This new approach is able to address workflow variability and complexity by describing workflow explicitly in the context of entity-relationship (E-R) models. This work has provided enhanced ability to model and guide human behavior, minimizing the need to constrain it. An early version of the approach is described in [15].

An Example of Complex Dynamic Workflow – Diagnostic Imaging

To illustrate the complexity and dynamic nature of workflow in a healthcare setting, such as a hospital, consider the sequence of processes in a diagnostic imaging (DI) department where many different types of images are produced (CT, X-ray, MRI,…). It is possible to describe a standard set of steps that define the basic workflow in the DI department. A somewhat abbreviated version of these steps follows in **Table 1**.

This group of 11 steps might be described as the ideal workflow for a DI department in which an entity (the patient) enters the department and proceeds through the examinations without interruption. The group of 11 steps is called a 'Lifecycle' for the patient entity. Each step in a Lifecycle is called a 'Stage' and each Stage can consist of one or more 'Services' or 'Tasks.' For example, step 6, the preparation stage, may comprise several tasks as the patient is readied for the actual imaging stage. Tasks within a stage generally have pre- or post-conditions; for example, a patient will not normally undergo a procedure until the correct paper work has arrived. Note that preparing and forwarding paper work is part of another lifecycle involving a different entity, namely, the ordering physician, and, normally, step 4 will not occur until the paper work has arrived. Thus, tasks in one lifecycle can be dependent on tasks in another lifecycle.

1. Order a diagnostic examination – a physician or some other health professional has decided that imaging is required to understand the patient's anatomy.
2. Patient arrives at the DI department.
3. Patient is received.
4. Patient identity is verified against the order and received ID for the DI department.
5. Patient goes to an imaging suite (CT, X-ray, MRI,…).
6. Patient is prepared for the procedure (tested for allergy, injected with dye, protected by lead apron,…).
7. Patient goes into the imaging suite and is imaged and eventually leaves.
8. Images are reviewed to determine if adequate for diagnostic purposes.
9. Images sent to diagnostician (normal or emergency processing).
10. Report created and report and images filed together.
11. Patient is released from DI department.

Table 1: Basic Workflow in a Diagnostic Imaging Department

There can also be many reasons why stages or tasks might be skipped. Departments in a healthcare setting are limited by resources, often creating the need for dynamic changes in the departmental workflow. For example, equipment might need repair, the pre-conditions for a DI session may not be met or an emergency may occur at any time. We provide a few detailed examples in an attempt to illustrate the complexity of the workflow that might occur.

In step 1, the patient is suffering from severe trauma (emergency) owing to an accident, so there may not be time to complete the initial paper work. The patient, in this case, may go directly to step 7, as it has been judged that there is no time for normal prep. Since DI departments are resource-limited, this emergency would have side effects as it could impact the entire schedule within DI and could force less critical cases to be returned to their ward and be rescheduled to another day. Normally there is a pre-condition for a DI examination, namely, that the paper work is complete; in this case, because of the emergency, the pre-condition would be skipped and the initial paper work would come later.

In step 3, the patient arrives, but there is no accompanying order (paper or electronic order pre-condition). Does the patient wait while the order is found or the ordering physician is located, or is the patient released from the DI department and returned to his or her ward or sent home? The decision made in this case can also impact the DI schedule.

In step 5, the patient is scheduled for imaging using certain equipment. However, the equipment needs maintenance. Does the patient wait until another equivalent system is available or is he/she returned to his/her room or told to return another day?

Once a patient arrives in step 2, medical records could indicate that the examination has been preceded by another (a pre-condition task that needs to be verified) and that the second exam should not be done. For example, a gastrointestinal study using Barium will make it impossible to do other exams, like a lymphogram, for several days. Thus the patient must be rescheduled.

In each of these cases the workflow has the potential to be quite complex and highly dynamic because of the many uncontrollable factors from both outside and inside the DI department. In fact, each of the examples could be more complicated than illustrated as the factors used here could compound. The objective has been to illustrate the complexity of just one unit in an organization to show the inherent complexity of healthcare workflow.

On the Concept of Dynamic Workflow:

Now that the need for 'dynamic workflow' has been illustrated, we should provide a general model of its meaning. First, we distinguish the notion of a Workflow Scenario (the real-world environment and processes; what is actually happening) from a Workflow Application (the computer-encoded form of a workflow scenario; a computer model of reality).

By separating application from scenario, we can see that workflow scenarios range from static at one extreme to dynamic at the other. In static scenarios, all instances of scenario execution are substantially similar, both in terms of the work performed (the goals: "what" and "why") and the specifics involved (the context: "who," "when," "where" and "how"). In dynamic scenarios, on the other hand, each instance of scenario execution addresses the same work goals, but, as can be seen from the diagnostic imaging example, the context varies from instance to instance. A workflow scenario is (highly) dynamic when there is a (high) degree of instance-to-instance variability and unpredictability (in tasks, sequence, actors, etc.) to achieve the same goal. In our diagnostic imaging example, the aim is to produce an image, although the path to obtaining the image may vary significantly.

Traditional workflow technologies adequately support classic 'prescribed'/pre-defined workflow applications: workflow is prescribed at design-time to model a static scenario that, at run-time, is supported by the prescribed workflow application. Dynamic workflow applications, on the other hand, must dynamically assemble the workflow at run-time, in response to the dynamic scenario being modeled. To rationalize the workflow problem space further, we note that workflow applications can also be viewed as proactive and retroactive, where these two concepts are closely related to static and dynamic workflow.

In proactive workflow, the workflow application guides scenario execution, and the application relies on design-time completeness for run-time efficacy. Anything undefined at design-time becomes a run-time exception, which might be dealt with by an action such as consulting a manager. Processing forms for insurance applications or claims would be an example; an exception or consultation might occur if a piece of information or a document is missing.

In retroactive workflow the reverse is true; the workflow application is retroactively realized from scenario execution; the workflow is captured from the actual tasks and their sequencing. Capturing the workflow requires recording the current step followed by the next step in the scenario sequence until the scenario is complete. This recording can be manual, but technologies such as RFID tags can be used in dynamic situations such as a DI department to automate the process of capturing the workflow. Once workflow is captured it could be mined to:

- do an analysis of the department work or drive a simulation to visualize and improve current processes or

- create a semi-automated guideline for workflow that can prompt and assist the staff as a patient is processed through the DI suite.

Note that the mining can be static or dynamic. The workflow could be mined periodically or continuously. Continuous mining would lead to guidelines that are being improved in real-time, which should provide almost instantaneous improvement in the workflow guidelines. Continuous mining could also allow an ad-hoc process to be matched to an existing guideline, to provide improved decision support or even to provide early warning for ‘bad’ workflow paths.

In order to clarify the concept of dynamic workflow further, consider the following issues:

(1) static/prescribed workflows are characteristic of many commercial processes: the workflow (e.g., actions, actors and sequencing) can be almost completely described and the pre-defined workflow will be executed similarly for each occurrence of a transaction;

(2) static workflows are easy to represent and execute as they have few variations and no unpredicted exceptions;

(3) dynamic workflows, characteristic of health care, will adapt and vary according to the situation and, although there are instance-to-instance similarities, the sequence of actions, the facilities/equipment used, and the agents participating will change in response to activities such as an emergency, the patient’s clinical state or response or clinician intra-workflow insights;

(4) dynamic workflows are challenging to represent, as the workflow is a record of how the process was performed as well as guiding the process from previous workflow executions. For this reason we say that dynamic workflows are historical records of what was at the time decided and performed.

Workflow descriptions can provide a bound on the workflow, but not a rigid definition of processes. Looking at workflow this way enables the efficient coupling of humans with computer-guided processes, reducing or eliminating the perception of protocol or workflow rigidity.

Representing and executing dynamic workflow

Workflow is usually described by connecting services or tasks in a programmatic format such as a workflow language, flowchart or other sequencing mechanism. Thus, prescribing a workflow in a dynamic environment could require a protocol or ‘program’ of immense complexity, as one tries to ensure that all possible choices and their sequencing are captured in advance.

The approach used in our research to capture workflow and include dynamic services and context, which has been used to describe over 70 service-oriented information systems [3], is based on the work by Jackson and Twaddle [4]. Workflow is represented using an entity-relationship (E-R) model [5] connecting the atomic services that constitute the workflow. This E-R model is then transformed into a relational database. Thus, the control structure and all information about services are captured in a data structure that can be easily modified and then either compiled or interpreted into a workflow ‘program’ using some programmatic representation such as a flowchart. An expanded description of the model can be found in [16].

Entities: Following Jackson/Twaddle [4], the first step in producing an E-R workflow representation is to develop a data model in terms of entities that are central to the description of the business processes. In other words, we must determine the entities to which the workflow is applied. In a medical laboratory context, the data model could consist of entities such as admissions-clerk, patient, physician and test-order. These entities have corresponding entries in the lab database. In the diagnostic imaging example one key entity is the patient whose image is being captured.

Lifecycles, Stages and Services: Each entity in an operational information system has an associated workflow or 'lifecycle.' In the diagnostic imaging example, the lifecycle is the entire set of 11 steps that is the normal process for a patient.

Each lifecycle goes through a number of sequential stages; each stage contains a number of services/tasks applied to the entity. A simple example of a lifecycle is the movement of a patient through the admission stage, to the testing stage and finally to the diagnosis stage. In the case of the diagnostic imaging example, each of the 11 steps in the lifecycle represents a stage. Stages can be divided into sub-stages. The lifecycle for the patient ends as he/she is discharged or passed on to a new lifecycle for further intervention. Services can be executed sequentially, conditionally, or repeatedly. Parallel execution is also possible through the splitting/fork and merging/join of services. Although stages must not be skipped, they can contain a skip-this-stage task. For example, if a patient does not complete the admission stage, then skip-this-stage would be executed in all subsequent stages. Services/tasks are interdependent: the application of one service may have to await the completion of services in other stages or lifecycles associated with other entities.

Generally, services are interconnected through control flow (sequence, condition, repetition, fork and join). However, it is possible to introduce dynamic services into the model that can be chosen based on the current situation. These dynamic services are chosen as the workflow progresses and specific situations arise. For example, there may be several devices that could provide a needed testing service. The choice of device could occur dynamically, based on variables such as machine or operator availability.

Workflow as an E-R and Database Model: Workflow lifecycles, stages, services and interdependencies can be represented by an E-R model, as shown in **Figure 1**. E-R models can be further mapped into a relational database where the concepts in the system (entity, lifecycle, stage and service) and the relationships among them each correspond to a table. Each entity has one corresponding lifecycle; each lifecycle can have multiple stages and each stage can have multiple services. The relationships among the services can further be annotated to include sequence (pre-conditions and post-conditions), choice, repetition, fork and join. Pre-conditions and post-conditions can be used with dynamic services to specify services that must be performed before or after the selected service.

Thus, workflow is represented as a data-structure, a significant innovation, rather than a program. Code related to a service can be stored in the database as an embedded procedure, or could reference an external service such as a Web service (as seen in SOA) [25] or a RESTful Web service [24]. A data structure makes it possible to present the entire process graphically, supporting visualization as needed by a program or flowchart.

The relationship between lifecycles, stages, and services in a workflow can be modified by changing a data structure rather than a program structure. Modifying workflow is therefore easier, because the abstractions corresponding to the workflow or control structure are clearly identified. End-users can easily be shielded from these details using a visual interface. In addition, E-R models represented by XML-tagged structures enable the development of a declarative domain-specific language for describing workflow.

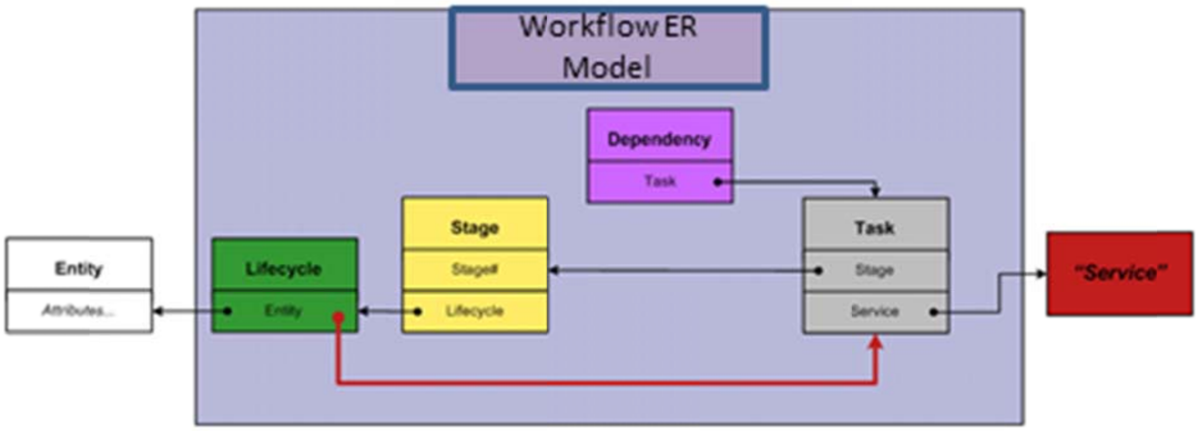


Figure 1: Entity Relationship Model for Workflow

The declarative workflow representation can be transformed into other representations such as BPEL [6], UML [7] or XMI [8] through transformations defined in languages such as XSL/XSLT [9]. This approach has been demonstrated in [10] where XML-based declarations for agents were transformed into code written in the programming language C. This workflow model associates services with entities through a workflow structure, and services are loosely coupled to each other and to the related entity.

A Database for the Workflow

In order to illustrate how the database would be constructed, we outline the database structure and use the diagnostic imaging example described to illustrate specifics. Each structure in the workflow has a corresponding database table namely entities, lifecycles, stages and services/tasks and there are also tables (relationships) connecting these structures. Each of the entries in each table has a unique identifier.

Figure 2 shows the table structure for entities, lifecycles and the relationship between them. There is one table containing each entity in the process whose workflow is being modeled and one table containing all the lifecycles, one for each workflow associated with an entity. Each entity and each lifecycle has a unique identifier and some accompanying description. Note that each entity may refer to a fuller description of the entity in another connected database such as personnel or equipment. The table (relationship) connecting entities to lifecycles in **Figure 2** has two columns to identify, the entity id within the table and the corresponding lifecycle. Note that the plus (+) sign is used after each entity and lifecycle name. This just indicates that there may be more information that is not shown. We use this convention throughout this example. In the DI example each patient entering the imaging suite would be an entity and each patient would have an associated lifecycle that would depend on the type of imaging to occur, but would be similar to the 11 steps described in the DI example.

Figure 3 is similar to **Figure 2**; it shows the connection between lifecycle and stages and the table (relationship) connecting the two. Note that a lifecycle can have many stages, but this concept can be captured in the relationship by having the same lifecycle id associated with many different stage ids. In the diagnostic imaging example, each lifecycle could have all 11 stages, although as shown in the example a number of them could be skipped, as the procedure does not require all of them because of an emergency or some paperwork being missing.

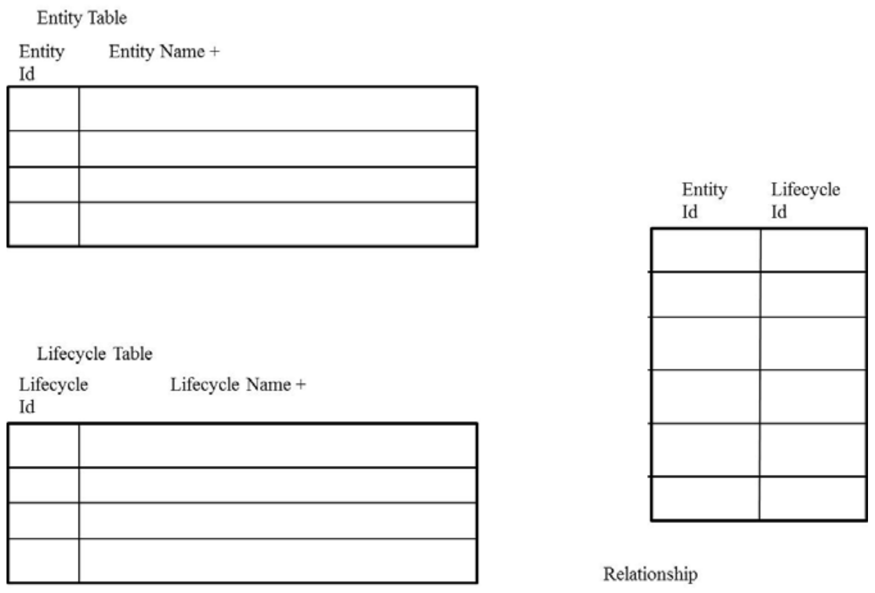


Figure 2: Tables for Entities, Lifecycles and Entity-Lifecycle Relationship

Figure 4 show the relationship among stages and tasks. The relationship shows what stages contain what tasks. A task in the DI example may relate to preparing the patient for the imaging sequence by administering a contrast agent or asking him or her questions about implants or dentures. It does not show the relationship among tasks.

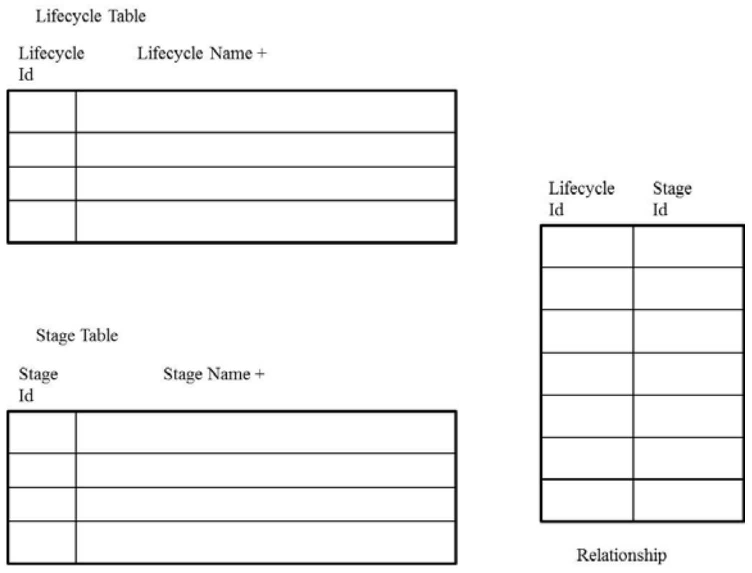
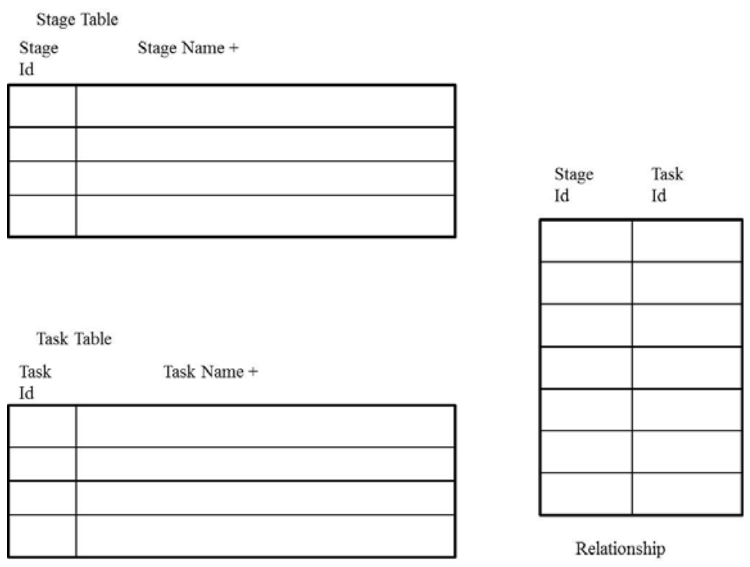


Figure 3: Tables for Lifecycles, Stages and Lifecycle-Stage Relationship



Stage Table

Stage Id	Stage Name +

Task Table

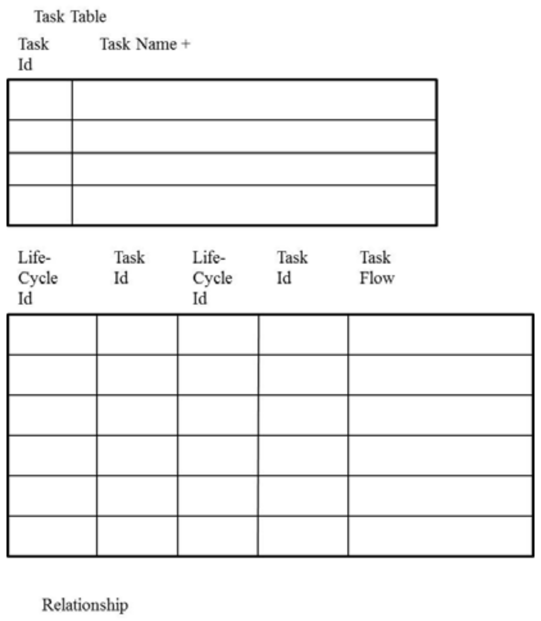
Task Id	Task Name +

Stage Id	Task Id

Relationship

Figure 4: Tables for Stages and Tasks and the Stage-Task Relationship

The relationship among tasks is left to **Figure 5**, where the relationship shows which task follows which and if the relationship is sequential, conditional (more than one task follows another), loop (go back to an earlier task if the condition is not satisfied), fork or join. This information is contained in the column task flow. Thus, we have encoded both the sequence of tasks and, as a consequence, which tasks are pre- and post-conditions for other tasks. For example in the diagnostic imaging case, certain tests must not occur until a time period has elapsed or until paper work is available. Notice that in the relationship, each task is also preceded by its lifecycle id as interdependent tasks (pre- and post-conditions) may come from different lifecycles. We could also include the lifecycle id and stage id in the task table, but this information can be recovered from the other structures.



Task Table

Task Id	Task Name +

Life-Cycle Id	Task Id	Life-Cycle Id	Task Id	Task Flow

Relationship

Figure 5: Tables for Tasks and the Relationship among Tasks

Now that all the relationships among the various components of the workflow are specified in a set of tables, re- sequencing of workflow is very easy. Just change the entries in the relationships. For example, if we want to remove a stage, we just change the stage ids in two rows in the stage relationship. The same principle applies to tasks. Thus it is possible to capture workflow in a dynamic situation and mine this information to create a guideline that can mimic the workflow practices currently being used in any healthcare situation.

Making the Workflow Model into an Operational System

Workflow has now been converted to a data structure where the atomic service or task encapsulates the work to be done. The remainder of the data structure captures the relationships and dependencies among the services or tasks. The services can be manual, meaning there is a requirement for human intervention as in completing a form, or automatic, as in processing a credit card payment.

Workflow Engine for Dynamic Workflow

We now define a workflow machine or 'engine' driven by a set of rules that is capable of traversing the workflow data structure in the proper sequence, including dealing with dependencies. A depiction of the engine is shown in **Figure 6**. The machine processes (traverses) the data structure, interpreting or compiling the data in the structure into code which can then be launched.

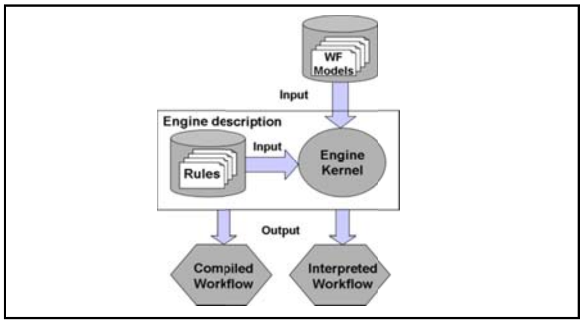


Figure 6: The workflow engine/machine

To address the representation of dynamic real world processes, we have designed an approach involving service selection and real-time composition of services into workflow instances shown in **Figure 7**. In this approach, the activities or tasks within a workflow are viewed as services. Example services include such activities as a receive-patient service, a register-patient service, and a produce-report service. These services are pre-defined and may themselves be composed of more granular services. In the DI example described earlier, the services could be match-paper-work-to-patient or test-for-allergy, to name two examples. These services can be provided by a services-server on a request from the workflow generation engine. Services have pre-conditions (services that must be executed before this service) and post-conditions (services to be executed after this service). These pre- and post-conditions are captured in the tables shown in **Figure 5**.

To choreograph these loosely-coupled services that comprise a workflow, a service selection unit is incorporated that can select services both statically (sequential prescription) and stochastically (by way of inference). An inference engine-based service selection mode selects services (the next activity in a workflow) based on the decisions of the participants in a workflow as well as on historical data collected from past instances (via mining). If a selected service indicates that it must be preceded by another service, this other service will be selected and executed, otherwise the first service will be executed, and so on. In this probabilistic model, services are selected at run-time based on three key criteria: available contextual knowledge of the current workflow instance, knowledge collected and mined from previous workflow instances, and composition rules and constraints – a rules-engine. This rules-engine allows enforcement of service sequencing based on temporal dependencies, data availability, business rules, and role restrictions.

As mentioned, services can also be sequentially invoked, thus providing compatibility with classic prescribed workflow models. A workflow instance continues selecting services (prescriptively and stochastically) until it has addressed all the required services to complete the process. In highly dynamic environments, workflows may possibly enter a state where the next appropriate activity is unknown and cannot be inferred. This may occur during times of process discovery or reengineering within an organization. To address this, we incorporate workflow mining services. A workflow mining service can record data from the current workflow instance and subsequently examine previous instances of whole workflows to guide the selection of services until participant decisions redirect the workflow. This forms a closed-loop feedback cycle with the inference-engine used for service selection.

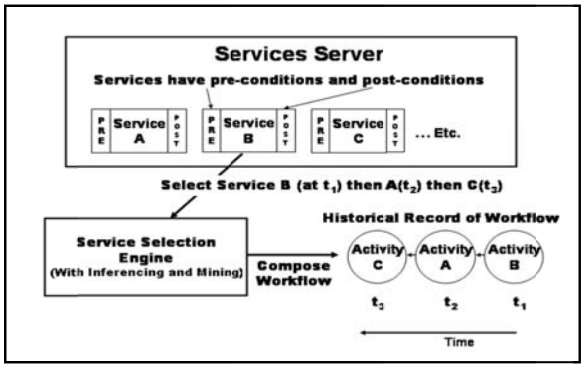

Services Server
Services have pre-conditions and post-conditions
PRE
Service A
POST
Service B
Service C
... Etc.
Select Service B (at t_1) then $A(t_2)$ then $C(t_3)$
Historical Record of Workflow
Service Selection Engine (With Inferencing and Mining)
Compose Workflow
Activity C
Activity A
Activity B
t_3
t_2
t_1
Time

Figure 7: Service Selection and Workflow Composition

Walking over an Example - How the Workflow Engine Operates

With the complete model in place, an example is now provided to show how the workflow engine operates in practice. Complete details of the model and workflow engine are in [16].

We use an In-Patient Booking (IPB) Workflow for illustration and a view of this workflow appears in **Figure 8**. The IPB Lifecycle is tied to an exam requisition form, and the lifecycle details the movement of this form. The tasks performed in this workflow are manual tasks carried out by people and largely involve the movement of an exam requisition form and actions taken to complete it. In order to simplify the description we have kept this as a static (proactive) workflow. We now 'walk over' this workflow to illustrate how the workflow engine operates.

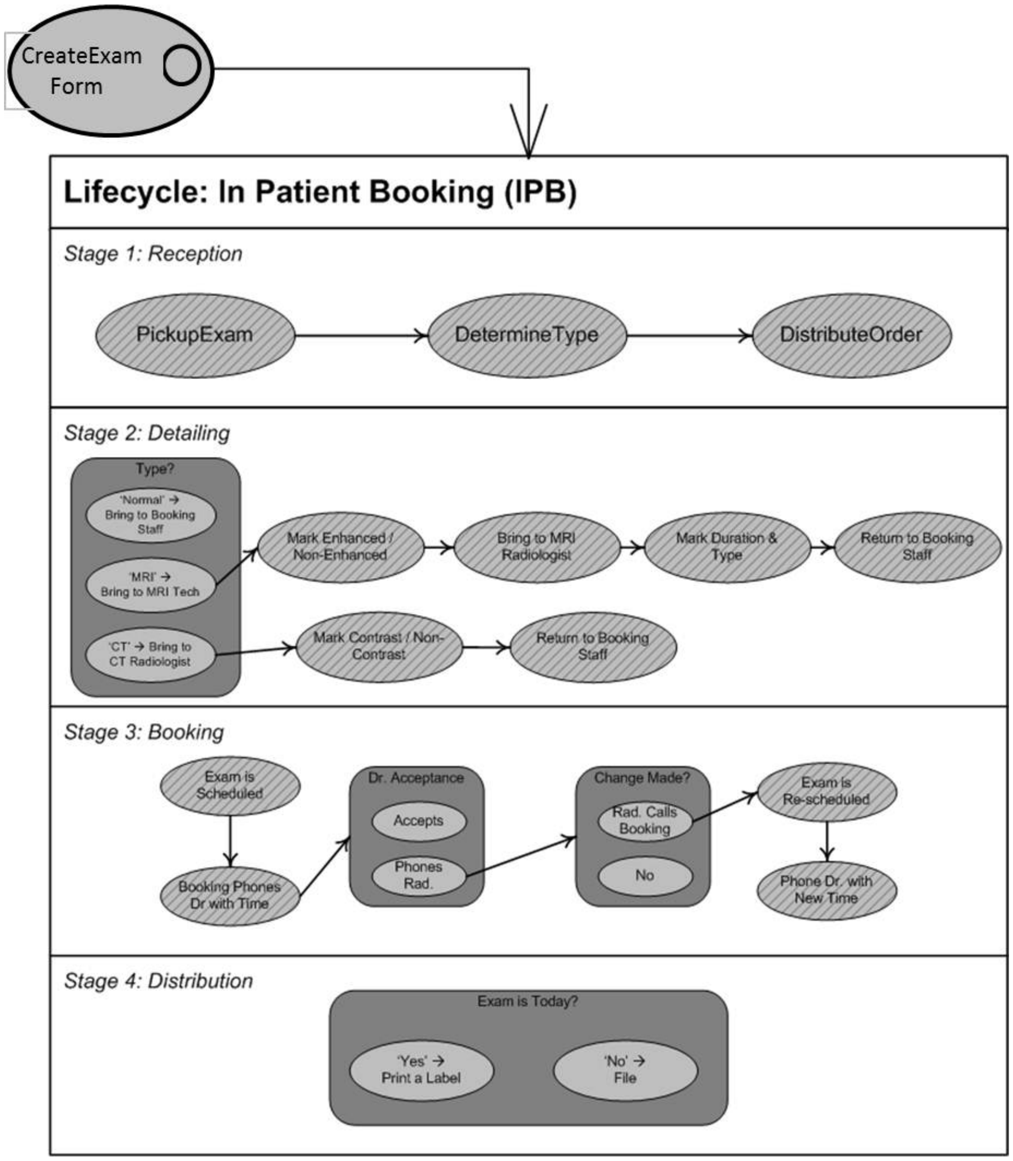


Figure 8: Sample In-Patient Booking Workflow

In **Figure 9**, we show an external task, CreateExamForm, which creates an 'Exam Requisition' entity in the system that is made available on exam clerk's terminal for completion and forwarding. Each time such an entity is created it receives an entity id and is stored in the entity table. The Requisition (entity) has an associated workflow for booking the exam/filling out the form, so performing the CreateExamForm task also starts the IPB lifecycle. The Exam Requisition entity connects to the IPB lifecycle that connects to Stage 1, the Reception, which further connects to the first task, namely PickUpExam. Of course, these connections are made through the tables in **Figure 2** through **Figure 4**, which contains entries for the IPB lifecycle, the Reception stage and the initial task for that stage and the relationships between the entity, lifecycle, stage and service.

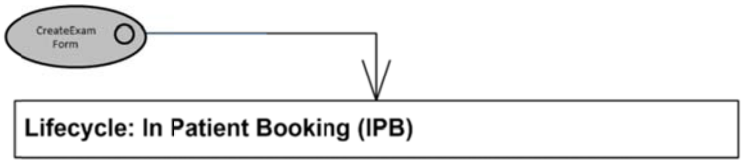


Figure 9: Creating the entity for the lifecycle

Stage 1 - Reception

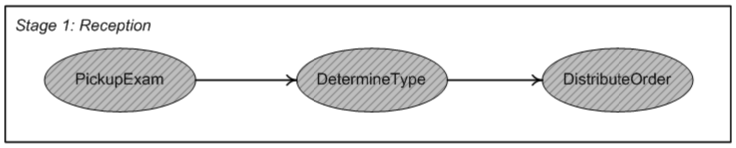


Figure 10: Stage 1 - Reception

Stage 1 ‘Reception’ shown in Figure 10 of the IPB Lifecycle begins. At the start of any stage all initial tasks are made available on the system. In this case PickupExam is the only initial task in this stage. When it is completed, activity routing through the tables in Figure 5 makes the DetermineType task available. Similarly on completion of DetermineType the task DistributeOrder comes next. The sequential relationship between tasks is encoded in the relationship in Figure 5.

In summary, this stage contains nothing more than a sequence of steps that must be followed to complete the reception of the exam requisition. This sequence involves accessing the form on the workstation computer, determining the type of exam required, and then forwarding the form to the proper department. Completing this sequence completes the stage and the IPB Lifecycle proceeds to the Detailing Stage.

Stage 2 – Detailing

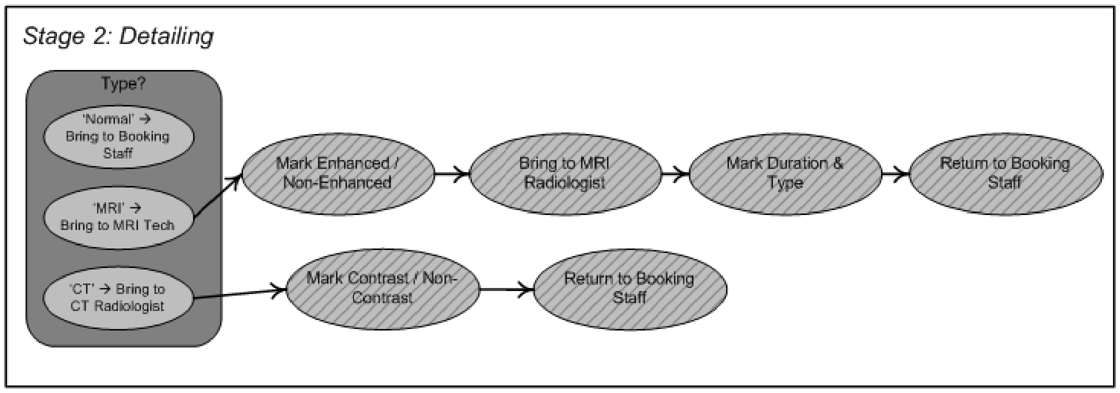

Stage 2: Detailing
Type?
'Normal' → Bring to Booking Staff
'MRI' → Bring to MRI Tech
'CT' → Bring to CT Radiologist
Mark Enhanced / Non-Enhanced
Bring to MRI Radiologist
Mark Duration & Type
Return to Booking Staff
Mark Contrast / Non-Contrast
Return to Booking Staff

Figure 11: The Detailing Stage

Once again at the start of Stage 2, ‘Detailing,’ causes the initial tasks to be made available on the system. In this case three tasks are available and the choice of task depends on the exam type required. If the exam is an MRI: the form will be delivered to an MRI Technician, the Tech will mark it enhanced/non-enhanced, the form will be delivered to the MRI Radiologist, the

Radiologist will fill in the duration and type of the exam, and finally the form will be taken to the booking staff. If the exam is a CT, the form will be delivered to the CT Radiologist, the Radiologist will mark it contrast/non-contrast and finally the form will be taken to the booking staff. For all other exams (X-Ray for instance), the form is taken directly to the Booking Staff.

In this stage we find three alternatives, the selection of one path excluding the other two. Two of the paths result in different sequences of tasks being carried out. When one of these paths concludes, the stage is over, the exam requisition form contains all the details required to carry out the exam and the IPB Lifecycle proceeds to the Booking Stage

Stage 3 - Booking

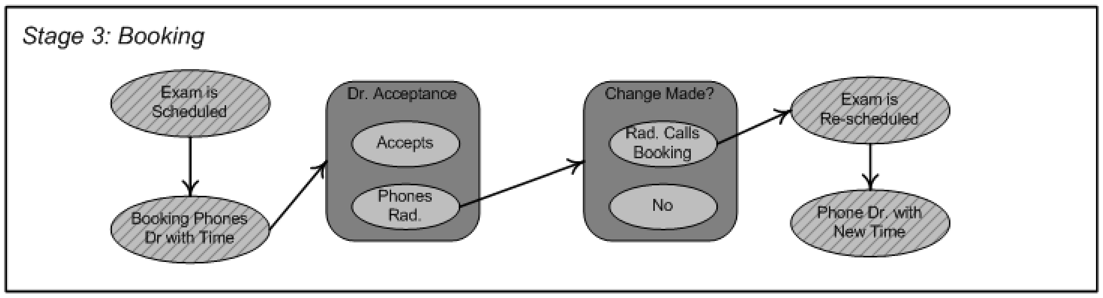


Figure 12: Booking Stage

Stage 3, 'Booking,' shown in Figure 12, begins with one task, scheduling the exam. Once this occurs, a more-or-less sequential process takes place where the Doctor is consulted, given an opportunity to have the exam rescheduled, and the exam is rescheduled if requested and possible. This stage can end if the doctor accepts the initial test date, if a change is not made, or if a change is made after the doctor has been alerted. Once the exam is booked, the IPB Lifecycle proceeds to the final stage, Distribution.

Stage 4 - Distribution

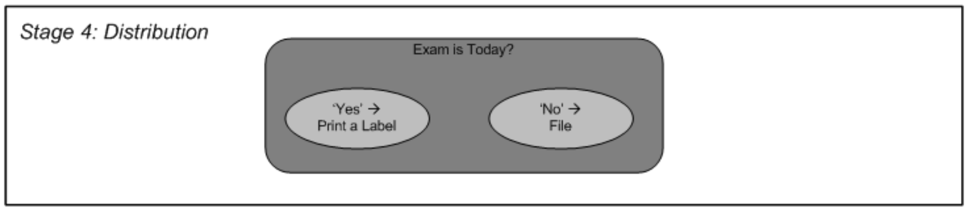


Figure 13: Distribution Stage

Stage 4, Distribution in Figure 13, is trivial. If the exam is today, a label is printed and fixed to a file for the exam; if it is not, the exam requisition form is filed.

The workflow engine looks at each step in the workflow database much as we have in this example and determines what can happen at each step in the workflow by examining the content of the database tables.

Making the workflow model practical

How can the patient be taken through the DI department? At each stage, the stage may be skipped, the remainder of the lifecycle may be skipped, or some tasks within a stage may be skipped. For example, the question “Is the patient properly prepared for the exam?” can lead to proceeding, or to waiting until the patient is properly prepared, or to aborting the whole procedure, re-scheduling and then sending the patient home or returning the patient to the medical ward.

These steps or similar ones must be captured as the patient is processed. The issue is not to add to the number of steps being carried out by staff in an already hectic department. A patient’s identification must always be compared to the patient order at every stage to ensure that the correct patient is being processed. This is usually a visual verification compared against the orders. If the information is to be processed in a database, it must either be re-entered or captured from the order form or bracelet.

It is clear that a variety of new and long-available technologies can enable and automate this interaction. For example, RFID systems can wirelessly address the need to identify an individual, object or location and near collocation is possible. More advanced RFID systems already permit the location of any entity within a small volume of space, enabling a system to track activities. For example, each patient can have an identification bracelet containing an RFID tag that has a unique patient id. Such tags are inexpensive and do not require power. In addition, speech recognition can support hands-free interaction for at least well-defined information capture. Underlying all of this is the information system that supports the care process – in our example this would be a Radiology Information System with PACS capability. Our work is an enhancement of this foundational system through the addition of a workflow engine and the components that can interact with the above-mentioned devices.

As the workflow is no longer rigidly prescribed, it can be made to fit the dynamics and demands of an incredibly complex department. Medical judgment can be brought into play; no longer is it necessary to explain breaking the rules when the rules do not really apply. In addition, the workflow for each procedure is captured as it happens, and provides a history of individual workflows and all workflows that have occurred in a department over a period of time. These workflows can be mined to determine the practices that have been followed and if these practices are best practices. The workflows also provide a repository of what has occurred and can be used as both legal and process quality evidence.

Obviously, once a workflow has been instantiated, its description is possible through the use of a flowchart or process map. Even complex workflows can be represented this way. However, the challenge we face in healthcare situations is not just the intricacy of a process, but also the explosion in the number of potential sequences enacted in response to the large number of decision factors. Representing all possible actions and sequences becomes at least burdensome if not intractable. At the least, the framework we have developed provides the basis for an adaptive system.

Feasibility

Is the process being described feasible? Can we build a workflow based on the principles just described? The book by Jackson and Twaddle describes such a database system which is a commercial product designed to capture workflow in a commercial environment. Our research group has also looked at adapting the systems described in Jackson and Twaddle to the health field. The details of this research can be found in [16]. These results give us confidence that the approach we have described is possible in a healthcare setting.

Related Work

A number of approaches to the representation and realization of dynamic workflow have been proposed. As examples, workflow languages such as BPEL4WS [11], WSFL [12], and XLANG [13] have evolved. However, these solutions are limited in their ability to represent dynamic situations and they do not consider context information as transition constraints of services. A situation-adaptable workflow system was described in [14] that can support service demands generated dynamically in a business process. This system can dynamically handle a user's requests using open-ended adaptation techniques. Our approach, however, also incorporates context and relies on context attributes to constrain service executions within a database-oriented realization.

Van Aalst et al. in [17] identify four types of workflow (declarative, human-based decision workflows, procedural (static-workflow) and constraint based) and they bind them together into an amalgamated workflow. They then use workflow engines that are available for each type and call upon the specific engine as needed, thus taking an approach of best tool for the job. In the approach described in this paper we argue that one carefully chosen E-R representation works best for all types of workflow thus reducing both the complexity of representation and interpretation.

In this paper, we focus on representation of workflow, although we mention that dynamic (retroactive) workflow does depend on being able to mine the workflow to determine modifications. Thus, being able to mine workflow is a key concept underlying our approach. There has been substantial work in this area by several authors [18, 19, 20, 21].

The work described in this paper and an earlier publication [15] is focusing on ways of thinking about workflow or meta-models for workflow. Other perspectives on meta-models can be found in [22, 23]. These are more slanted toward workflow as just an assembly of services.

Summary and Conclusions

It has become clear that classic, rigid workflows are too constraining to address the complexity and variability of workflows in healthcare environments. We have presented a framework for addressing the dynamic nature of healthcare workflows in a 'computationally tractable' form, implementing a new understanding of dynamic workflow through the use of concepts from database theory, inference engines and service oriented architectures.

This work, together with work in context informed workflow can be found in greater detail in [15, 16].

Acknowledgements

This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Agfa Healthcare

References

1. Johnson N. Simply Complexity: a clear guide to complexity theory. Oneworld Publications. 2007. ISBN 978-1-85168-630-8.
2. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns, Distributed and Parallel Databases, 14(3), pp. 5-51, July 2003.

3. Cowan DD et al. Software System Generation from an Enterprise Service Model. D.R. Cheriton School of Computer Science Report CS-2007-04. http://www.cs.uwaterloo.ca/research/tr/2007/(last accessed Dec. 16, 2009).
4. Jackson M and Twaddle G. Business Process Implementation: Building Workflow Systems. Addison-Wesley, 1997.
5. Chen PP-S. The entity-relationship model: toward a unified view of data. ACM Transactions on Database Systems 1. 1976; 1:9-36.
6. Business Process Execution Language for Web Services version 1.1 http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ (accessed Dec. 16, 2009).
7. Rumbaugh J, Jacobson I, Booch G. The Unified Modeling language Reference Manual. Addison Wesley 2004.
8. XML Metadata Interchange (XMI), v2.1 http://www.omg.org/technology/documents/formal/xmi.htm (accessed Dec. 16, 2009).
9. Extensible Stylesheet Language (XSL) http://www.w3.org/Style/XSL/ (last accessed Dec. 16, 2009).
10. Alencar PSC, Oliveira T, Cowan DD, Mulholland DW. Towards Monitored Data Consistency and Business Processing Based on Declarative Software Agents. Software Engineering for Large-Scale Multi-Agent Systems – Research Issues and Practical Applications. Garcia A, Lucena C, et al. (Eds), Lecture Notes in Computer Science (LNCS), 2003; 2603:267-284.
11. Andrews T, Curbera F, Golan Y. Business Process Execution Language for Web Services, BEA Systems, version 1.1., 2003.
12. Leymann F. Web Services Flow Language (WSFL 1.0). IBM 2001.
13. Thatte S. XLANG Web Services for Business Process Design. Microsoft Corporation 2001.
14. Vieira P, Rito-Silva A. Adaptive workflow management in WorkSCo, 16th International Workshop on Database and Expert Systems Applications (DEXA05). 2005; 640-645.
15. Fisher L. 2008 Business Process Management and Workflow Handbook, Workflow Management Coalition. Chapter. Covvey HD, Cowan DD, Alencar P, Malyk W, So J, Henriques D, Fenton S. The Representation of Dynamic, Context-Informed Workflow” 2008, pages 273-286.
16. Malyk W.J., Towards an Extensible Workflow System for Healthcare, MMath thesis, University of Waterloo, 2006. http://gradworks.umi.com/MR/23/MR23747.html.
17. van der Aalst, W. M. P., Adams, M., ter Hofstede, A. H. M., Pesic, M. and Schonenberg, H. 2009. “Flexibility As a Service”, Proceedings of the 1st International Workshop on Mobile Business Collaboration (MBC’09).
18. Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, Luigi Pontieri, Domenico Saccà. Mining Constrained Graphs: The Case of Workflow Systems. In Proceedings of Constraint-Based Mining and Inductive Databases'2004. pp.155~171
19. R. Agrawal, D. Gunopulos, F. Leymann. “Mining Process Models from Workflow Logs”, in Sixth International Conference on Extending Database Technology, 1998, pp. 469–483.
20. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, J. Maruster, G. Schimm, A.J.M.M. Weijters, “Workflow Mining: A Survey of Issues and Approaches”, Internal Report, 2002.
21. L. Maruster, W.M.P. van der Aalst, A.J.M.M. Weijters, A. van den Bosch, W. Daelemans, “Automated discovery of workflow models from hospital data”, in: B. Kr€oose, M. de Rijke,

G. Schreiber, M. van Someren (Eds.), Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001), 2001, pp. 183–190.

22. JJ. Dubray. “Business Process Metamodels and Services” from The Workflow Handbook 2005, Future Strategies Inc., 2005, ISBN 0-9703509-8-8, pp. 159-178.
23. C. Lawrence. “Integrated Function and Workflow” from The Workflow Handbook 2005, Future Strategies Inc., 2005, ISBN 0-9703509-8-8, pp. 31-52.
24. K.D. Swenson. “ASAP/Wf-XML 2.0 Cookbook - Updated” from The Workflow Handbook 2005, Future Strategies Inc., 2005, ISBN 0-9703509-8-8, pp. 257-280.
25. JJ. Dubray. “Business Process Metamodels and Services” from The Workflow Handbook 2005, Future Strategies Inc., 2005, ISBN 0-9703509-8-8, pp. 159-178.
26. http://www.xpdl.org/nugen/p/adaptive-case-management/public.htm.