Goal-Compliance Framework for Self-Adaptive Workflows

Budoor Allehyani, Stephan Reiff-Marganiec

Department of Informatics
University of Leicester
Leicester, UK
Email: {baaa2,srm13}@le.ac.uk

***Abstract*—Workflow adaptation involves two major research topics: flexibility and correctness. The former is related to the ability to react to change and adapt workflow structure, while the latter is related to managing this flexibility and ensuring syntactical, semantical as well as behavioural consistencies. Current approaches range from providing flexible workflows to flexible and consistent workflows. They mostly focus on syntactical consistency and generic properties (such as deadlock-freedom), but rarely consider semantic aspects. However, not providing semantic guarantees neglects the importance of preserving the original goal. The primary focus of this research is to ensure goal compliance during workflow reconfiguration. Thus, we analyse the impact of workflow automatic adaptation on the goal in question. As a result, we define goal-compliance constraints and develop a goal-compliance framework, which automatically and dynamically adapts workflow instances through Event-Condition-Action policies. Furthermore, it validates the adaptation against the goal-compliance rules and constraints through model checking and ontology-based approach.**

Keywords–BPMN; Reconfiguration; Goal-Compliance; Model Checking; Ontology; Runtime Verification.

I. INTRODUCTION

The Business Process Model and Notation (BPMN) [1] is an efficient language for modelling business processes. However, it is insufficient for analysis and verification purposes. This is due to the fact that BPMN lacks in techniques and tools that support process analysis. However, there exist some successful approaches that map BPMN semantics to several formal languages, such as Petri nets [2] and Communicating Sequential Processes (CSP) [3], which are formal and tool supported. As the business domain is well-known for its dynamicity and complexity, processes should be self-adaptable and self-manageable. Gorton [4] provides self-adaptive workflows (WF) using (Event-Condition-Action(s)) policies to change WF specifications at runtime and on an instance level. We build upon his work aiming at self-management workflows, whic correctly and safely react to change. We aim to make WF systems as flexible as possible without sacrificing their functionality. The main focus of this research is on guaranteeing goal compliance in self-adaptive workflows. The goal model is considered as the main reference for the WF functionality in its entire lifetime from design to development. Therefore, any changes or updates applied to a WF must satisfy the original goal. The novel contributions we present in this paper are: (1) the goal-compliance framework for runtime reconfiguration and verification and (2) the mechanisms to preserving business goal. This research is basically motivated by the following research questions: 1) How can we write specifications that are precise enough to exclude bad implementations (undesired behavior) while at the same time being flexible enough to cope with the kind of changes we wish to allow? 2) How can we detect consistency with a high level specification?

The remainder of this paper is structured as follows: a brief background about the main concepts used throughout this paper is in Section II. Section III provides an overview about the goal-compliance framework, Section IV gives more details about the development of the verification mechanisms used for goal-compliance check. We include an initial evaluation for our framework in Section V. Section VI discusses some related approaches and we conclude the paper in section VII.

II. BACKGROUND

We briefly introduce the main concepts used in this paper: BPMN, goal specification, domain knowledge. The BPMN process model can be defined through a BPMN diagram, which illustrates what activities are to be executed and in what order. Thus, business process functionality is captured by the BPMN process model. An activity is defined by the BPMN specification as a generic term for work a company performs within its business process. It can be atomic or composite and it is of three types: task, subprocess and call activity.
A goal can be defined as ”high-level objectives of the business, organization or system; they capture the reasons why a system is needed and guide decisions at various levels within the enterprise” [5].

In Requirements Engineering, there are different techniques and methods used to ”formally” model and declare goals. One of the methods is requirement specification, which relates business goals to functional system components. Keep All Objectives Satisfied (KAOS) [6] is a goal modelling method aimed at requirement elicitating and validating. It encompasses five major concepts: goals, assumptions, agents, objects, and operations. In this research, we only consider the goal concept of KAOS and relate it to the BPMN. In KAOS, a goal model consists of the strategic goal and its refinement objectives. The refinement relation is of two types: 1) AND refinement where all related objectives must be achieved and 2) OR refinement where at least one of the related objectives is achieved.

Domain knowledge is derived according to the goal in question. Ontologies are a common technique for knowledge representation. An ontology is defined as ”a formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on

 ISBN: 978-1-61208-532-6

slots (facets (sometimes called role restrictions))" [7].

Reconfiguration policies, which are introduced in [4], are used to adapt running BPMN instances and their syntax is defined as follows:

polrule ::= appliesto location [when triggers] [if conditions] do actions
triggers ::= trigger | triggers or triggers
conditions ::= condition | not conditions | conditions or conditions | conditions and conditions
actions ::= action | actions actionop actions actionop ::= and | or | andthen | orelse

III. GOAL-COMPLIANCE FRAMEWORK

The presented Goal-compliance framework supports on-the-fly WF adaptation while preserving the WF semantics. Generally speaking, the notable aspects of the framework are:

- Online Workflow Reconfiguration at Instance Level: The framework provides flexibility for WF systems by inserting, deleting and replacing workflow tasks. This flexibility is provided by three important factors for runtime adaptation: change per instance, online adaptation (i.e., change on running instances), automatic adaptation using ECA policies and change management.
- Goal-Compliance Validation Capabilities: A Goal-Compliance check is the key feature of this framework. Before applying any workflow change, the framework has the ability to check the corresponding constraints and decide whether to accept the change or not. Each change variability has its corresponding constraints based on the analysis of its affect on goal satisfaction. Therefore, a goal-task dependency check is related to deleting a task from the running process, while the task-domain conformance check is related to inserting a task to the running process.
- Facilitating Other Semantic Checking: Using the ontology within the framework could also facilitate other types of semantic checks by enhancing/reusing the ontology to add more constraints or define different rules. Furthermore, it could be used for querying the ontology while performing such a semantic verification.

A. Architecture

The runtime framework assumes an adapted process execution engine. For simplicity, we assume here that we have an engine that can execute BPMN processes directly (this allows us to focus on the main aspects rather than worrying about converting these into some executable formats). The engine is able to pause a process instance and also to make changes to instances. Fig. 1 presents the block diagram of the proposed framework.

As the process instance executes it will raise triggers e.g., at the start of a task which are passed to the policy server (a policy enforcement point), which either returns a no change allowing the instance to be processed as it is or a specific change action, e.g., the need to insert a task, which will lead to updating the process structure of the instance. The action that the policy server demands depend on the policies in the repositories and of course the instance data in the process. The policy server retrieves policies from the policy store, checks for the applicability and then considers the actions to be applied. Once it has determined what actions should be applied, the process instance is updated accordingly and would continue executing in its new shape. Through the work presented here an extra phase is added, namely that of checking that the change is appropriate in the sense that it maintains the goal semantics of the original process.

As can be seen from Fig. 1, the proposed framework accepts the original WF specification, the modification details and the domain compliance constraints as inputs. The WF specification is in BPMN file and is in xml format and it is automatically transformed to CSP. The modification details can be in a configuration file that determines the changes to the WF specification through ECA policies.The framework consists of three components namely, Specification Reader, Reconfigurator and Validator. The brief description of each one is provided below.

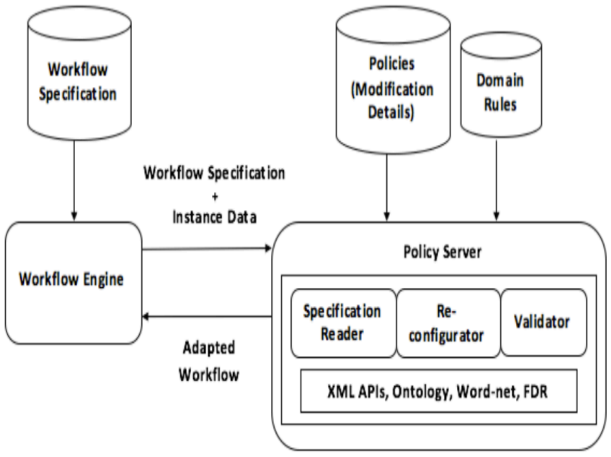

Figure 1. Goal-Compliance Framework Architecture

- Specification Reader: This component is responsible to read the existing WF specification and transform it into an in memory state for fast processing and easy manipulation of the modification. This can be achieved by utilizing some XML interfacing APIs (Application Programming Interface).
- Re-configurator: The re-configurator is responsible to process the actual modification operations e.g., insertion of the new task into existing WF specification. This component is responsible to interact with the Ontology and WordNet tools to carry out the requested modification on the existing WF specification.
- Validator: The validator is responsible to ensure that the modification is according to the given specification and it does not violate any domain compliance rules or constraints. The domain compliance rules are the constraints that help to ensure the preservation of the original goals of the WF. This can be achieved by exploiting the Failure-Divergences Refinement (FDR) and other necessary validation tools.

 ISBN: 978-1-61208-532-6

B. Implementation

The proposed framework is implemented with Java and works in the following sequential order:

1) Read the existing WF specification
2) Read the reconfiguration details
3) Validate the reconfiguration against goal-domain compliance constraints
4) Adapt the WF, if the validation from (3) was successful
5) Produce adapted WF specification for the running instance

The reconfiguration component of the framework supports three processes including insertion, deletion and replacement. The details of each process are included in the following sub sections.

1) Insert: The insert process refers to the facility where the proposed framework allows the modification of existing workflow by allowing the insertion of a new task into the given WF specification. The new inserted tasks can be of any of the following kind of tasks:

- Atomic sequential: This refers to the insertion of a new task in sequential order immediately after a given task. This operation requires the new task name as well as the name of an existing task.
- Atomic parallel: This refers to the insertion of the new task in parallel to an existing task. The operation will insert the parallel gateway to connect the new task and existing task in parallel. The new and existing task names must be provided to perform the operation.
- Composite: The composite task is itself a collection of multiple tasks. The framework allows the insertion of a new composite task. In this operation, the framework will receive multiple task names, which collectively represents the composite task. The framework will then insert those tasks as a composite task in reference to an existing task.

The procedure developed for the insertion of the new task to an existing workflow is the same irrespective of the above mentioned types. The domain-conformance constraint is implemented for insertion verification with ontology support. The short explanation is provided below.

a) In the first step, the framework reads the existing WF specification and then obtains the new task name from the configuration file that contains the modification details. This name is then searched from the available ontology. This search query targets that the task name must match an individual name in the ontology satisfying the constraint that the individual must belong to the same domain as the domain of the WF specification. If the search succeeds, then the Re-configurator will allow the insertion of the new task. Otherwise, it will carry out Step-b. b) In case the given task name is not available in the ontology, then the framework will attempt to explore the possibility to confirm the suitability of the task name through WordNet. The framework assumes that the task name must consist of two words separated by a special character (e.g., _). The first word represents action, while the second word represents object (e.g., Register_Student). The framework interacts with WordNet repository to obtain the synonyms of both words (i.e., action and object). The object part and their synonyms help to identify the corresponding domain of the workflow, whereas the action part hints at the type of the action. For example, "Register" indicates that the task should be of type "Registration" and the "Students" indicates the BPMN domain "UniversityAdmission".
c) Once the synonyms are retrieved, then they are searched in the ontology. The framework will allow the insertion of new task, if any of the synonyms of both parts are found in the ontology. Otherwise, the framework will not allow the insertion of new task.

2) Delete: The delete process of the proposed framework refers to the facility of modifying a given WF specification through allowing the deletion of an existing task. Similarly, to the insert process, the framework allows the deletion of Sequential atomic, Parallel atomic and composite tasks. The goal-task constraint is implemented here with FDR support. The brief description of the main steps is provided below.

a) The framework reads the WF specification file and the configuration file that contains information on the task that is to be deleted. The framework first ensures that the task to be deleted exists in the specification. b) The framework then ensures that the deletion operation does not violate any of the domain compliance rules or any other constraints. If not, then the requested task is deleted from in-memory representation of the WF specification. A modified WF specification must be produced at the end of the process. c) If the deletion of the task violates any of the domain compliance rules or other constraints, then the framework will not allow the deletion of the task.

3) Replace: The replace process of the proposed framework allows the replacement of an existing task with a new one. Within the ontology, all semantically equivalent tasks are defined using the ontology semantical relation "SameIndividualAs" to indicate they hold the same semantic. Therefore, replacing one with another does not affect the process semantic. The brief description is provided below.

a) The framework reads the WF specification file and the configuration file that contains information of the existing task and the new tasks that will be needed to replace. The framework first ensures that the existing task that has to be replaced exists in the specification. b) The framework then searches the ontology to identify whether both of the tasks are the same individuals or not (i.e., they must be semantically equal). If both tasks are the same individual in the ontology, then the framework will allow the replacement.

However, there could be different ways to define semantical relations among task individuals, which might be used to define other constraints for the replace policy.

IV. Goal-Compliance Assurances for Runtime Verification

We individually analyse the impact of the reconfiguration policies on goal satisfaction based on the action indicated within the policy; insert, delete or replace. Therefore, two types of constraints are defined: goal-task dependency and domain-task conformance. The former is defined as a result of deleting BPMN tasks while the latter for inserting or replacing tasks. Model checking is used to validate the goal-task dependency constraint and this is due to its applicability for this type of validation, where a property capturing a certain behaviour must

 ISBN: 978-1-61208-532-6

satisfy a model specification. However, inserting new tasks to the BPMN differs from removing existing ones. Therefore, it implies different type of constraints using an ontology-based approach as it encompasses everything about the domain and facilitates this type of verification. If the new task is consistent with the domain, it should satisfy the goal.

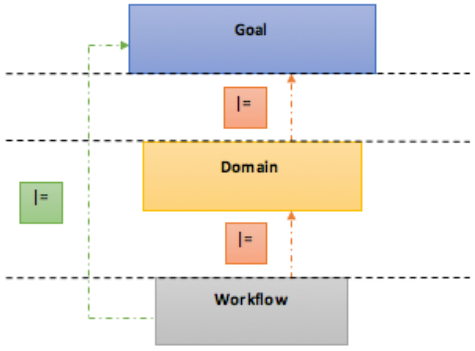

Figure 2. Conformance relationship among workflow, its domain and goal

There exists a satisfaction/conformance relationship between the running process (workflow), the goal in question and its domain, depicted in Fig. 2. Goal specification is located in the top layer since it is considered the main reference for workflow designing as well as development. The middle layer is domain knowledge, which represents concepts of a specified domain and their relationship/semantical relations. The WF specification is localised at the bottom layer. The original as well as the adapted WF specification must satisfy the domain and the goal in question. If the WF satisfies the domain rules, this will lead to goal satisfaction. In the following subsections, we are going to discuss the goal-compliance constraints and their implementation in details.

1) Goal-Task dependency: We define goal-compliance properties based on original goal specification in order to keep it consistent during workflow reconfiguration. As goal and process models are dependent, we establish a link of satisfaction based on the dependency between goals in goal model and tasks in process model called goal-task dependency link. Note that the establishment of this task was inspired from [8] but we consider goal satisfaction at a high level of abstraction. KAOS is used to model the goal formally allowing for specification in LTL (Linear Temporal Logic) with variant patterns [8]: (1) Achieve goals, (2) Cease goals, (3) Maintain goals, (4) Avoid goals. The first and third patterns help to verify the availability of certain desired behaviour.

The establishment of the goal-task dependency link allows us to indicate property specifications, which in turn guarantee goal achievement. Hence, the constraint formulae are written as WF $\models$ P, where P is property specification. CSP is candidate as a process and property specification language. The process model we have is expressed as BPMN diagram and this BPMN is transformed to CSP using Wong's tool [3]. Goal specification is expressed in LTL patterns and they are converted to CSP specifications using Wong's property specification patterns [9]. The above constraint formulae then can be automatically checked using the FDR tool [10] through refinement assertions.

The following represents the steps we follow in order to implement the verification of goal-task dependency constraint:

1) Define the goal for a given domain
2) Identify goal-related tasks based on goal-task dependency link
3) Define property specifications using the result from (1). Property specifications should state the availability of all goal-related tasks and must be consistent with goal specification
4) Convert property specifications from (3) into CSP specifications
5) Check the refinement relation (satisfaction function "P [=R WF"), which indicates that the process specification satisfies the property under Refusal refinement (R).

There are three types of goal-task relationships:

1) One task is contributing to achieve a single objective
2) Groups of tasks are contributing to achieve a single objective and this could be:
   a) OR-grouped tasks
   b) AND-grouped tasks

Those variants are classified according to the refinement relation among their corresponding objectives in KAOS goal specification. CSP refinement notion together with the hiding operator make it possible to model check self-adaptive workflows in a sufficient way. In particular, it facilitates to check the availability of certain events (tasks). So, in property specification we identify the functional behaviour that is related to a goal specification. Then, this property specification is tested through refinement assertion with hiding particular events. Based on the type of the property, the hiding is provided. For properties that are of type (1), we need to hide all process alphabets from WF specification in the right hand side excluding the event that the property holds in the left hand side. This allows the model checker to check for a certain behaviour. In case of properties of type (2), when property specification states at least one of the events is available, then the removed tasks by policy should be hidden from WF specification.

For example, suppose a BPMN process consists of sequenced tasks A, B and C. A OR B are contributing to achieve an objective O_1. C is contributing to achieve O_2. The CSP property specification that captures the availability of A OR B is defined as follows:

P= let Spec0= A $\rightarrow$ Spec2 Spec1= B $\rightarrow$ Spec2 Spec2= C $\rightarrow$ *SKIP* within Spec0 $\sqcap$ Spec1

Now, suppose a policy wants to delete task A from process specification. The framework is going to verify this by checking the refinement relation between P and WF as follows:

assert P $\sqsubseteq_R$ WF $\setminus$ (A) where $\setminus$ indicates "hide" and it means hide A from WF specification because it is the targeted task by the policy. In this case, the refinement relation holds because B is still running in the process. If A and B are going to be removed, the assertion will fail.

2) Domain-Task Conformance: All desirable actions or functionalities that any organization wishes to achieve are determined basically through goal specification. Those functionalities in predefined order are captured by WF systems. The insert function is used to add extra functionality to the workflow. It can insert a new workflow item (activity or operator) at any position. As a result, it might have a significant impact on achieving the original goal if left uncontrolled. We

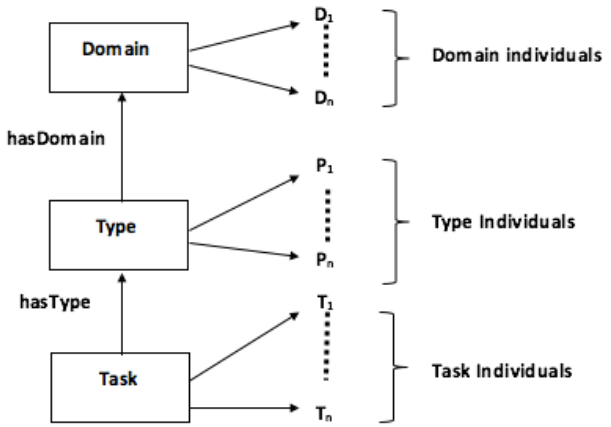

Figure 3. Ontology Structure

focus on the semantical impact in which the functionality being added deviates from the goal or is inconsistent with the knowledge in a given domain. Semantical impact is conceived as undesirable actions that might causes unexpected outcome, which in turn might affect business outcome. Hence, the satisfaction between a workflow and its domain will lead to goal satisfaction. As a result, we develop an ontology for verifying the consistency of adapted BPMNs.

The ontology is consistent with the goal in question. We assume it encompasses everything about the BPMN domain in terms of BPMN tasks, their classification and relationships. The classification allows to group tasks based on their semantic in order to be able to verify consistency. For example, for verification purposes, we classify BPMN tasks according to the work they are designed for, (e.g., the tasks "Notify_ Cancellation" and "Notify_ Timeout" are classified as Notification tasks). However, they could be classified according to different criteria for other types of checking.

Although BPMNs are domain specific, i.e., domains differ in their goals and the purpose they are designed for, we develop a generalized semantic constraint. For example, Flight-Booking is a different domain than Pizza-Delivery as the concepts used within the processes as well as their outcomes are different. The domain is captured in an ontology following the structure Domain-Type-Task. It is defined in the Web Ontology Language (OWL) using Protégé [11].

The ontology combines three classes: Domain class includes different domains, Type class includes type classifications of domain tasks and Task class encompasses all BPMN tasks related to specific domains. Individuals of the three classes are linked using OWL object properties. Basically, we have two object properties: hasType linking tasks with types and hasDomain linking types with domains, see Fig. 3. In this work, BPMN tasks are considered to be the domain concepts as they are the main artifacts in process execution since they are designed to perform work within the process. We use domain knowledge to reason about goal satisfaction. This is due to the fact that goal specification holds the desirable actions but is abstracted from any detail about the process it is designed for. For this reason, we use the domain knowledge to prove consistency with the goal as it holds more details about the executed process and adheres to the goal.

V. Related Work

The correctness of self- adaptive workflow systems has been an active research area in recent years. Correctness is a broad concept and it varies according to adaptation level. Adaptation could be at process, infrastructure, domain or resource level [12]. Generally speaking, process correctness can be divided into three major criteria; syntactic, semantic and behaviour. Each of these can further be divided into several criteria, for example syntactic correctness covers properties like reachability and inheritance. Current approaches focus on syntactic and behaviour correctness. However, semantic assurances, such as data flow correctness, task compatibility, rule compliance are also important aspects to ensure safe adaptation. In the literature, three semantic constraints are defined for workflow validation.

(1) Task-task dependency [13], which is developed to ensure compatibility among tasks in terms of order correctness among running tasks.

(2) Mutual exclusion and Coexistence constraints [14], which express the incompatibility between two tasks to avoid running them together and vice versa.

They are implemented over semantic conformance-oriented ontology for verifying workflow correctness at design time. [15] developed dependency models in order to manage process model variants not instance variants. Satisfying goal is another semantic criteria that must be addressed for self-adaptive systems. Koliadis and Ghose [8] developed GoalBPMN for studying and analysing the effect of changing goal specification in respect with its BPMN. BDI agent technology was used to develop agile goal-oriented business processes [16]. This approach handeled both modeling as well as adapting processes but they assume changing at goal level and restructure the process model accordingly.

In this work, we provide assuarances on goal-compliance (adapted process model is compliance to its original requirements) considering instance variants for running workflows at a high-level of abstraction.

VI. Conclusion and Outlook

A. Conclusion

In this paper, we presented a goal-compliance framework, the motivating approach behind it and its implementation. Basically, our approach focuses on providing assurances that the goal of self-adaptive workflows is still satisfied. As a result, we introduced two major compliance constraints: goal-task dependency and domain-task conformance constraints. The goal satisfaction is considered at a very high-level of abstraction neglecting the implementation details following the fact that workflows are designed to capture business goal. This allows to prevent errors and inconsistency at the abstract level, which in turn will reduce the effort, error and cost at data level.

B. Outlook

The Goal-Compliance framework performs runtime verification in a feasible as well as straightforward fashion. We run ana evaluation process based on the following criteria: 1) framework performance, 2) framework adequacy and 3) ontology accuracy. The performance is to measure time taken to read a BPMN, change its structure as required by policies and verify its compliance to the constraints. The main objective

 ISBN: 978-1-61208-532-6

to measure the time is because the framework supposed to do its work at runtime and ensuring that its performance is reliable in practice. This point is planned as a future work and JProfiler [17] was chosen for this purpose.

However, we measure the ime taken by FDR to perform the verification related to the delete policy. FDR showed that the average time to calculate a simple assertion (e.g., the availability of the task "Confirm_Booking" to achieve the objective "FlightBooked" in the Travel domain) is 0.81s. Note that some objectives are achieved by the contribution of more than one task and the property is defined based on the refinement relations between their corresponding objectives in goal specification. For example, the tasks "Quote_Flight" OR "Quote_Hotel" OR "Quote_Car" are contributing to achieve the objective "TravelPlanGenerated". For these types of properties, the average time taken is 0.2s.

The framework adequacy is concerned with the workflow patterns [18] as they are widely accepted and capture most of the WF behaviours. Case by case analysis shows that 33 out of 43 of those patterns are supported within our framework. The unsupported patterns are those that are not implemented by BPMN.

The proposed ontology can be generalised to represent any BPMN domain. It is based on an assumption that it encompasses all tasks (designed and un-designed) that belong to a specific domain. However, predicting all tasks related to instance variants is impossible at modelling time. As a result, WordNet was integrated within the framework for synonyms search. We analysed the proposed ontology taking its accuracy into consideration. The accuracy is classified as a correctness metric and it includes precision, recall and coverage as the main measures [19]. We conducted a number of experiments on different BPMN(s) from different domains. In general, the number of verified tasks, which matched with Task individuals in the ontology, was 33 out of 39. Six tasks were not found in the ontology directly, but 4 were matched through synonyms finding with WordNet, making a total of 37 matches. However, two tasks failed to meet the domain-conformance constraints and as a result were rejected.

Based on theses results, the precision of the D-T-T ontology is 94.8% and the recall is 100%. The results show that this is a very promising approach, as long as the structure of the task name is 'well formed' in a verb-noun form (action followed by object: Send_Mail or Place_Order). The approach will extend to more complex task names, but more parsing and intelligence in the matching with the ontology is required.

REFERENCES

[1] "Object management group business process model and notation," URL: http://www.bpmn.org [accessed: 2016-10-25].

[2] P. C.A., Kommunikation mit Automaten. PhD thesis, Institut fur instrumentelle Mathematik, 1962.

[3] P. Wong, Formalisations and Applications of Business Process Modelling Notation. PhD thesis, University of Oxford, 2011.

[4] S. Gorton, Policy-driven Reconfiguration of Service-targeted Business Processes. PhD thesis, University of Leicester, 2011.

[5] A. Antón, "Goal-based requirements analysis," in Requirements Engineering, 1996., Proceedings of the Second International Conference on. IEEE, 1996, pp. 136–144.

[6] A. Lapouchnian, "Goal-oriented requirements engineering: An overview of the current research," University of Toronto, 2005, p. 32.

[7] N. Noy and D. McGuinness, "Ontology development 101: A guide to creating your first ontology," URL: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html [accessed: 2016-09-14].

[8] G. Koliadis and A. Ghose, "Relating business process models to goal-oriented requirements models in kaos," in Advances in Knowledge Acquisition and Management. Springer, 2006, pp. 25–39.

[9] P. Wong and J. Gibbons, "Property specifications for workflow modelling," in Integrated Formal Methods. Springer, 2009, pp. 56–71.

[10] "Fdr3 released, oxford university computing laboratory," URL: http://www.cs.ox.ac.uk/projects/concurrency-tools/ [accessed: 2016-10-12].

[11] "Protege," URL: http://protege.stanford.edu [accessed: 2016-08-22].

[12] Y. Han, A. Sheth, and C. Bussler, "A taxonomy of adaptive workflow management," in Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work, 1998.

[13] L. T. Ly, S. Rinderle, and P. Dadam, "Semantic correctness in adaptive process management systems," in International Conference on Business Process Management. Springer, 2006, pp. 193–208.

[14] T.-H.-H. N. Tuan Anh Pham and N. L. Thanh, "Ontology-based workflow validation," in Computing Communication Technologies - Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on, Jan 2015, pp. 41–46.

[15] C. Sell, M. Winkler, T. Springer, and A. Schill, "Two dependency modeling approaches for business process adaptation," in International Conference on Knowledge Science, Engineering and Management. Springer, 2009, pp. 418–429.

[16] B. Burmeister, M. Arnold, F. Copaciu, and G. Rimassa, "Bdi-agents for agile goal-oriented business processes," in Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 37–44.

[17] "ej technologies," URL: https://www.ej-technologies.com/products/jprofiler/overview.html [accessed: 2016-09-20].

[18] "Workflow patterns," URL: http://www.workflowpatterns.com [accessed: 2016-09-22].

[19] H. Hlomani and D. Stacey, "Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey," Semantic Web Journal, 2014, pp. 1–5.

 ISBN: 978-1-61208-532-6