Document-based Dynamic Workflows: Towards Flexible and Stateful Services

Mohammad Ashiqur Rahaman*, Yves Roudier* and Andreas Schaad†

* EURECOM, 2229 route des Crêtes, 06904 Sophia Antipolis, France, {mohammad.rahaman,yves.roudier}@eurecom.fr
† SAP Research, Vincenz-Priessnitz-Str. 1, 76131, Karlsruhe, Germany, andreas.schaad@sap.com

Abstract—Task-based workflows describe a set of predefined tasks executed in a predefined sequence flow in which documents representing business objects are sent to activate tasks according to some business goal. The increasingly agile nature of business processes implies that neither the potential tasks nor their sequence flow can be defined a priori. In this context, documents can contribute to the central abstraction in a business process execution while services are stateless entities. While business goals and associated business rules drive models and their executions, document content and its structure may additionally be used to determine how the document can be processed and how multiple processing tasks may be composed dynamically.

This paper introduces a document-based workflow model that implements such agile business processes. The described approach relies on the use of a rule-based system as a means to capture diverse concerns such as business goals and associated rules within a uniform framework. To this end, we illustrate this approach with an electronic health record (EHR) application.

Keywords—Workflows, Business goal; Document; Constraint

I. INTRODUCTION

Despite being derived from long term strategic business goals, business processes are increasingly executed in dynamic, uncertain, and data centric distributed environments. This involves diverse aspects of workflows and their application domains including changes in tactical business goals and rules, confronting exceptional or new scenarios etc.

A task-based workflow, being an imperative approach, models a series of all possible sequences of tasks (if known) that are executed at runtime by business actors. Constraint-based declarative workflows [1], [2] argue that such exhaustive task specification makes a model over specified with limited flexibility. Similarly, business rules that capture domain expertise and organizational knowledge may also be over specified. However, in agile business scenarios like treating critical patients, managing disasters, all business tasks and rules may not be known a priori due to their individual peculiarities.

Many flexible workflows are proposed to tackle those scenarios and communities [3], [4], [5] develop techniques for flexible workflow management. Some advocate to avoid changes by providing alternative paths [6], [7] or to control execution paths by declarative constraints expressed in LTL formulas [1], [2]. Others allow changes in one execution and/or changing the process model while migrating all current executions [8], [9], [10].

In business processes actors of different business boundaries primarily deal with business documents or portions thereof from their creation to destruction to achieve goals, all such processing being collectively termed as document handling. A document-based workflow (DocWF) actor models such processes over tactical goals and functional business rules (FBR) independently of any underlying tasks. FBRs capture domain expertise such as goal precedence. Organizational business rules (OBR) over document content and process models further support in finding suitable tasks/services and as well as their dynamic enactment. In nature these rules are declarative that can be expressed using LTL as in [1] and set boundary constraints over goals and document content. When an actor receives a handled document from another actor, the recipient’s OBRs that set constraints on document content are enabled. The role of a DocWF system is to assist an actor in her pro-active task enactment to reach goals rather than to instruct her. As such it focuses on what can be the tasks (i.e. potential task) to achieve a goal rather than what should be the tasks and their precise succession. A comparison of document-based workflows with traditional workflows (e.g. task-based, declarative and case handling [6]) is reported in the Table I.

Recent industry adoption of SOA-based business process tools and frameworks [11], [12] emphasize the reuse of existing capabilities for which we introduce the notion of organizational knowledge-base (KB) that contains existing business process models, their execution history among others. This knowledge is used to determine whether existing tasks/services solve a current business problem. For this, a KB must be searchable using for instance semantic tagging of BPMN with goals and documents with concepts as in [13] and [14] respectively.

We believe that a DocWF system must support (1) dynamic definition of tasks and their sequence flow and (2) shared understanding of document content amongst distributed actors. Document content and KB are utilized in business rule specification for (1) as to either select or define a task for enactment. Also, we model the states (e.g. potentially (not) executable) of these tasks/services at runtime to tackle various runtime exceptions such as service unavailability. A BPEL analogy would be: unlike invoking services in a pre-defined sequence of <invoke> elements a DocWF system finds suitable services or may define services before invoking. For (2) as we showed in [14], a shared
ontology of concepts describing the business documents serves as a stable interface among interacting peers and allows their document schemas to evolve. In the rest of the paper, we show how a business user can model such a DocWF at design time using goals and FBRs and how the model can be executed by business actors operating in distinct business boundaries.

Section II gives an overview of DocWF and followed by a meta-model showing the relationships of different design and run time entities. Section III describes a DocWF execution with an illustrative EHR document generation example. This also includes design and run time error detection techniques. Section IV positions our work with related literature and finally Section V concludes the paper.

II. A DOCUMENT-BASED WORKFLOW (DocWF) MODEL

A. Functional Overview

Fig 1 shows an overview of the envisioned DocWF system. A business expert at design time after possibly browsing its KB, models tactical goals, their precedence and associated FBR as opposed to modeling tasks and their precise succession. At runtime, upon receipt of a document an actor can define OBR associated to its goals. FBR and OBR being merely constraints make it possible not to specify precise tasks and their sequence flow at design time yet enable task enactment at runtime. These constraints specify restrictions on goals, documents and application specific requirements, for instance, credentials associated to a business process. This model may possibly be shared with all the involved actors.

As Fig 1 shows, it is possible to transform the rules into corresponding LTL formulas which are then represented using automata. Based on the automaton of the rules a DocWF model and its execution can be verified by detecting errors such as deadlock and conflicts (cf Section III). The dynamic task enactment component can first find possible tasks/services with the help of a discovery component which performs semantic search in the KB as mentioned previously. In case no such tasks/services are found the actor may define/implement the required services. Then the states of these tasks/services are modeled in the DocWF status component (cf Section III). Based on the states of executed and potential tasks dynamic task enactment is performed which in turn handles associated documents/document portions before sending those to another actor. Note that, the KB can be browsed and/or updated (i.e. consultation) at design or runtime.

B. DocWF Meta Model

An object oriented approach in Fig 2 shows the design and runtime entity relationships of a DocWF meta-model.

Knowledge-base (KB): A KB of an organization is a collection of document semantics (e.g. shared ontology), existing business process models (e.g. BPMN), process execution history (e.g. executed BPEL processes). To enable an actor to consult its KB whenever needed, KB is not associated to any design or run time entities.

Goal and documents: A DocWF system has to achieve a business goal \( G \) from which a set of tactical goals can be derived (i.e. \( G = \{ g_1, g_2, \ldots, g_m \} \), for \( m \geq 1 \)). Such goals are abstract definitions of potential tasks that will be executed by actors.

Such goals may have causal relationships, meaning one goal may not be achieved before another goal. For two goals \( g_i \) and \( g_j \) of \( G \), if \( g_j \) cannot be achieved unless \( g_i \) is achieved then \( g_j \) succeeds \( g_i \), denoted by \( g_j > g_i \).

Let \( D \) be a set of documents/document portions \( d_i \) denoted by \( D = \{ d_1, d_2, \ldots, d_n \} \), for \( n \geq 1 \), that need to be handled. If \( d_j \) cannot be handled before \( d_i \) then \( d_i \) precedes \( d_j \), denoted by \( d_j > d_i \). A simple task/service or a composition of tasks/services can be invoked to achieve a goal that implicitly handles documents/document portions.
The derivation of tactical goals and data instantiation upon a goal achievement are represented by a recursive association of goals that form a goal precedence and another association between goal and document respectively. One goal achievement implicitly enables subsequent goals and documents to be handled until the business goal is achieved. No such precedence between two goals and two documents/document portions are denoted by: $g_j \not\preceq g_i$ and $d_j \not\preceq d_i$ respectively.

Example 1: In Fig 3 (a), the business goal $G$ of the EHR workflow is to generate a complete EHR document $D$ containing patient $<\text{Contact}> d_1$, $<\text{HealthInsurance}> d_2$, $<\text{Symptoms}> d_3$, possible $<\text{DiagnosisTests}> d_4$ and $<\text{Treatment}> d_5$ data maintained by departments (e.g. administration, pathology, diagnosis, operation theater etc). The derived set of goals is, $G = \{g_1, g_2, g_3, g_4\}$; with goal precedence $g_4 > g_3 > g_2$ but $g_{[2-4]} \not\preceq g_1$ (see Fig 1). A diagnosis result can not be recorded without knowing a symptom. Similarly for a successful treatment record a diagnosis result is required. However, patient’s contact and insurance data can be recorded at any time.

**Document semantics:** An actor can define its individual document data model (e.g. schema) and still maintain interoperability by associating its document instances to shared ontology concepts describing relationships of concepts and yet can map those to department’s documents/document portions. For instance, a business concept 'treatment' can be mapped to $<\text{Medicine}>$, $<\text{Therapy}>$ and $<\text{Surgery}>$ related XML fragments of operation theater department (Fig 3 (a)). Such agreed document semantics are a pre-requisite for a DocWF system.

**Potential Task:** As mentioned before, an actor can define potential tasks at runtime. An execution of a potential task handles documents which is reflected either by creating new documents or updating existing documents.

Previously defined business tasks in a BPMN model or invoked services in a BPEL model can be reused as returned by the discovery component of Fig 1. The discovery component can for instance match current goals with the semantically annotated BPMN tasks [13] in the KB. Such tasks/services and their sequence flows are termed as recipe tasks and recipe flow respectively.

However, in exceptional situations where no suitable process models or execution history exist in the KB, new tasks (i.e. free tasks) need to defined and implemented. Their sequence flow (i.e. free flow) can be defined utilizing rules (OBR) over the current goals and document content. Consider a doctor orders diagnostic tests for a surgery patient according to a FBR, but cannot wait for the results as patient condition is critical. She may start treatment by providing medicine or therapy following some OBR. As soon as the test results arrive, she might need to achieve a new goal requiring a completely different treatment (i.e. free tasks) depending on the result (applying another OBR).

The *recipe* and *free* tasks (and recipe/free) are important entities for agile business processes. By finding *recipe tasks* and/or their *recipe flow* of existing business processes of KB allows an actor to reuse existing services. If no such tasks/services are found the free tasks and free flow elements allow her to define tasks and determine their flow at run time by applying business rules.

**Policy:** A policy base of an organization is formed by functional business rules (FBR) and organizational business rules (OBR). At design time by the usage of domain knowledge, FBR describes functional boundaries and relationships of goals. On the other hand, OBRs may be defined at runtime describing organizational concerns such as expected data in documents, their further processing and desired credentials.
of an actor etc. Note that the goal precedence mentioned before can also be a rule.

As mentioned rules are internally represented as LTL constraints. In terms of later deployment of a DocWF system UI templates may be used to abstract such LTL formulas from business actors where each formula is transformed into finite automaton as in [1] (Fig 3 (b)). Transitions to an accepting state of such an automaton satisfies a rule (i.e. true) or false otherwise. This value may change during the execution meaning a rule that is false for the time being may be true eventually. Consider Fig 3 (d), for the FBR = 'Treatment needs to be performed for each diagnosis result' the LTL formula $\mathbf{□}$ (Diagnosis $\rightarrow$ d$_5$) where 'Diagnosis' and d$_5$ refer to the associated goal $g_3$ 'Diagnosis test results' and document portion containing treatment data respectively, i.e. each <Test> record is eventually has a corresponding <Treatment> record. Intuitively, for some diagnosis results corresponding treatment data may not be instantiated at some time but will be filled in d$_5$ after corresponding successful treatment. In Fig 3 (c), an OBR of the pathology department is 'Symptoms can be identified at any time', i.e. $\mathbf{□}$ $\mathbf{◊}$ Symptoms which can be true after a patient’s arrival. Equivalent automata of a FBR and an OBR for the goals $g_3$ and $g_4$ are illustrated in Fig 4 (c) and (d) respectively.

III. DocWF PROCESS EXECUTION

At runtime, an actor can verify the current model and execution so far before proceeding through suitable task enactment.

A. DocWF Verification

For correct execution of a DocWF model, it is important that the model does not contain errors and its current execution is correct. One or more rules are enabled at runtime when a corresponding goal is enabled. Errors are detected using automaton of these rules. The execution of a DocWF is correct at some point of time if all the constraints representing the rules evaluate to true. Similarly, at the end of an execution if all the constraints evaluate to true, the execution has completed successfully. Moreover, for no transition in the automaton of the rules for some goals is considered as a deadlock. Also if the automaton has no states and transitions then there is a conflict. A deadlock or a conflict may occur at both modeling and execution time and can be reported to the actors accordingly.

These are illustrated in Fig 4 (a) and (b) respectively. (a) Consider the FBRs 'Symptom identification must be followed by a diagnosis' and 'Symptom identification must not be preceded by a diagnosis record' of the goals $g_2$ and $g_3$ respectively. If $g_2$ is achieved then $g_3$ can never be achieved as the enabled FBR of $g_3$ states completely opposite of the corresponding FBR of $g_2$ and the system would be in deadlock for a design time error. Similarly, a runtime deadlock may occur when the actor specifies an OBR 'All identified symptoms do not need to be diagnosed' that contradicts with the corresponding FBR of $g_2$. (b) A design time conflict occurs for the OBRs 'At least one diagnosis result must be recorded before proceeding further' and 'Treatment may start at anytime without a diagnosis' of the goals $g_3$ and $g_4$ respectively as both rules in combination constitute no states and transitions in the automaton. A
similar situation arises for the FBR and OBR of the goal \( g_4 \) at runtime.

B. Task Enactment

A DocWF system can find the possible tasks/services by performing a semantic matching of current goals with the corresponding tagging in BPMN models of a KB. Such services may not be reachable or unavailable for various reasons for instance communication or system failure. We model the possible states of a task using simple integer values that capture a task’s re-usability.

Let \( T_i \) be an executed task for a goal \( g_i \), for which the associated rule is \( r_j \) and \( r_j \) be an enabled rule after \( T_i \)’s execution. A task \( T_j \) is a potential task after \( T_i \), denoted by \( T_j \succ T_i \) if either (1) both tasks are chosen from the KB having same precedence, i.e. \( T_j \succ T_i \) or (2) both \( r_i \) and \( r_j \) evaluate to true (if not true yet possibly be true later). Similarly, two potential tasks \( T_i \) and \( T_m \) can be executed in parallel if they satisfy (1) and/or (2), denoted by \( r_{lm} = 0 \) otherwise \( r_{lm} = 1 \).

A directed graph \( G = (T, >) \) represents these precedence relationships, where each potential task is a node in the graph and a directed edge from a node \( T_i \) to node \( T_j \) represents a sequence flow, denoted as \( f_{ij} = 1 \).

**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:

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

By the above definition, only those potential tasks having a value either 1 or 3 can be executed (see Fig 5). Based on the task state value we define a DocWF status that represents current execution status of a DocWF system.

**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 an execution of a DocWF process. An execution of a document-based workflow is a tuple \( \text{DocWFexec} = (G, D, T, F, R, S_1, S_F) \), 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.
4. \( F = [f_{ij}]_{|T| \times |T|} \) is a sequence flow matrix of the tasks of \( T \). i.e. \( f_{ij} = 1 \) if \( T_j \succ T_i \) otherwise \( f_{ij} = 0 \) for \( i = 1, 2, ..., |T| \) and \( j = 1, 2, ..., |T| \).
5) \( R = [r_{ij}]_{T \times [T]} \) is a rule violation matrix of the tasks of \( T \), i.e. \( r_{ij} \in \{0,1\} \) for \( i, j = 1, 2, \ldots |T| \).

6) \( S_I \in \{0,1,2,3\}^{|T|} \) is the initial status of the DocWF.

7) \( S_F \in \{2,3\}^{|T|} \) is the final status of the DocWF.

In order to provide a comprehensive picture of an execution to an actor, the DocWF status component can compute the DocWFexec tuple. As an execution goes on \( T \) is increased and so are the matrices \( F \) and \( R \) and all together show the states of executed tasks, their sequence flow and rules applied. Given initial status \( S_I \), for a task \( T_j \in T \) if there is no \( T_i \) such that \( T_j > T_i \), i.e. \( f_{ij} = 0 \) then \( S_I(T_j) = 1 \); otherwise \( S_I(T_j) = 0 \). Note that, initial status may consist of state values of executed tasks (i.e. \( \{2,3\} \)). As such a DocWF execution may start from an unfinished workflow execution as opposed to a task-based workflow which is typically in a blocking situation in case of an exception. On the other hand, in the final status \( S_F \), all the documents/document potions are handled and all the goals and thus the business goal is achieved (implies a potential task is executed at least once as denoted by \( \{2,3\} \)).

**DocWF Status Transition Rules:** The dynamic execution of the tasks is governed by a set of transition rules based on which the incremental set of executed tasks \( T \) is built. We define general transition rules that take uncertainty and re-usability into account during a DocWF execution. For example, in the EHR generation workflow (Fig 6 (f)) it is possible to perform additional symptoms tests, i.e. \( T_6 \) while doctors are doing medication treatment, i.e. \( T_5 \) and can decide dynamically depending on the diagnosis record whether they proceed with therapy or surgery (explained in the latter section). Details of these transition rules can be found in [16].

**Knowledge-base (KB) update:** A successful execution trace (i.e. sequence flow, executed BPEL), handled document's semantics and associated rules can be added to the KB by the actors and thus the KB gets continuously enriched for further consultation.

**C. A DocWF Task Enactment Example**

Assuming no occurrence of modeling and execution error, we now illustrate a dynamic task enactment of the fictitious EHR generation workflow by applying dynamic task enactment approach of Section III B (see Fig 6). The actors are hospital administrative employees, pathologists, diagnosis technicians and doctors. Some tasks require human interaction and other can be invoked as automated services. For each transition the DocWFexec elements (i.e. executed tasks, handled documents \( D \), sequence flow \( F \) and rule violation matrices \( R \) are instantiated. Fig 6 illustrates an execution and the complete instantiation can be found in [16].

Assuming a patient arrival service, i.e. task \( T_0 \), triggers the business goal 'Generating a complete EHR' (Fig 6(a)). The initial goal \( g_1 \), i.e. 'up-to-date patient particulars' with its FBR is 'Separate patient’s contact and health insurance data'. For this an OBR refers to the document portion \( d_1 \) and \( d_2 \) of the administration department and allows parallel updates on \( d_1 \) and \( d_2 \). Also, a human oriented task i.e. 'Contact update service' (\( T_1 \)) and an automated service, i.e. 'Health insurance service' (\( T_2 \)) are found in the KB (i.e. recipe tasks) for the goal \( g_1 \). As the administration employee is available the task \( T_1 \) is immediately executable (state 1). However, the 'Health insurance service' is currently unavailable (indicated by state 2) but can be executable as soon as it is available (Fig 6 (b)). The tasks \( T_1 \) and \( T_2 \) might not need to be performed if the patient was previously admitted in the hospital and as such patient’s particulars would have had up-to-date already. This can be determined for instance from a BPEL execution history of the generating EHR process in the KB.

As the goal \( g_2 \) 'Symptoms identification' can be achieved independently of the goal \( g_1 \), a human oriented 'Pathology service', i.e. pathologists can already record the problem symptoms of the patient, (i.e. task \( T_3 \)) into \( d_3 \) (Fig 6 (c)) as suggested from the KB. Note that the state of the executed task \( T_1 \) is 3 meaning patient’s contact can be updated whenever required.

Now, for the goal \( g_3 \) 'Diagnosis tests results’ an associated
symptom must be identified in \( d_3 \) according to its FBR (of Fig 3 (c)). So, after \( d_3 \) is filled in with a symptom data by the task \( T_3 \), a diagnosis technician can record the test result by another human oriented service 'Diagnosis service', i.e. \( T_4 \) (Fig 6 (d)). Note that the states of task \( T_3 \) and \( T_2 \) are now 3 meaning symptom identification can occur again and 'Health insurance service' is executable now respectively.

As soon as the diagnosis test results are recorded in \( d_4 \), the task \( T_4 \)'s state is changed to 3 that allows more diagnosis records whenever required (Fig 6 (e)). Note that the state of the task \( T_2 \) is changed to 2 indicating that service is again unavailable. Now for the goal \( g_1 \) 'Successful treatment', operation theater has OBRs enabling suitable treatment (e.g. medication, therapy, surgery etc.). The doctor may start treatment, i.e. \( T_5 \), by suitable 'Medication'. As the rules of the pathology, diagnosis and the operation theater allow further tests and ongoing according treatment, at a later time the doctor may need some other pathology diagnosis test records before advancing further in the treatment (i.e. \( T_6 \)). It indicates that the \( d_4 \) may need to be handled again while the doctor is performing the treatment i.e. \( T_5 \). Now depending on the current test results the doctor may need provide 'Surgery' and/or 'Therapy' services for a successful treatment (Fig 6 (f) (g)).

Now, when the treatment is successfully completed, the DocWF execution will reach a final status by generating a complete EHR and thus the business goal is achieved (Fig 6 (g)).

### IV. Related Work

Although we are not aware of other proposals which are directly comparable with our approach, many researchers have addressed dynamic and flexible workflow aspects in the last decades that vary diversely. Document oriented workflows proposed in [17], [18], [19], [20], [21] largely follow the task-based approach where [17] focuses on a manufacturing process and [18] demonstrates the usage of XML technology to realize a document and workflow based collaborative system. In line with [18], we developed a secure XML document-based collaboration technique [22], 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. X-folders described in [20], trigger a task from a predefined orchestrated tasks depending on a given state of the documents inside a folder.

The authors in [6] proposed a case handling approach to support business processes where each case is handled in isolation (i.e. for each instance). In [1], [2], constraint-based declarative workflows are introduced. Apparently the problem area of case handling and declarative workflows are close to DocWF approach for agile business processes. However, case handling still considers the tasks and their sequence flow of a case are specified a priori and declarative approach applies constraints on a set of pre-defined tasks for flexible execution. Our approach is fundamentally different from these as we allow dynamic task definition, its enactment and our rules set constraints on goals and document content. Moreover, they do not consider any semantic approach to enable re-usability of tasks/services. We may also allow constraints on tasks as organizational rules, for instance for a 'successful treatment' a doctor advises a specific medication service due to its previous successful service delivery. Table I points out the main differences of a DocWF system with other workflow approaches.

The author in [23] describes a goal oriented workflow modeling technique to generate alternative workflows whenever necessary. Our proposed DocWF system differs from 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].

### V. 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. A business actor can model a DocWF process utilizing her domain expertise without thinking about precise tasks/services which will be enacted by an actor pro actively at runtime possibly by defining new tasks. The employed business rules set constraints over goals and documents and are expressed as LTL formulas. Such rules treat a DocWF execution as goal achievements while a goal can be grounded to one or more task enactments. Utilizing the automaton of these formulas, actors may report modeling and execution errors at design and run time. A problem with such a rule-based system is possible contradiction, in particular when rules are introduced by different actors. However, associating priorities with rules may resolve such contradiction.
REFERENCES


