Institut Eurécom
Department of Corporate Communications
2229, route des Crêtes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-09-231

A Document-based Dynamic Workflow System

April 10th, 2009
Last update April 10th, 2009

Mohammad Ashiqur Rahaman and Yves Roudier

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {mohammad.rahaman,yves.roudier}@eurecom.fr

1Institut Eurécom’s research is partially supported by its industrial members: BMW Group Research & Technology - BMW Group Company, Bouygues Télécom, Cisco Systems, France Télécom, Hitachi Europe, SFR, Sharp, STMicroelectronics, Swisscom, Thales.
A Document-based Dynamic Workflow System

Mohammad Ashiqur Rahaman, and Yves Roudier

Abstract

A typical workflow is specified by a set of predefined tasks executed in a sequence flow in which business objects represented as documents are handled to reach a business goal. Workflow actors with precise roles handle documents reflecting the results of executed tasks. However, increasing agile nature of business processes implies neither the potential tasks nor their sequence flow can be defined a priori. In this context, a document-based workflow (DocWF) in which the handled documents may preserve the statefulness of a business process helping an actor to define potential tasks and their sequence flow at runtime.

In this paper, we present a formal model of a DocWF to address agile business processes. We first classify the associated entities and their relationships in a meta-model. To this end, a formal execution model of a DocWF system is provided. We illustrate a dynamic execution with an electronic health record (EHR) generation workflow that handles an XML document of EHR.

Index Terms

Potential Task, Knowledge-base, Document-based Workflow
Contents

1 Introduction 1

2 Document-based Workflow (DocWF) 3
   2.1 DocWF Terminologies . . . . . . . . . . . . . . . . . . . . . . 3
   2.2 DocWF Meta Model . . . . . . . . . . . . . . . . . . . . . . . . 4

3 A Formal DocWF Model 6
   3.1 DocWF Model Definitions . . . . . . . . . . . . . . . . . . . . . 7
   3.2 DocWF Status Transition Rules . . . . . . . . . . . . . . . . . . 11
   3.3 A DocWF Execution Example . . . . . . . . . . . . . . . . . . . 14
   3.4 Workflow Patterns in DocWF . . . . . . . . . . . . . . . . . . . 16

4 Related Work 17

5 Conclusion 18
List of Figures

1  Document-based Workflow (DocWF) Terminologies .............................................. 2
2  DocWF Meta model ................................................................................................. 4
3  Document Meta-model ............................................................................................. 5
4  A fictitious electronic health record (EHR) document of a patient should be handled in a EHR generation workflow. Different portions of the EHR \((d_1, d_2, d_3, d_4, d_5)\) need to be handled during workflow execution ................................................................. 6
5  A behaviour model of a DocWF design time objects (final states are omitted) ................................................................. 8
6  A behaviour model of a DocWF run time objects after receipt of documents/document portions (final states are omitted) ............... 9
7  A dynamic execution of Electronic Health Record (EHR) generation workflow ........................................................................... 11
1 Introduction

Today’s business processes with certain business goals are executed in dynamic, uncertain and data-centric environments. Uncertainty includes diverse aspects of a workflow and its associated environment such as changes in the business goal and data, execution errors such as unavailability of workflow actors, violations of policies, lack of evidence of workflow actors’ eligibility (e.g., missing credentials), or even incompletion of goals and data that may not yet reflected in the handled business documents. These uncertainties amount to lack of statefulness of workflow execution and thus inability to define and execute suitable tasks at runtime.

Unlike task-based workflow process that is structured a priori in a sequence of task flow to decide what should be the task in a business process, a DocWF focuses on what can be the task to achieve a business goal rather than what should be done. A user (i.e., a workflow actor), with a comprehensive knowledge of the business process domain, proactively determines how the goal can be reached. The role of a DocWF system is to assist the actors rather than instructing them.

Existing knowledgebase (KB) containing legacy business processes, best practices etc. of an organization can be a recipe for future business process needs. For example, in a conference proceeding process the tasks in a sequence flow (i.e., publishing call for papers by the organizers → submitting several papers by the authors → reviewing the papers by PC members → selecting accepted papers by the PC chairs → sending notifications with the reviews to the authors → submitting the camera-ready versions by the authors and finally → publishing those in the proceedings) can be used as a recipe for a research project proposal granting workflow as both business processes have similar goals (i.e., to be successful to publish papers and to have approvals of research funds respectively) with similar tasks. However, in agile business scenarios like health care, natural calamities one can not decide upon suitable tasks a priori as each incident has individual peculiarities. Many authors also argue that existing workflow management systems [1–8] proved to be limited in dealing with such dynamic nature of a workflow. A simple workflow model may fail or block in an exceptional situation unless the user gets into the backend [9] which is infeasible as actors are forced to bypass the workflow systems quite frequently. To address this issue, several research communities [1–3, 7] are devoted to devise techniques for flexible workflow management.

In business processes, actors primarily handle business documents or portions thereof from their creation to destruction, archiving or version consolidation (collectively termed as document handling). For the conference proceeding process, several actors (i.e., organizers, PC members, authors and attendees) handle research papers in a sequence of tasks. In addition to document handling in such a DocWF system needs to tackle several security aspects. For example: a PC member may not be allowed to review a paper of which he is a co-author (i.e., conflict of interest), authors affiliations should be anonymous (i.e., access control), the same paper must be reviewed by at least three reviewers and once the PC chair takes the decision
of acceptance/rejection no new reviews of the same paper are not considered (i.e. usage control).

To cope with the agile business nature, recent industry adoption of SOA-based business process tools and frameworks [10,11] emphasize reuse of existing capabilities and their governance as key factors. Regarding reuse, our approach utilizes the KB of an organization to determine whether existing capabilities can be a help to solve a current business need. Governance rules (e.g. legal) for handling documents and their security can be supported in that approach with a set of rules forming a policy-base. We believe that a document-based workflow system must support (1) runtime definition of tasks and their sequence flow and (2) document handling and security aspects. For (1) our DocWF system allows runtime definition of tasks and their sequence flow based on the statefulness of a DocWF execution captured as a DocWF status (see Section 3). A BPEL analogy would be: unlike invoking services in a pre-defined sequence of <invoke> elements at runtime a DocWF system defines those before invoking. A task can be essentially a human task or a web service. Document handling and security aspects must be enforced in distributed fashion as documents can be handled by services hosted in different business boundaries. In our previous work [12], we developed a distributed access control enforcement mechanisms for XML documents relying on a distributed key agreement protocol which is equally applicable in a workflow context. In [13], we proposed an XML tree comparison technique that can be directly used for versioning and archiving of XML documents.

In this paper, we provide a formal foundation of a DocWF system by focusing on its dynamic aspects. Section 2 introduces different terminologies of a DocWF followed by a meta-model showing the relationships of different entities in such a system. Section 3 describes the formal model of DocWF system and illustrates the model with an example. Section 4 positions our work with related literature and finally Section 5 concludes the paper.
2 Document-based Workflow (DocWF)

A DocWF is a document oriented business process that is capable of assisting workflow actors at runtime to decide upon (i.e. define and execute) potential tasks to reach a business goal.

2.1 DocWF Terminologies

Figure 1 illustrates the document-based workflow terminologies. A DocWF system has to achieve a business goal from which a set of subgoals can be derived. Goals are abstract definitions of potential tasks that will be executed by actors. In the conference proceeding process, the business goal is to organize the conference whereas a subgoal is to publish papers in the proceedings of the conference by authors (i.e. actors).

The main concept of a DocWF system is the documents in which data are instantiated during a workflow execution. A goal achievement instantiates data flow in documents. Based on the achieved and remaining goals and consultation of the KB, potential tasks are defined and their execution handles the documents which is reflected either by creating new documents or updating existing documents by legitimate actors who possess credentials (proving the roles).

A goal achievement implicitly may enable subsequent goals and documents to be handled until the business goal is not achieved. Actors may trigger tasks from a pool of defined tasks called recipe tasks in a predefined sequence which we term as recipe flow, if the goals can be achieved by those recipe tasks in that recipe flow. Examples of such DocWF systems are: conference proceeding, research proposal granting, employee performance evaluation, leave approval, tax evaluation processes.

**Example 1** A research proposal granting process can follow the conference proceeding process as a recipe process. Several actors (i.e. proposal writers, reviewers, granting authority) can handle research proposals in a sequence of recipe tasks: publishing call for research proposals by the organizers → submitting several proposals by the authors → reviewing the proposals by reviewers → selecting accepted proposals by the authorities → sending notifications with the reviews to the authors → submitting the final versions by the authors and finally → granting funds for the accepted research projects.

However, there might be exceptional situations as indicated before that can not be handled by any recipe process (from KB). In this context, potential tasks (i.e. free tasks) and their sequence flow (i.e. free flow) may need to be defined at runtime to achieve the business goal. Consider, in a clinical environment, a doctor orders diagnostic tests for a patient, but cannot wait for the results in case of an emergency. Thus she may start treatment for the emergency patient. As soon as the test results arrive (in the middle of the current treatment), she might need to achieve a new
goal requiring a completely different treatment (i.e. free tasks) depending on the result.

Workflow actors are such stakeholders with precise roles in a business process who may also need to comply with security requirements. An actor possesses domain knowledge but may require assistance in the form of statefulness in business documents and DocWF execution to decide upon potential tasks. While we consider the role hierarchy, role assignments and security aspects should be integrated for a secure DocWF system, we do not discuss further these issues other than showing their relationships in the DocWF meta-model.

### 2.2 DocWF Meta Model

An object oriented approach, in particular UML constructs, are used to describe the relationships of DocWF entities as to deal with a complex DocWF system. While the DocWF meta model contains design and run time entities separated by a straight line, the DocWF role class is significant with respect to security in both design and run time (see Figure 2).

DocWF is the main class in the DocWF meta model. A DocWF consists of a business goal, documents, potential tasks and their sequence flow. One goal can derive other subgoals and achievement of a goal instantiates data flow into documents. These properties are represented by a recursive association of goals that form a goal precedence and another association between goal and document respectively (Figure 2).

As a DocWF is document centric an associated document meta-model showing the relationships of involved entities in document handling, i.e. document concepts, actors, document portion, version is provided in Figure 3. Document concepts
and their relationship represent the business domain semantics which then can be mapped to document portions []. For example, in a patient health care domain, a business concept 'treatment' can be mapped to <Medicine>, <Therapy> and <Surgery> related XML fragments (see Figure 4).

To facilitate an actor to decide upon a potential task the organizational best practices, legacy business process models, document semantics, execution history of processes and policies are captured in the knowledge base (KB) class. As KB class is not associated to other entities in the meta-model, KB can be consulted at both design and run time of a DocWF.

Potential task definitions are associated with document definitions. In particular, one task may be associated to at least one document or document portion and vice versa. Abstract definitions of potential tasks are made concrete in further definitions of recipe and free tasks. Such concrete definitions of tasks can be realized by the document semantics (e.g. domain concepts, document structure). Consequently, Successful execution of a potential task handles one or more documents/document portions. The sequence flow among such tasks can also be defined at run time (i.e. free flow). Intuitively, one or more tasks can be in one sequence flow and vice versa.

The recipe and free tasks (and their flows) are important entities for agile business processes. The recipe tasks and their sequence flow i.e. free flow taken from other business processes of the KB might be directly applicable to achieve the goals of a DocWF and as such promote the reuse of existing services. However, the free tasks and free flow elements allow an actor to define tasks and their flow at run time depending on the current business needs and thus to operate in an uncertain environment.

There are multiple roles associated with a given DocWF process. In particular, a role type for defining goals, document semantics and policy (i.e. class DocWF role type) and a role type for defining a task (i.e. class task role type) and its sequence flow (i.e. class flow role type). The association classes link the DocWF role class with other meta elements, i.e. Goal, Document, recipe task, recipe flow, free task, free flow.
Table 1: Differences between task-based workflow, case handling and document-based workflow

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus</td>
<td>Task</td>
<td>Whole case</td>
<td>Document data</td>
</tr>
<tr>
<td>Orientation</td>
<td>Task</td>
<td>Whole case</td>
<td>Documents/business goal</td>
</tr>
<tr>
<td>Primary perspective</td>
<td>Sequence flow</td>
<td>Case data</td>
<td>Document data</td>
</tr>
<tr>
<td>Task and Sequence flow definition</td>
<td>A priori</td>
<td>A priori</td>
<td>Runtime</td>
</tr>
<tr>
<td>Process statefulness</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reuse of processes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Document handling</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Access control focus</td>
<td>Task</td>
<td>Task and case data</td>
<td>for documents</td>
</tr>
<tr>
<td>Document usage control features</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 4: A fictitious electronic health record (EHR) document of a patient should be handled in a EHR generation workflow. Different portions of the EHR \( (d_1, d_2, d_3, d_4, d_5) \) need to be handled during workflow execution.

3 A Formal DocWF Model

In this section, we present a formal model of a DocWF system followed by its dynamic execution semantics with an example of EHR generation workflow.
3.1 DocWF Model Definitions

We begin with design time objects of a DocWF system followed by runtime objects for which behaviour models are depicted in state chart diagrams in Figure 5 and 6 respectively. Dotted arrow means that an enabled status may enable other object’s status.

Goals and documents: Given a business goal \( G \) of a DocWF, the derived set of subgoals is denoted as \( G = \{g_1, g_2, \ldots, g_m\} \) for \( m \geq 1 \) subgoals. Let \( g_i \) and \( g_j \) be two subgoals of \( G \). If \( g_j \) can not be achieved unless \( g_i \) is achieved then there is a goal precedence between them, denoted as \( g_j \triangleright g_i \). Let \( D \) be a set of documents/document portions denoted by \( D = \{d_1, d_2, \ldots, d_n\} \), for \( n \geq 1 \), that need to be handled when a DocWF is executing. For any two documents or document portions \( d_i \) and \( d_j \), if \( d_j \) can not be handled before \( d_i \) is handled then there exists a precedence between \( d_i \) and \( d_j \), denoted as \( d_j \triangleright d_i \). For no such precedence relation between two goals and two documents/document portions: \( g_j \prec g_i \) and \( d_j \prec d_i \) respectively. Intuitively, pairs of goal and documents/document portions may also exhibit precedence relationship which implicitly sets constraints for document handling with respect to a goal achievement. For two such pairs one can be a successor of the other if for the goal and all the documents/document portions of the former, i.e. \( g_j \) and \( \forall d_j, g_j \triangleright g_i \) and \( d_j \triangleright d_i \) hold where \( g_i \) and \( d_i \) are the goal and documents/document portions of the other.

Policy: A set of rules each denoted by \( p_i \) representing document handling constraints and security requirements form the policy base of a DocWF system. One or more such rules may be enabled and need to be enforced for a goal achievement. Moreover, one rule may infer new rules as evaluation of a rule may enable or derive other rules at runtime. For example, in Figure 4, a rule '\( d_1 \) and \( d_2 \) should not be updated by the same actor' meaning once one actor updates any of the documents, her credential for the other document may be invalidated. In practice a policy rule can specify any legal and security requirements. As we focus on the dynamicity of a DocWF system, we consider the document handling constraints as the simplest form of a policy rule in the rest of the paper. As such, if a policy rule \( p_i = \{g_i, \{d_i\}\} \) is enforced a set of inferred policy rules e.g. \( p_j = \{g_j, \{d_j\}\} \) are enabled and denoted as \( p_j \triangleright p_i \). Two policy rules having no precedence is denoted by \( p_j \prec p_i \).

Example 2 In Figure 4 the business goal of EHR generation workflow is to generate an EHR for a patient that derives a set of goals, \( G = \{g_1, g_2, g_3, g_4\} \); \( g_4 \triangleright g_3 > g_2 \) but \( g_1 \triangleright g_2 \); \( D = \{d_1, d_2, d_3, d_4, d_5\} \); \( d_5 \triangleright d_4 \triangleright d_3 \) but \( d_2 \triangleright d_1 \triangleright \{d_1, d_2\} \) and \( d_1 \triangleright \{d_2, d_3\} \).

Policies are \( p_1 = \{g_1, \{d_1, d_2\}\} \); \( p_2 = \{g_1, \{d_2\}\} \); \( p_3 = \{g_2, \{d_3\}\} \); \( p_4 = \{g_3, \{d_4\}\} \) and \( p_5 = \{g_4, \{d_5\}\} \) where \( p_5 \triangleright p_4 \triangleright p_3 \) but \( p_2 \triangleright p_1 \triangleright p_1 \triangleright \{d_1 \triangleright d_2\} \).

Knowledge-base(KB): It is defined as a collection of organizational best practices, document semantics, process models, e.g. BPMN, process execution history,
Definition 1 Potential tasks: Let $T_i$ be an executed task which enforced policy rule $p_i$, and $p_j$ be an enabled rule after $T_i$’s execution. A new task $T_j$ is a potential successor task of $T_i$, denoted as $T_j > T_i$ if either (1) both $T_i$ and $T_j$ are chosen from a recipe process such that $T_j > T_i$ in the recipe process or (2) if $p_j > p_i$ holds. If there is no task $T_k$ such that $T_j > T_k > T_i$ then $T_j$ is the immediate potential successor task of $T_i$, denoted as $f_{ij} = 1$ and otherwise $f_{ij} = 0$.

Two potential tasks $T_i$ and $T_j$ are independent when neither $T_j > T_i$ nor $T_i > T_j$. A directed graph $G = (T, >)$ represents this precedence relationship among potential tasks of $T$, where each potential task is a node in the graph and a directed edge from a node $T_i$ to node $T_j$ satisfies $f_{ij} = 1$. In the rest of the paper, the term ‘task’ refers to ‘potential task’ unless otherwise stated.

Policy violation: It determines whether (1) an executed task enforced the policy rules for a goal by handling the associated documents/document portions, (2) two documents/document portions can be handled in parallel or in a sequence. For (1) the success of a rule enforcement can be verified by either manually observing the documents or an automated verification service. For (2) if there exists no precedence relation between any two enabled rules. Then a policy violation condition for two potential tasks $T_i$ and $T_j$ is denoted by $v_{ij} = v_{ji} = \{0, 1\}$. $d_i$ and $d_j$ can be handled in parallel if (1) is successful and $T_i$ and $T_j$ are independent, denoted by $v_{ij} = v_{ji} = 0$. If neither (1) is successful nor $d_j < d_i$ then only one of the documents/document portions can be handled in one task and the other is disabled for the other task, denoted as $v_{ij} = v_{ji} = 1$. For example in Figure 4, the document portions $d_1$ and $d_2$ can be handled in two parallel tasks: $T_1 =$’updating patient information’ and $T_2 =$’updating her insurance information’ as $p_1 < p_2$. However, the document portion $d_4$ can not be handled before handling of $d_3$ to achieve the goal $g_2 =$’Symptoms are identified’ as $p_4 > p_3$ and thus $T_4 > T_3$.

Definition 2 The preset of a potential task, denoted by $^*T_k$, is a set of document handling constraints $^*T_k = \{(g_i, \{d_j\})|g_i$ is the goal of $T_k$ or $g_k > g_i$ and $d_j$ are the documents/document portions that are handled in tasks $T_i$ where $T_k > T_i$ or $d_k > d_j\}.$

Definition 3 The postset of an executed task, denoted by $T_k*$, is a set of document handling constraints $T_k^* = \{(g_i, \{d_j\})|g_i$ is the goal of $T_k$ or $g_i > g_k$ and $d_j$ are the documents/document portions that need to be handled in task $T_k$ or $d_j > d_k\}.$
Figure 6: A behaviour model of a DocWF run time objects after receipt of documents/document portions (final states are omitted).

The **preset** of a task $T_k$ is the collection of achieved goals and handled documents/document portions just before executing the task $T_k$. On the other hand, the **postset** of an executed task represents the remaining goals to be achieved and the documents that need to be handled for the successful execution of the workflow. The facts of the KB and the dynamic knowledge of $^\ast T_k$, $T_k^\ast$ and policy violation conditions enable an workflow actor to decide upon potential tasks (i.e. recipe or free) for the remaining goals and documents/document portions to be handled (i.e. what can be done rather than what should be done).

**Workflow Pattern:** For $|^\ast T_k| \geq 1$, the potential execution of a task $T_k$ will not occur until all the subgoals are achieved by handling the associated documents. This follows the classical AND workflow pattern [14]. An alternate of this is to allow a potential task to be executed when some (not all) of the preceding goals are achieved meaning some documents are handled. Unlike the policy violation condition, this is a relaxed constraint comparable to an OR workflow pattern. Intuitively, the OR workflow pattern allows more flexibility than the AND pattern which can also be seen as a special case of OR pattern. As such we define a feasible condition that enables this OR pattern.

**Feasible Condition:** Let $^\ast T_k = \{(g_{k1}, \{d_{k1}\}), (g_{k2}, \{d_{k2}\}), ...,(g_{kl}, \{d_{kl}\})\}, l \geq 1$. We define a feasible condition to execute a task, denoted as $C(T_k) = \{C_1, C_2, ...C_m\}, m \geq 1$ such that

1. $C_i \subseteq ^\ast T_k, i = 1, 2, ...m$ i.e. $C(T_k)$ is a set of subsets of $^\ast T_k$.

2. For any two $C_i, C_j$ where $i \neq j$ and $d_i \cap d_j = \emptyset$ for $i, j \in \{1, 2, ...m\}$.
   This implies that the same document/document portion is not handled in two different subsets.

3. $T_k$ is potentially executable iff all documents/document portions in $C_i \in C(T_k)$ are handled in achieving goals of $C_i \in C(T_k)$. So, a task $T_k$ can be triggered by any subset of $C(T_k)$, but only if all documents/document portions in that subset are successfully handled.
A feasible condition serves two purposes: (1) it allows an actor to avoid blocking situation due to uncertainty and failures, such as a crucial goal would not be achieved by some potential tasks because some previous goals are not achieved. Making the documents/document portions that are being handled in unachieved subgoals as part of \( C(T_k) \) enables a workflow actor to continue in a DocWF execution as some other achieved goals will let the item (3) of feasible condition be true. (2) It allows an actor to keep track of unhandled documents in case those are required to be handled again for further goal achievement. For simplicity, a feasible condition \( C \) is shown as a set of executed tasks instead of goal and documents pairs in the rest of the paper.

To make a DocWF execution stateful, the modeling elements (e.g. \( T_k, T^*, C \)) can be embedded as meta-data in the handled documents. However, handled documents/document portions themselves are testimony of statefulness as the instantiated data in the documents are the result of a DocWF execution.

**Task state:** The state of a task (i.e. executed or potential) \( T_i \), denoted as \( S(T_i) \), in a DocWF execution is an integer value in \( \{0, 1, 2, 3\} \) such that:

1. \( S(T_i) = 0 \), i.e. \( T_i \) is not executed before and not a potential task.
2. \( S(T_i) = 1 \), i.e. \( T_i \) is not executed before and is a potential task.
3. \( S(T_i) = 2 \), i.e. \( T_i \) was successfully executed before and not a potential task.
4. \( S(T_i) = 3 \), i.e. \( T_i \) was successfully executed before and is a potential task.

For any new task (i.e. recipe or free), its state value indicates whether it can be a potential task or not. By the above definition of task state, only those potential tasks having a value either 1 or 3 can be executed.

Based on the task state value we define a DocWF status that represents current execution status of a DocWF system.

**Definition 4** DocWF status: A DocWF status of a DocWF execution is described as an array of the state values of executed tasks. Let \( S \) be a DocWF status of \( p \) executed tasks then \( S = \{ S(T_1), S(T_2), ..., S(T_p) \} \).

Now, we formally define our DocWF model.

**Definition 5** DocWF: A document-based workflow is a tuple \( \text{DocWF} = (G, D, T, F, P, C, S_I, S_F) \), where

1. \( G = \{ g_1, g_2, ..., g_m \}, m \geq 1 \), is a set of goals derived from a business goal \( G \).
2. \( D = \{ d_1, d_2, ..., d_r \}, r \geq 1 \), is a set of documents/document portions that need to be handled.
3. \( T = \{ T_1, T_2, ..., T_n \}, n \geq 1 \), is an incremental set of executed tasks where a task is defined as either a free or a recipe task.
3.2 DocWF Status Transition Rules

The dynamic execution of a DocWF system is governed by a set of status transition rules based on which the incremental set of executed tasks $T$ is built. Any chosen task (i.e., recipe or free) has an initial state value of 0. Let $T_i$ be a potential task and there is no task $T_k$ such that $T_i > T_k$ then the state value of $T_i$ is $S_a(T_i) = 1$ (Rule A). If the new status resulted from a successful execution of $T_i$ is $S_b$, then the execution of $T_i$ is denoted by $S_a(T_i)S_b$. This implies $S_b(T_i) \in \{2, 3\}$ (Rule B).

Now, $\forall T_j$ (i.e., potential tasks) such that $T_j > T_i$, the state value of $T_j$ at DocWF status $S_b$ is determined by (1) If $T_j = T_i$ then $f_{ij} = 1$ and the state value
of $T_j$ at new status is: $S_b(T_j) = 3$; (Rule C) (2) If $T_j \neq T_i$ then the state value of $T_j$ at new DocWF status $S_b$ depends on the state value of $T_i$ at the status $S_b$. There are four possibilities depending on the policy violation and feasible condition:

- **Rule I** - $S_b(T_i) = 2$ and $v_{ij} = v_{ji} = 1$:
  
  (a) If $\exists C' \in C(T_j)$ such that $S_b(T_k) \in \{2, 3\}$ for any $T_k \in C'$, then $f_{ij} = 1$ and $S_b(T_j) = 1$;
  
  (b) Else if $\exists k, j T_{k=j} \in C(T_j)$ such that $S_b(T_{k=j}) = 2$ then $f_{ij} = 1$ and $S_b(T_j) = 2$;
  
  (c) Else $f_{ij} = 0$ and $S_b(T_j) = 0$.

- **Rule II** - $S_b(T_i) = 2$ and $v_{ij} = v_{ji} = 0$:
  
  (a) If $S_a(T_j) = 1$ then $S_b(T_j) = 2$;
  
  (b) Else if $\exists k, j T_{k=j} \in C(T_j)$ such that $S_b(T_{k=j}) = 2$ then $f_{ij} = 1$ and $S_b(T_j) = 3$;
  
  (c) Else $f_{ij} = 1$ and $S_b(T_j) = 1$.

- **Rule III** - $S_b(T_i) = 3$ and $v_{ij} = v_{ji} = 1$:
  
  (a) If $\exists C' \in C(T_j)$ such that $S_b(T_k) \in \{2, 3\}$ for any $T_k \in C'$, then $f_{ij} = 1$ and $S_b(T_j) = 1$;
  
  (b) Else if $\exists k, j T_{k=j} \in C(T_j)$ such that $S_b(T_{k=j}) = 2$ then $f_{ij} = 1$ and $S_b(T_j) = 2$;
  
  (c) Else $f_{ij} = 0$ and $S_b(T_j) = 0$.

- **Rule IV** - $S_b(T_i) = 3$ and $v_{ij} = v_{ji} = 0$:
  
  (a) If $S_a(T_j) = 1$ then $S_b(T_j) = 2$;
  
  (b) Else if $\exists k, j T_{k=j} \in C(T_j)$ such that $S_b(T_{k=j}) = 2$ then $f_{ij} = 1$ and $S_b(T_j) = 3$;
  
  (c) Else $f_{ij} = 1$ and $S_b(T_j) = 1$.

According to the above transition rules, a potential task $T_j$’s state value at a new DocWF status $S_b$ is 0 iff one of the following holds:

- $T_j$ is just chosen as a recipe or a free task.

- If the state value of the task $T_i$ in $S_b$ is 2, and the violation condition $v_{ij} = v_{ji}$ is 1; meaning the associated documents/document portions can not be handled as feasible condition is false in the current DocWF status (Rule I(c)).

- If the state value of the task $T_i$ in $S_b$ is 3, and the violation condition $v_{ij}$ is 1; meaning the associated document/document portion can not be handled as feasible condition is false in the current DocWF status (Rule III(c)).

A potential task $T_j$’s state value at a new DocWF status $S_b$ is 1 iff one of the following holds:

- If the state value of the task $T_j$ in $S_a$ was 0; meaning it is ready to be executed. (Rule A)
• If the state value of the task $T_i$ in $S_b$ is 2, and the policy violation condition $v_{ij}$ is 0; meaning the associated document/document portion can be handled immediately (Rule II (c)).

• If the state value of the task $T_i$ in $S_b$ is 2, and the policy violation condition $v_{ij} = v_{ji}$ is 1 and at least all tasks in one of $T_j$’s feasible condition sets is successfully executed; meaning if some of the preceding goals are achieved if not all then a subsequent document/document portion can be handled for an enabled goal (Rule I(a)).

• If the state value of the task $T_i$ in $S_b$ is 3, and the policy violation condition $v_{ij}$ is 0; meaning the associated document/document portion can be handled immediately (Rule IV (c)).

• If the state value of the task $T_i$ in $S_b$ is 3, and the policy violation condition $v_{ij} = v_{ji}$ is 1 and at least all tasks in one of $T_j$’s feasible condition sets is successfully executed; meaning if some of the preceding goals are achieved if not all then a subsequent documents/document portions can be handled for an enabled goal (Rule III(a)).

A potential task $T_j$’s state value at a new DocWF status $S_b$ is 2 iff one of the following holds:

• If the state value of the task $T_i$ in $S_b$ is 1, and the goal is achieved after its execution; meaning associated document/document portions are handled completely (Rule B).

• If the state value of the task $T_i$ in $S_b$ is 2 or 3, and the policy violation condition $v_{ij} = v_{ji}$ is 1 but the task $T_j$ is executed before; implies the goal is achieved by handling associated documents/document portions (Rule I(b) and Rule III(b) respectively).

• If the state value of the task $T_i$ in $S_b$ is 2 or 3, and the policy violation condition $v_{ij}$ is 0 and previously the state value of $T_j$ was 0; meaning the execution of $T_j$ should handle the associated documents/document portions and the goal should be achieved (Rule II(a) and IV(a) respectively).

A potential task $T_j$’s state value at a new DocWF status $S_b$ is 3 if the following holds:

• If the state value of the task $T_i$ in $S_b$ is 1, and the goal is not achieved yet after its execution; meaning associated document/document portions are handled partially and the same task may need to be executed later (Rule B).

• If the state value of the task $T_i$ in $S_b$ is 2, and the policy violation condition $v_{ij} = v_{ji}$ is 1 and the task $T_j$ is executed before; but the goal is not achieved yet; implies the same task may need to be executed later to handle associated document/document portions (Rule II(b)).
• If the state value of the task $T_i$ in $S_0$ is 3, and the policy violation condition $v_{ij}$ is 0 and the task $T_j$ is executed before; but the goal is not achieved yet; implies the same task may need to be executed later to handle associated document/document portions (Rule IV(b)).

• If the same task $T_i$ is potentially executable leading to a self loop (Rule C).

The transition rules take uncertainty into consideration during $DocWF$ execution to decide upon a potential task to execute. For example, in the electronic health care record generation workflow of Figure 4 it is possible to perform additional diagnosis tests, i.e. $T_4$ while doctors are doing treatment, i.e. $T_5$; even doctors can postpone the treatment and asks for additional diagnosis tests for further treatment (explained in the following).

3.3 A $DocWF$ Execution Example

We now illustrate an execution example of the fictitious electronic health record generation workflow (EHR) by applying our $DocWF$ modeling approach (see Figure 4 and 7). The business goal $G$ of the EHR workflow is to generate a complete EHR document $D$ containing patient information $d_1$, patient’s insurance information $d_2$, the symptoms of the patient $d_3$, possible diagnosis test results $d_4$ and treatment information $d_5$, i.e. $D = \{d_1, d_2, d_3, d_4, d_5\}$. The workflow actors are hospital administrative employees, nurses, pathologists and doctors. Human actors are assumed to illustrate transitions. However, any automated service can be an actor as long as transition rules are implemented in the service.

We show each transition by instantiating the DocWF modeling elements (i.e. executed tasks, sequence flow matrix $F$ and policy violation matrix $P$ which shows the enabled policy rules precedence relation and in turn determines the values of feasible condition $C$), enabled policy rules and $DocWF$ status. Assume a new patient enters into a hospital, denoted by the task $T_0$ which initiates the EHR generation process (Figure 7(a)).

\[
p_0 = \emptyset, T = \{T_0\}, F = \begin{bmatrix} 0 \end{bmatrix}, P = \begin{bmatrix} 0 \end{bmatrix}, C(T_0) = \emptyset, S_1 = S_0(T_0) = 1
\]

The goal $g_1$ is to record the patient particulars into $d_1$ and $d_2$. As the patient is a new one, according to the best practices for a patient management from the KB of the hospital, the administrative employees decide two potential recipe tasks $T_1$, i.e. taking patient information and $T_2$, i.e. taking her insurance details to execute (Figure 7(b)). Moreover, both tasks can be performed in parallel as $p_2 \prec p_1$. This is because $d_1 \prec d_2$ to achieve the same goal $g_1$. As such $f_{01} = f_{02} = 1$ is set for Figure 7(b). The tasks $T_1$ and $T_2$ might not be performed if the patient was previously admitted in the hospital and as such patient’s particulars would have had recorded al-
ready. This can be obvious for instance from a BPMN model of the EHR process in the KB. Now, the DocWF system has:

\[ p_1 = \{g_1, \{d_1, d_2\}\}, p_2 = \{g_1, \{d_2\}\} \]
\[ T = \{T_0, T_1, T_2\} \]

\[
\begin{bmatrix}
0 & 1 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\times & 0 & 0 \\
0 & \times & 0 \\
0 & 0 & \times
\end{bmatrix}
\]

\( C(T_2) = C(T_2) = \{\{T_0\}\} \)
\( S_1(T_2) = 2 \text{ Rule B} \)
\( S_1(T_0) = S_1(T_2) = 1 \text{ (Rule II(c))} \)

While the administrative employees are recording patient particulars in tasks \( T_1 \) and \( T_2 \), the goal \( g_2 \) is to identify the problem symptoms of the patient. Now, the nurses can already record the problem symptoms of the patient, i.e. task \( T_1 \) does not violate. (Figure 7 (c)). If the task \( T_1 \) is executed then it leads to:

\[ p_3 = \{g_2, \{d_1\}\}, p_2 = \{g_1, \{d_2\}\} \]
\[ T = \{T_0, T_1, T_2, T_3\} \]

\[
\begin{bmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\times & 0 & 0 & 0 \\
0 & \times & 0 & 0 \\
0 & 0 & \times & 0 \\
0 & 0 & 0 & \times
\end{bmatrix}
\]

\( C(T_3) = \{\{T_0\}\} \)
\( S_2(T_3) = 2 \text{ (Rule B)} \)
\( S_2(T_1) = 2 \text{ (Rule II(a))} \)
\( S_2(T_2) = 1 \text{ (Rule II(c))} \)
\( S_2(T_3) = 1 \text{ (Rule A)} \)

Note that, state value of the task \( T_2 \) is 1 considering that patient insurance details is not yet recorded in \( d_2 \). Now, the goal, i.e. \( g_3 \) is to perform diagnosis tests for which the potential task \( T_4 \) can not be triggered unless the symptoms are recorded in \( d_3 \), i.e. there is a policy violation condition between the potential tasks \( T_3 \) and \( T_4 \), i.e. \( v_{34} = v_{43} = 1 \). So, when \( d_3 \) is filled in with symptoms, the patient is asked to take some diagnosis test in the pathology department, i.e. \( T_4 \) (Figure 7 (d)). The DocWF system now have:

\[ p_4 = \{g_3, \{d_4\}\} \]
\[ T = \{T_0, T_1, T_2, T_3, T_4\} \]

\[
\begin{bmatrix}
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
\times & 0 & 0 & 0 & 0 \\
0 & \times & 0 & 0 & 0 \\
0 & 0 & \times & 0 & 0 \\
0 & 0 & 0 & \times & 1 \\
0 & 0 & 0 & 1 & \times
\end{bmatrix}
\]

\( C(T_4) = \{\{T_0\}, \{T_3\}, \{T_1, T_2\}\} \)
\( S_3(T_0) = 2 \text{ (Rule B)} \)
\( S_3(T_1) = 2 \text{ (Rule II(a))} \)
\( S_3(T_2) = 1 \text{ (Rule II(c))} \)
\( S_3(T_3) = 3 \text{ (Rule B)} \)
\( S_3(T_4) = 1 \text{ (Rule III-a)} \)

As soon as the diagnosis test results are recorded in \( d_4 \), the goal \( g_4 \) is to perform the treatment that results into updating the document portion \( d_5 \). The responsible doctor may start treatment, i.e. the potential task \( T_5 \), by advising, instructing medicines, therapy, surgery etc. (Figure 7(e)). This results:
\[ p_5 = \{g_4, \{d_5\}\} \]

\[ T = \{T_0, T_1, T_2, T_3, T_4, T_5\} \]

\[ F = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad P = \begin{bmatrix}
\times & 0 & 0 & 0 & 0 & 0 \\
0 & \times & 0 & 0 & 0 & 0 \\
0 & 0 & \times & 0 & 0 & 0 \\
0 & 0 & 0 & \times & 1 & 1 \\
0 & 0 & 0 & 1 & \times & 1 \\
0 & 0 & 0 & 1 & 1 & \times \\
\end{bmatrix} \]

\[ C(T_5) = \{\{T_0\}, \{T_4\}, \{T_3\}, \{T_1, T_2\}\} \]

\[ S_4(T_0) = 2 \text{ (Rule B)} \]

\[ S_4(T_1) = 2 \text{ (Rule II(a))} \]

\[ S_4(T_2) = 2 \text{ (Rule II(a))} \]

\[ S_4(T_3) = 3 \text{ (Rule B)} \]

\[ S_4(T_4) = 3 \text{ (Rule B)} \]

\[ S_4(T_5) = 1 \text{ (Rule III-a)} \]

Note that, \( f_{23} = 1 \) indicating that patient insurance details have been recorded in \( d_2 \). The state value \( S_4(T_4) = 3 \) implies the diagnosis tests can be performed i.e. \( T_4 \) at later time if needed. Now, the doctor may need to start treatment in emergency basis even if some diagnosis results are not recorded in \( d_4 \). However, at a later time the doctor may need some other pathology diagnosis test records before advancing further in the treatment. It indicates that the \( d_4 \) may need to be handled again while the doctor is performing the treatment (see Figure 7 (f)).

\[ p_4 = \{g_3, \{d_4\}\} \]

\[ T = \{T_0, T_1, T_2, T_3, T_4, T_5, T_4\} \]

\[ F = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}, \quad P = \begin{bmatrix}
\times & 0 & 0 & 0 & 0 & 0 \\
0 & \times & 0 & 0 & 0 & 0 \\
0 & 0 & \times & 0 & 0 & 0 \\
0 & 0 & 0 & \times & 1 & 1 \\
0 & 0 & 0 & 1 & \times & 1 \\
0 & 0 & 0 & 1 & 1 & \times \\
\end{bmatrix} \]

\[ C(T_4) = \{\{T_0\}, \{T_3\}, \{T_1, T_2\}, \{T_5\}\} \]

\[ S_5(T_0) = 2 \text{ (Rule B)} \]

\[ S_5(T_1) = 2 \text{ (Rule II(a))} \]

\[ S_5(T_2) = 2 \text{ (Rule II(a))} \]

\[ S_5(T_3) = 3 \text{ (Rule B)} \]

\[ S_5(T_4) = 3 \text{ (Rule B)} \]

\[ S_5(T_5) = 1 \text{ (Rule III-a)} \]

Now, if the treatment is successfully completed after receiving the diagnosis results (Figure 7 (g)), the DoCWF execution will reach a final status as the business goal is achieved.

### 3.4 Workflow Patterns in DocWF

As described before in Sections 3.1 and 3.2, OR pattern is followed in DocWF execution to enable workflow actors to cope with uncertainty and exceptions. This is particularly achieved by the feasible condition that allow actors to define and execute potential tasks based on the current DocWF status as shown in the example. Along this line various workflow patterns [14] such as sequence, parallel, synchronize and loops also prevail in a DocWF execution.
• **Sequential pattern**: In the EHR workflow execution example, tasks $T_3$ and $T_4$ are executed sequentially as $p_4 > p_3$ which in turn makes $T_4 > T_3$. This is represented as $f_{34} = 1$ in the sequence flow matrix.

• **Parallel pattern**: In the example, tasks $T_1$ and $T_2$ are executed in parallel as the goal $g_1$ to be achieved in these two tasks does not require documents to be updated that have any precedence relation. This is specified in the policy violation matrix as $v_{12} = v_{21} = 0$.

• **Synchronize pattern**: Very often, to handle a document/document portion in a $DocWF$ requires other documents/document portions to be handled before. This document precedence makes a potential task to wait for other tasks to be executed successfully. The resulting synchronization can be captured in the feasible condition of a potential task. Consider, in Figure 7, the task $T_3$ needs to wait for the updates of document portions $d_1$ and $d_2$ that are performed in $T_1$ and $T_2$. This can be specified in the feasible condition of $T_3$, $C(T_3) = \{T_1, T_2\}$, means $T_1$ is synchronized with $T_2$ and vice versa for $T_3$.

• **Loop pattern**: This pattern can also prevail in a $DocWF$ execution where some document/document portions need to be handled repeatedly. For example, in Figure 7 (f), new diagnosis records may need to be appended to $d_4$ in task $T_4$ based on which further treatment can be performed. So, tasks $T_4$ and $T_5$ can be performed repeatedly until treatment is not finished.

### 4 Related Work

Although we are not aware of other proposals which can be directly comparable with our approach, many researchers have addressed dynamic workflow aspects in the last decades that vary diversely. Document oriented workflows proposed in [15–19] largely follow the task-based approach where [15] describes a document oriented workflow for a manufacturing process and [16] demonstrate the usage of XML technology to realize a document and workflow based collaborative system. X-folders described in [18], triggers a task from a predefined orchestrated tasks depending on a given state of the documents inside a folder. In our earlier work [12], we developed a secure XML document-based collaboration that allows exchanges of fine grained documents among anonymous actors. Upon receipt of such documents, our approach of a $DocWF$ can be applied to reach a business goal.

While the authors in [2, 20–22] describe various workflow models to support business processes, in [1–8], authors pointed out the necessity of flexible workflow management system as typical workflow models are not made for uncertainty or exception handling.

The authors in [9] proposed a case handling approach to support business processes where each case is handled in isolation (i.e. for each instance). While we
consider the problem area of case handling is close to our problem area of agile business processes, they still consider the tasks and their sequence flow of a case can be specified a priori. Our approach is fundamentally different from this as we allow dynamic task definition and its enactment (i.e. service provisioning). We pointed out the differences in the Table 1 of a DocWF system with traditional workflows and the case-base handling approach.

The author in [23] describes a goal oriented workflow modeling technique to generate alternative workflows whenever necessary. Our proposed DocWF system differs with that approach in two aspects: (1) unlike the goal of [23] which depends on stakeholders goal, our model supports derivation of subgoals including security goals from a business goal independently of actors involved; (2) a goal achievement is recorded by data instantiation in the documents making a document a stateful representation of a workflow which is not considered at all in [23].

The importance of applying formal approaches to the workflow modeling and analysis has been recognized and many formal approaches for task-based workflows have been proposed in [24–27]. Unfortunately, a common major drawback that in all the above formal approaches, only specialized users who have the expertise in these respective formal methods can build their dynamic workflows. In our approach, the formal model utilizes the business notions such as KB, documents, goals, policy and the associated behavior is also depicted using intuitive status and action management models.

5 Conclusion

We proposed a document-based dynamic workflow system that is particularly suitable for agile business processes where required tasks and their sequence flow may need to be decided dynamically. An actor in a DocWF can proactively define tasks to achieve goals. The described formal approach is business intuitive and rule-based that captures various business notions and constraints. A problem with such a rule-based system is possible conflicts, in particular if rules are introduced by different actors. However, associating priorities with rules may resolve such conflicts.

We are currently implementing this DocWF system and investigating various security issues (e.g. document usage control) including conflicts in such a document-based workflow which we believe is equally important in an agile business process.

References


