Service-Oriented Framework for Human Task Support and Automation

Ana Sasa, *Member, IEEE*, Matjaž B. Juric, and Marjan Krisper, *Member, IEEE*

***Abstract*—Due to increasingly demanding requirements for business flexibility and agility, automation of end-to-end industrial processes has become an important topic. Business process execution needs to support automated tasks execution as well as human tasks. In this paper we show that for certain types of human tasks it is relevant to consider their further automation. We propose a service-oriented architectural framework for human task execution, which improves their execution by automating and semi-automating decision making based on ontologies and agent technology. The approach is generic and can be used for any type of industrial or industrial support business process. As a proof-of-concept we have developed a system providing the above-described support for human task intensive business processes in an electric power transmission company, which has shown considerable improvements in the efficiency of human tasks.**

***Index Terms*—Business process automation, multi-agent system, ontology, service-oriented architecture.**

I. Introduction

DUE to increasing requirements for business flexibility and agility, automation of end-to-end industrial processes has become an important topic. Nowadays one of the most promising and widely acknowledged approaches for business process automation is based on the principles of service-oriented architecture (SOA). In SOA a business process is composed of services, which represent different tasks that have to be performed in a business system [3]. Tasks can be implemented by automated services (automated tasks) or delegated to users (human tasks). In this paper we focus on human tasks. Human tasks can represent different forms of work, from installing a new device to making a decision about hiring a new employee, for example. In order to integrate human tasks into an SOA-compliant business process execution system, a human participant responsible for task execution (task owner) has to be informed about when to perform a task, what is required to be done and, after its completion, the task result needs to be passed back to the process. We have identified four levels of support for human tasks by such systems:

- Level 1—Human tasks: For every human task, its input is provided to the task owner. The task owner performs the task and enters the task result, which is passed back to the process.
- Level 2—Human tasks with user support: When possible, the system is capable of providing the task owner response suggestions, propose decision models, agenda, etc. together with the task input. Based on this support the task owner performs the task and enters the task result, which is passed back to the process.
- Level 3—Semi-automated human tasks: The difference between this level and level two is that the system is capable of performing certain tasks on behalf of task owners, such as coming to certain conclusions, decision making tasks etc. These tasks become automated human tasks. Even if the system performs a task on behalf of its task owner, their confirmation still may be required (a semi-automated task with confirmation).
- Level 4—Fully automated human tasks: The level of automation of the system is so high, that it is capable of performing human tasks on behalf of task owners. The overall business process execution is automated. This is only a long-term future vision, which may not even be always desirable.

The levels can coexist within the same system. Current software solutions for SOA and business process execution provide support for human tasks at the first level of automation only. This paper determines what types of human tasks should be considered for automation by such systems and demonstrates how a level three of human task support can be achieved. For this purpose a service-oriented architectural framework for human task execution (human task service) is proposed. It is based on a simple, efficient and holistic approach to dealing with human tasks and extending possibilities of their automation. Techniques to accomplish this are presented, among which one of the main mechanisms is automatic or semi-automatic decision making based on ontologies. This mechanism is enhanced with agent technology to represent different human agents. As a proof-of-concept, a system was developed for human task intensive business processes of an important electric power transmission company. The approach presented in this paper is generic and can be used for any type of industrial or industrial support business processes.

The paper is organized as follows. In the next section background information and related work are presented. In section three human tasks and their main characteristics are introduced. Their automation possibilities are examined and automation-relevant human task types are determined. Based on these characteristics the architectural framework is proposed and discussed in detail in section four. In the fifth section the proof-of-concept implementation is presented and analyzed. The last section contains the concluding remarks.

Manuscript received May 16, 2008; revised July 15, 2008. Current version published January 21, 2009. Paper no. TII-08-05-0068.R3.

A. Sasa and M. Krisper are with the Information Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana, 1000 Ljubljana, Slovenia (e-mail: ana.sasa@fri.uni-lj.si; marjan.krisper@fri.uni-lj.si).

M. B. Juric is with the Institute of Informatics, Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia (e-mail: matjaz.juric@uni-mb.si).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2008.2008641

II. Background and Related Work

A. *Service-Oriented Architecture and Business Process Automation*

The SOA provides a platform for closer alignment of IT with business processes. The objective is to increase the capability of a business system to address new business requirements in the short term by reusing existing business logic and data models, thus incurring only minimal cost, resource, and time overheads, while minimizing risks, especially when compared to rewriting entire application systems [26]. One of the most important SOA concepts is service composability [7], which allows structuring of services into different layers. It is recommended that lower service layers consist of application services and that the top layer consists of process services. Process services perform the orchestration of lower level services so that their operations are executed in a certain order. In this way the execution of a business process is automated.

Several languages for business process execution have emerged in the past decade. The best known are XLANG [10], WSFL [11], YAWL [12], and BPEL (BPEL4WS 1.0, BPEL4WS1.1, and WS-BPEL 2.0) [7], [21]. BPEL combines principles of XLANG and WSFL and is the prevalent orchestration language in SOA. It is an XML-based language for business processes execution based on Web services, which have been recognized as an important integration technology in industrial automation [4], [6].

The notions of business process automation and human tasks are closely related to workflow technology. Authors use the term *workflow* differently: as a synonym for a business process [31], as a specification of a process, as software that implements and automates a process, or software that supports the coordination and collaboration of people that implement a process [28]. However, the term *workflow* is traditionally more office-work oriented [29] and most Workflow Management Systems (WfMSs) deal with automation of document-intensive parts of business processes. In such parts documents, information and tasks are passed from one participant to another [27], [29], [30]. On the other hand business process execution systems are focused on orchestration of machine-performed activities and integration issues. Due to the need for overall business process support, nowadays both approaches have merged and the distinction has become less evident. In order to provide standardized support for human tasks in BPEL, a pair of specifications has been proposed and submitted to OASIS for standardization: BPEL4People and WS-Human Task specifications [18].

An important subdomain of business process automation is business contract automation. This field is concerned with assisting human users in performing actions that are compliant with business contracts and policies, monitoring for noncompliance, and providing guidance as to what needs to be done next in response to the satisfaction or violation of terms of the contracts or clauses of the policies [37].

In this paper we do not try to propose a completely different approach to implementing human tasks as part of the business process execution, but rather enhance the existing ones with higher level support for human tasks with the objective of automating their execution as much as possible. Due to the advantages of SOA, we focus on human task improvement in SOA based systems. Furthermore, the framework that we propose allows system implementation that is compliant with BPEL4People and Web Services Human Task specifications, and allows incorporation of business policies and rules into execution of human tasks. Our approach can also be applied to other types of human task systems, for example traditional workflow systems.

B. *Ontologies*

In the proposed framework, ontologies are used as a means of enabling human task support and automation. Lai defines the ontology as a means of enabling communication and knowledge sharing by capturing a shared understanding of terms that can be used both by humans and by programs [8]. There are several languages available for ontology representation, such as DAML, CGs, OIL, DAML+OIL, and OWL [8], [19], [25]. The Web Ontology Language (OWL) is a W3C's Semantic Web Activity effort and is expected to play an important role in helping automated processes to access information [48]. Even though our approach is not directly related to the Semantic Web vision, it is based on OWL due to its ability to represent a useful group of ontology features, a high level of support, and its XML foundations, which make it appropriate to be used in conjunction with other Web technologies [19]. More specifically, our approach is based on OWL's description logics (DL) based sublanguage (OWL DL), enhanced with Semantic Web Rule Language (SWRL) [20]. OWL ontology consists of Individuals, Properties, and Classes. Individuals (also known as instances) represent objects in the domain that we are interested in. Properties state relationships between individuals or between individuals and data values. OWL class defines a group of individuals that share certain properties [19]. SWRL is based on a combination of OWL DL and OWL Lite with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. It enables usage of Horn-like rules in OWL ontology—implications between antecedents and consequents, where the consequent holds true if the antecedent holds [20].

C. *Organizational Memory*

Abecker and Decker pointed out that analyzing the problems of expert systems had lead to the development of organizational memory information systems approaches [34]. The approach proposed in this paper uses the concept of ontology-based organizational memory (organizational ontology) as its basic component and combines ontologies with decision support, more specifically multi-attribute decision making. Multi-attribute decision making has been a topic of research for many decades; however, our work is based on multi-attribute decision modeling as presented in [1]. Unlike expert systems [49], our approach aims to support coordination of multiple human agents during a business process execution and to enable further automation of human tasks, rather than just providing decision guidance to a single human agent.

The concept of organizational memory, sometimes also referred to as corporate memory or organizational knowledge base, refers to stored information from an organization's history that can be brought to bear on present decisions [32] and supports sharing and reuse of individual and corporate knowledge and lessons learned [33]. Considerable research effort has been dedicated to this field of science; examples are [32]–[36]. However, there seem to be only few authors concerned with application of these approaches to the business process execution domain, similar to the proposal in this paper. Abecker *et al.* in [33] developed a model of an organizational memory system composed of three levels: 1) object level, which comprises manifold information and knowledge resources; 2) knowledge-description level, which enables intelligent access to a diversity of object-level sources; and 3) application level, in which processes and tasks are modelled and executed. Kaathoven *et al.* [38] systematically investigated the integration of organizational memory systems with WfMS. Based on this work, Reimer *et al.* proposed a model with the objective of assisting office-clerks in doing their job, offering them active support and guidance [39]. The main difference between these papers and ours is that our focus is not on support of human actors when performing tasks, but on further automation possibilities of their work with the methods applied to the organizational ontology.

Fox *et al.* [40] and more recently Ba *et al.* [41] are concerned with automation in the domain of decision making based on organizational memory. Fox *et al.* observe that the role of information systems should not be only that of providing simple repositories of data, but also of offering more sophisticated support for manual and automated decision making. They believe that information systems should be able to answer queries of explicit and implicit representations in their enterprise model. They present the TOVE Enterprise Modeling project as an answer [40]. Ba *et al.* propose a conceptual model and a Web-based architecture for implementation of an enterprise-wide knowledge network aimed at organizing and integrating model-based knowledge components from various sources and providing better knowledge access to decision makers located across the organization. The approach considers explicit knowledge expressed in a formalized representation and automatic building of models for the scenario given with a query [41]. The important differences in comparison with our approach are that we place the decision making in the business process context based on the SOA principles, that we focus on tacit knowledge and its translation into explicit knowledge, and that we propose a method which enables automatic querying and query answering, and thus a higher level of process execution automation. Furthermore, we do not try to propose an approach to integration of information from heterogeneous sources, which would be then used to support human actors at their work. Rather, we presume the use of the SOA integration platforms in order to gather information, which we then map to the organizational memory.

Organizational memories and business process execution systems are categories of Computer Supported Cooperative Work (CSCW) systems [50]. Even though our focus is not on CSCW aspects of the framework, the framework possesses several CSCW characteristics: it supports collaborative work, especially when a human task involves more than one participant, it allows partitioning of human work into units and its reintegration when the work is performed, and it supports reusability of information stored in the organizational memory.

D. Multi-Agent Systems

Interest in agent technologies has been rising over the past two decades, due to the wide range of their applicability. While there are many definitions of agents and multi-agent systems, the following are the most commonly referred to. An agent is a computer system that is situated in some environment, and is capable of autonomous actions in this environment in order to meet its design objectives [42]. Among other properties, often attributed to agents, there are also reactivity, proactiveness and social ability. A multi-agent system (MAS) is a system composed of cooperative or competitive agents that interact with one another in order to achieve individual or common goals [43]. In our framework, different human task owners are represented by agents. The objective is to enable support for human tasks with appropriate coordination characteristics.

A wide range of proposals for business process automation improvement based on agent technologies have been discussed, however, they approach it in a different way than we do. Most authors in this field are concerned with different proposals to administer or improve process composition with agents or multi-agent systems, for example [5], [9], [30], [31], [45]. Taveter and Wagner [44] have proposed the Radical Agent-Oriented Process (RAP) based on Agent-Object-Relationship (AOR) modeling, and the RAP/AOR methodology geared towards business process modeling, simulation and automation. Their approach is agent-oriented and not service-oriented. Other authors propose to implement Web services with agent technology in order to realize complex interaction and coordination of services, for example [23] and [24].

On the other hand, some authors discuss managing the organizational memory with multi-agent systems, for example [46] and [47]. Even though the agents in our system work as intermediaries between the business processes, human actors and the organizational ontology, their main role is to provide coordination support for human task execution. Therefore, our work is not closely related to this specific field.

III. Human Task Automation Possibilities

All business activity within a business system can be regarded as a set of different tasks, which can be either automated or performed by human participants. A task (also an activity for some authors) has many definitions. In this paper we regard a task as all the work that needs to be accomplished in order to transform its input into the required output.

Based on their composition, tasks can be elementary or composite. A BPEL business process instance is an example of a composite task: every service it invokes represents a subtask and the BPEL code implements their composition. Every composite task can be decomposed into elementary tasks. Consequently, if subtask composition can be automated then the problem of task automation translates into a problem of elementary task automation—or within our context, the problem of business process automation translates into a problem of elementary human task automation.

A composite task can be restructured in a way that the composition does not require human participation. If human participation is involved in a composition of certain subtask results, the human participation can be implemented as a subtask itself, while the composition is what connects these subtasks together. For example, let us say that 1) A and B are subtasks of a composite task C, 2) the result of task C is chosen among the results of the subtasks A and B, and 3) this choice has to be made by a human participant. In this case, we can introduce another subtask D, in which the human participant makes the decision.

Thus, A, B, and D are subtasks of task C. Depending on the composition required it can be automated with one of the business process execution languages (invoking a subtask as a service), e.g., BPEL, or using some other technology if more applicable and available. Thus, in this paper the focus is on further automation of elementary human tasks.

An elementary human task instance is performed by exactly one task owner. If a task instance is performed by two or more task owners, the task can be decomposed into such subtasks that their instances are performed by individual task owners.

Elementary human tasks can be classified into two groups. The first group comprises tasks for which a predefined procedure determines the way they should be performed. In this case eligible task owners are all those who are familiar with the procedure. Presuming that they perform the procedure as required, the task result does not depend on a specific individual; i.e., if another person with the knowledge about the procedure would perform it correctly, the result would be the same. Some of the most common subgroups of this group of elementary human tasks are: 1) physical tasks, which require the task owner to perform some physical task and return the result based on its completion; 2) information system tasks, which require the task owner to use different information systems, in order to be able to provide a task result; and 3) communicative tasks, which require its task owner to communicate a piece of information to another person. If a task with a predefined execution procedure is not automated, this is due to some practical reasons or technological or physical limitations. However, if it was automated, it would represent an automated service independent of any human participants and therefore not belong to the human task service. As a result, automation of these tasks is not a functionality of the human task service and they are enabled by delegating them to their task owners.

The other group of elementary human tasks represents human tasks, for which their result depends on every individual task owner. The procedure used to accomplish a task is not predefined, but is the result of their own information, perception of the world, their experience etc. These are mental process tasks, as we name them, and they cannot be automated easily. In this case, the difficulty arises not from practical reasons or physical limitations, but from the fact that their execution depends on people's mental processes. In order to be able to automate these tasks, a system should be able to act on behalf of the task owners. This includes performing tasks based on the task owner's mental characteristics, such as their knowledge and preferences. This paper focuses on automation of mental process human tasks and represents a step closer to this vision.

IV. Architectural Framework

Based on the principles of SOA, one of the goals of our research was to define a generic architectural framework for the human task service, which can deal with any type of elementary human tasks. Fig. 1 illustrates the general structure of the proposed architectural framework. It consists of two main components: the human task service and the organizational ontology. The human task service is responsible for human task execution. The organizational ontology comprises all relevant organizational information needed for enabling task execution and automation. Both components are discussed in more detail in the remainder of the section.

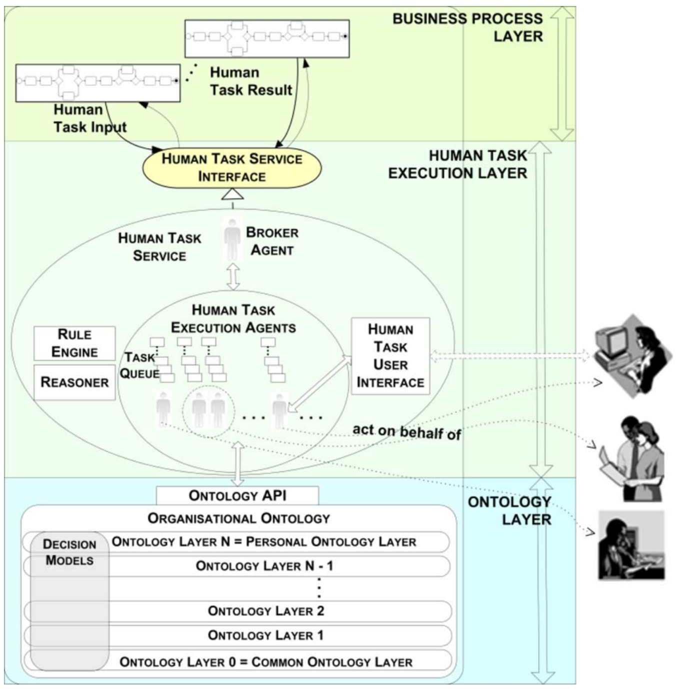

Fig. 1. General structure of the proposed system.

TABLE I
Description of Some Human Task Properties

Property	Description
Process data input	Task execution may require data gathered during process execution. In this case, process data needs to be provided.
People query	A specific person or people for human task execution cannot be always known at design time. Therefore, an approach based on the concept of a people query is used. They represent queries upon the organisational ontology, which is discussed in more detail in section "Organisational ontology".
Task output query	Description of what is expected as a result of the task. It is used as a query upon the organisational ontology
Business process administrator	If the human task service returns a fault, task properties together with the fault message can be used to invoke another human task assigned to the business administrator. The business administrator should take an appropriate action, for example, terminate the process or tackle the problem and retry the task. A people query is specified to determine the Business process administrator.

A. *Human Task Service*

Task properties required as input of the human task service are: people query, task output query, process data input (optional), timeframe, expected duration (optional), task description, priority and business process administrator. The types of task input and output variables have to correspond to the ontology schema as discussed in section "Knowledge Elicitation and Ontology Schema". Some properties require special explanations, which are given in Table I.

Human task service is implemented by a multi-agent system comprising a broker agent and human task agents. When the human task service is invoked, the broker agent receives the task, determines its task owners and assigns it to human task agents, who act on behalf of the task owners. By introducing only one agent type for human task execution, every agent is able to represent anyone of the human participants. Because the focus is on automation of mental process tasks, this is not a drawback. Every human task agent has its task queue. New tasks are assigned to agents based on deadlines and priorities of the tasks in their task queues. If there is no agent available, a new human task agent is instantiated and assigned the task.

Human task agents are responsible for enabling elementary human task execution. Process data input information is added to the ontology before human task execution. An agent tries to find an elementary human task result from the information in the ontology by using a rule engine and a reasoner. The result is determined by the task's output query property. If the result cannot be reached in this way, i.e., there is not enough information in the ontology for an agent to be able to come to a result, the task cannot be automated. In this case, the agent gathers the available relevant information from the ontology and passes it together with the task description, deadline and priority to the human task user interface, which delegates the task to the task owner.

Determining a task owner based on the people query is not different than any human task executed by human task agents: required output is determined by the people query, its timeframe and deadline are determined by the overall human task timeframe, priority is determined by its human task priority and the task owner is the business process administrator. Therefore, to determine task owners, the broker agent assigns a human task agent with the corresponding task, which is in this case called a people resolution task. If a result cannot be determined, the task is delegated to the responsible business process administrator.

B. Organizational Ontology

In order to be able to deal with different human tasks, the human task execution layer does not depend on a specific organization and can be used without any further adoption. By contrast, the organizational ontology is the organization specific part of the system and has to be developed for every organization separately. It provides all the necessary organization dependent information for the human task execution layer. In this section the organizational ontology framework is represented.

The ontology layer, illustrated in Fig. 1, demonstrates the basic structure of an organizational ontology. Hierarchical structure is proposed due to some important advantages, such as reducing search complexity and promoting reuse of stored knowledge [8]. Let us say that organizational structure element (OSE) is a generic term for any kind of organizational structure element. It may be an organization itself, a department, a role, a group or any other possible structural element used in an organization. An organizational ontology comprises ontologies belonging to different OSEs. Every ontology of the overall organizational ontology belongs to exactly one OSE. If OSE_1 is a part of OSE_2, OSE_1 can use the OSE_2's ontology. The common ontology layer represents the organization's common knowledge, such as its organizational structure and common business policies for example. The personal ontology layer comprises personal ontologies. They contain knowledge concerning people in the business system. An OSE's ontology is composed of two layers: the base ontology layer and the decision model layer. The base ontology layer contains information about the concepts of the corresponding OSE's domain and uses ontologies belonging to the OSEs, of which it is part. The decision model layer contains decision models, which are built upon the base ontology layer concepts and support decision processes required for executing mental process human tasks.

In this section we focus on ontology based decision modeling, which is the means for human task support and automation by the human task service. Other aspects of the organizational ontology, such as ontology development and ontology evolution processes, are beyond the scope of this paper. In the remainder of the section, first the notation is explained and afterwards the ontology-based decision modeling is presented.

1) Notation: For presentation of OWL DL and SWRL examples more readable notations are used. Bidirectional translations exist between both pairs of notations. For OWL DL examples notation is DL-based. In order to represent SWRL rules, variables are used with classes as unary relations, properties as binary relations and SWRL built-ins as n-ary relations, depending on the number of attributes used. All the variables used in an SWRL rule are tied to the quantificator $\forall$. For every n-ary relation, for which $\mathrm{n} > 2$, at most (n-1) arguments in a relation can be fixed with a specific individual or datatype. An antecedent of an SWRL rule is a conjunction of such relations and a consequent is a conjunction of class and property relations.

For task output queries and people queries the same notation as for antecedents of SWRL rules is used.

2) Ontology-Based Decision Models: In this section ontology-based decision models, which provide support and automation for mental process tasks, are discussed. They are based on the multi-attribute decision making approach proposed in [1], which was shown to be highly applicable for industrial decision making. First, a brief overview of multi-attribute decision making is given. Then, the main characteristics and steps of the decision making process are described. In the last subsection, two types of decision modeling processes are presented.

a) Multi-Attribute Decision Making: In complex decision-making processes the goal is to choose the best option from a set of possible options and to rank them from the best to the worst. To achieve this evaluation models are used. Evaluation models are a technique which helps to estimate the worthiness (utility) of each option for the decision maker [1]. If $\mathrm{A} = \{a_1, a_2 \cdots a_N\}$ is the set of options and $\mathrm{X} = \{\mathrm{x}_1, \mathrm{x}_2 \cdots \mathrm{x}_\mathrm{M}\}$ the set of attributes, x_i translates an option to a certain value: $\mathrm{x}_\mathrm{i} : \mathrm{A} \rightarrow \mathrm{D}_\mathrm{i}$, where D_i is the range of ith's attribute, $\mathrm{i} \in 1 \ldots \mathrm{M}$. This value represents the evaluation of the option with regard to the ith attribute. The final decision is the rational decision, i.e., the choice of the option a from A, which is the most preferred. This is defined with the utility function $\mathrm{v}(a)$ and the preference relation P as follows:

$$\begin{aligned}&\mathrm{v}(a) = \mathrm{v}(\mathrm{x}_1(a), \mathrm{x}_2(a) \cdots \mathrm{x}_\mathrm{M}(a)), a \in \mathrm{A}\\&a\mathrm{P}b \leftrightarrow \mathrm{v}(a) > \mathrm{v}(b); a\mathrm{P}b \cdots a \text{ is preferred over.}\end{aligned} \quad (1)$$

Models are not necessarily quantitative. Qualitative models use qualitative variables which are presented by words rather than numbers. In the proposed model, qualitative values are used.

Typically, steps of the decision making process are as follows: 1. Problem identification, 2. Attributes identification −2.1 Attributes definition (usually this means creating a list of all the attributes), 2.2 Defining a decision tree, 2.3 Defining the range

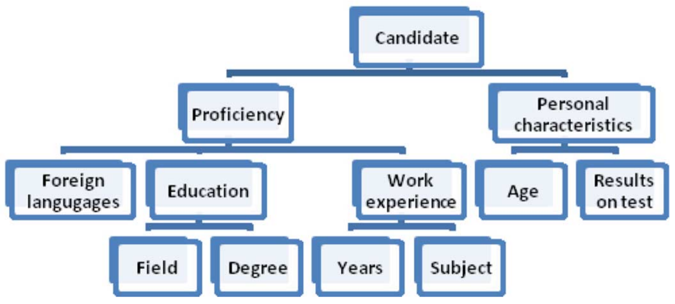

Fig. 2. Example decision tree.

of the attributes in the decision tree, 3. Decision rules definition, 4. Options description, and 5. Option evaluation and analysis.

b) Ontology-Based Decision Making Process: Concepts of OWL ontologies and concepts of multi-attribute decision making do not have a direct translation. Therefore, one of the requirements for defining the ontology-based decision making process was unambiguous correspondence of decision model elements in ontology-based and in multi-attribute decision making. The remainder of the section describes required steps of the proposed ontology-based decision making process and how the concepts in each step correspond to the concepts of the classical multi-attribute decision making process. For better understanding, an example of choosing the best job applicant is shown. The example ontology is simplified, as it serves only for explanation of the concepts. For a comparison with classical multi-attribute decision making a corresponding decision tree is given (Fig. 2).

Steps of the ontology-based decision making process

1) *Problem identification*: In OWL ontology, decision options correspond to instances of a class. This step requires identification of the class which represents the set of options.

In the case of job applicants' ontology, we would like to find the most appropriate candidate. Let us presume that the class which represents the set of options is the class Candidate.

2) *Attributes identification*: Attribute candidates are all classes and properties which have a direct or indirect relationship with the options' class. This step requires identification of classes and properties which influence the decision and identification of their relationships to the options' class. Classes and properties chosen as attributes correspond to the list of attributes, before they are structured in a decision tree. In the decision tree these attributes can be found in the leaves of the tree.

An example of an attribute in the decision tree is Degree. An example of a corresponding class in the ontology is a class with the same name. An example of the relationship between this class and the class Candidate is

$$\mathrm{Candidate} \sqcap \forall \mathrm{hasDegree.Degree}. \tag{2}$$

3) *Attributes structuring and attribute values identification:* This step comprises grouping of the related attributes and their structuring into subclasses. The difference in the name of the step in comparison to the multi-attribute decision making is due to the fact that there is no explicit decision tree in the ontology. For each derived attribute in the decision tree (all the attributes in the tree which are not leaves) its corresponding domain class in the ontology has to be identified (attribute class). If the attribute refers to a property, the corresponding class is its domain. A group of subclasses for each derived attribute has to be created. The number of subclasses of an attribute class equals the number of possible attribute values. For the attribute class, a covering axiom including all attribute groups of subclasses has to be added. If $\mathrm{N1}, \mathrm{N2} \cdots \mathrm{Nn}$ are attribute subclasses and M is the attribute class, then the covering axiom is

$$\mathrm{N1} \sqcup \mathrm{N2} \sqcup \cdots \sqcup \mathrm{Nn} = \mathrm{M}. \tag{3}$$

In addition, disjoint subclasses have to be defined for every attribute group subclass

$$\forall \mathrm{Nk}, \mathrm{Nl} : \mathrm{Nk} \sqcap \mathrm{Nl} = \perp, \mathrm{k}, \mathrm{l} \in 1 \ldots \mathrm{n}, \mathrm{k} \neq l. \tag{4}$$

In the example decision tree there are five derived attributes: Proficiency, Education, Work experience, Personal characteristics, and Candidate. If we focus on Education, let us suppose that the corresponding ontology class has the same name, i.e., Education. Possible values have to be identified based on which we create the corresponding subclasses (of the class Education), for example: VeryAppropriateEducation, AppropriateEducation, and NotAppropriateEducation. An example of a derived attribute which refers to a property is the Work experience derived attribute corresponding to the hasYearsOfWorkExperience datatype property. If its domain is the class Candidate, then the corresponding subclasses are subclasses of the class Candidate, for example: NotExperiencedCandidate, AveragelyExperiencedCanidate, and ExperiencedCandidate. In both examples covering axioms have to be added to the derived attribute classes Education and Candidate, for example

$$\begin{aligned}&\mathrm{Education}\\ &\quad = \mathrm{VeryAppropriateEducation}\\ &\qquad \sqcup \mathrm{AppropriateEducation} \sqcup\\ &\mathrm{NotAppropriateEducation}.\end{aligned} \tag{5}$$

In addition subclasses have to be specified as disjoint.

4) *Decision rules definition*: In the proposed ontology decision rules are represented by OWL's necessary and sufficient conditions of a class or by SWRL rules. If an individual satisfies the necessary and sufficient conditions of a class, then it must be a member of the class. Use of complementOf expression is recommended because it helps to decrease the number of conditions and possible errors.

In certain cases, such as when comparing datatype property values between individuals to classify them, OWL's necessary and sufficient conditions are not expressive enough and SWRL rules are used. Either necessary and sufficient conditions or SWRL rules have to be defined for all the attribute subclasses created in the step 3.

Example of an SWRL based decision rule is given for the hasYearsOfWorkExperience datatype property

$$\begin{aligned}&\mathrm{Candidate}(?x) \wedge \mathrm{hasYearsOfWorkExperience}(?x, ?y)\\ &\quad \wedge \mathrm{swrlb: greaterThan}(?y, 5)\\ &\qquad \to \mathrm{ExperiencedCandidate}(?x).\end{aligned} \tag{6}$$

5) *Option description*: We assume to work on an existing ontology. This means that the options are already described. Otherwise, the individuals have to be created and described.

In the example, options are all the individuals of the class Candidate.

6) *Option evaluation*: Option evaluation is done by an ontology reasoner and with a rule engine. Reasoner and rule engine compute the inferred types based on decision rules and thus options are sorted into corresponding subclasses. The subclass which we created to represent the most appropriate option now contains instances which represent the solution to the decision problem.

Supposing the attribute subclasses of the class Candidate are: ExcellentCandidate, VeryGoodCandidate, AppropriateCandidate, NotAppropriateCandidate, and that all the decision rules have already been defined; after computing the inferred types, instances belonging to the subclass ExcellentCandidate are the best options.

c) Options Based Decision Making Process Types: Two kinds of decision making processes can be distinguished [2]: 1) All the options exist and are described in the ontology; to make a decision the decision maker only has to define the derived attributes and their values; 2) Options are not sufficiently described to be able to make a decision; decision makers have to describe them based on their opinion and then define the derived attributes and their values.

With the proposed ontology-based approach to decision making we can gain more in the first type of decision making process, which can theoretically be completely automated. Once the decision maker has determined the attributes and their values, the task can be automated until they wish to change their criteria. An example is a process of hiring a new employee. When evaluating the applicants, the responsible task owner(s) would have to define their own criteria for evaluation of the applicants. The best way to achieve this is that the first time a person needs to evaluate the options, the person builds a decision model. As all the relevant information about the applicants can be acquired during the preceding activities in the process, this information is added to the ontology with the human task input. In every future process execution a human task agent representing the human decision maker would take the already developed decision model of this employee from their personal ontology, compute the best candidate and thus perform the evaluation automatically on the task owner's behalf.

In the second type of decision making, the way options are described depends on every individual decision maker. Even though options are a part of the common ontology, the related properties and classes that are the subject of the decision are part of the task owner's individual ontology. They have to be determined as part of each decision process. An example of such an activity can be found in the domain of conference papers management. In order to review the papers, reviewers are assigned papers for review. In comparison with the previous example, options (in this case papers) cannot be equally described. In the case of the job applicants, all the decision attributes already make part of the ontology before the human task is executed. They are the same for every decision maker. Their decision models differ in derived attributes (and so their values and decision rules). By contrast, in the case of paper evaluation every reviewer defines not only the derived attributes, but also

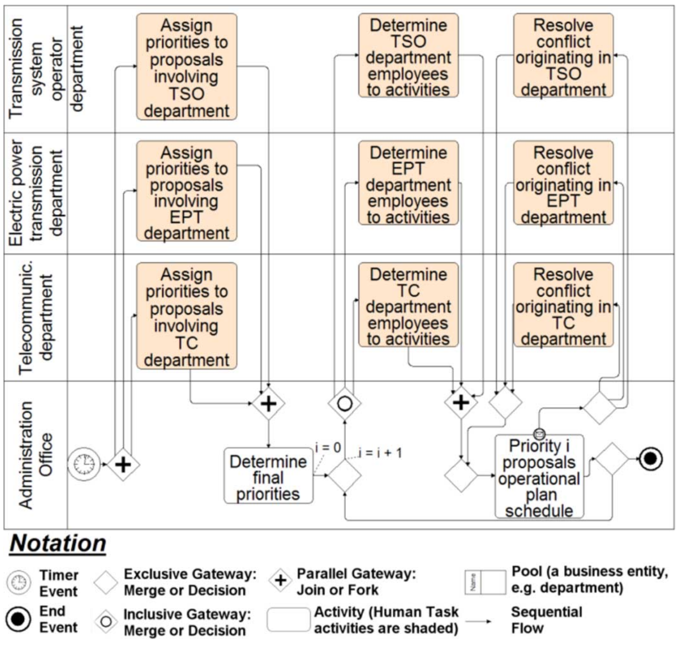
Transmission system operator department
Electric power transmission department
Telecommunic. department
Administration Office
Assign priorities to proposals involving TSO department
Assign priorities to proposals involving EPT department
Assign priorities to proposals involving TC department
Determine TSO department employees to activities
Determine EPT department employees to activities
Determine TC department employees to activities
Resolve conflict originating in TSO department
Resolve conflict originating in EPT department
Resolve conflict originating in TC department
Determine final priorities
i = 0
i = i + 1
Priority i proposals operational plan schedule
Notation
Timer Event
End Event
Exclusive Gateway: Merge or Decision
Inclusive Gateway: Merge or Decision
Parallel Gateway: Join or Fork
Activity (Human Task activities are shaded)
Name
Pool (a business entity, e.g. department)
Sequential Flow

Fig. 3. Create weekly operational plan process (Human task activities are shaded).

the base attributes of the decision model since they depend on each reviewer. In this case, one of the benefits of the proposed system is that the decision model can be reused, each time with different options.

These examples showed how ontology-based decision modeling can automate a decision making human task, and if automation is not possible, how it improves the decision process by reusable decision models for the task owner. Due to responsibility issues, even if a task can be automated, a confirmation still may be desired.

3) Knowledge Elicitation and Ontology Schema: Similar to other expert systems, an inherent limitation of the proposed approach is the problem of verbalizing and documenting the rules the experts implicitly use. Reasons why this problem can arise are implicit rules, too many rules, complex rule interactions, limited expressiveness of the rule language and other factors.

In order for process data input to be added to the ontology, its variables have to correspond to elements defined in the ontology schema. The ontology schema is an XML schema in which ontology classes and properties are represented as elements. In this manner process data input represents individuals and their properties. If the process data input variable does not correspond to the ontology schema, it cannot be added to the ontology. If the task cannot be automated, it is only passed to the human task owner.

V. Proof of Concept and Analysis

As a proof of concept, the proposed system has been implemented for a "Fault resolution" business process of an electric power transmission company. This is a human task intensive process. One of the things it involves is determining whether a fault has to be resolved immediately or should be included in the next week's operational plan. In this section we focus on the "Create weekly operational plan" subprocess. It is illustrated in Fig. 3 using Business Process Modeling Notation

TABLE II
EXAMPLE DECISION RULES

MediumPriorityFault(?x) ∧ MediumExperienceWithXS12_TSE(?z) ∧ HighestRatingSX12_TSE(?z) ∧ playsRole(?z, ?a) ∧ canResolve(?a, ?x) ∧ BeginnerWithSX12_TSE(?y) ∧ playsRole(?y, ?a) ∧ canResolve(?a, ?y) ∧ TSE_Team(?b) → hasMember1(?b, ?z) ∧ hasMember2(?b, ?y) ∧ hasMostAppropriateTeamForResolution(?x, ?b)	MediumPriorityFault(?x) ∧ HighlyExperiencedWithXS12_TSE(?z) ∧ HighestRatingSX12_TSE(?z) ∧ playsRole(?z, ?a) ∧ canResolve(?a, ?x) ∧ NoExperienceWithXS12_TSE(?y) ∧ playsRole(?y, ?a) ∧ canResolve(?a, ?y) ∧ TSE_Team(?b) → hasMember1(?b, ?z) ∧ hasMember2(?b, ?y) ∧ hasVeryAppropriateTeamForResolution(?x, ?b)
MediumPriorityFault(?x) ∧ MediumExperienceWithXS12_TSE(?z) ∧ AppropriateRatingXS12_TSE(?z) ∧ playsRole(?z, ?a) ∧ canResolve(?a, ?x) ∧ BeginnerWithSX12_TSE(?y) ∧ playsRole(?y, ?a) ∧ canResolve(?a, ?y) ∧ TSE_Team(?b) → hasMember1(?b, ?z) ∧ hasMember2(?b, ?y) ∧ hasMostAppropriateTeamForResolution(?x, ?b)	MediumPriorityFault(?x) ∧ HighlyExperiencedWithXS12_TSE(?z) ∧ HighestRatingSX12_TSE(?z) ∧ playsRole(?z, ?a) ∧ canResolve(?a, ?x) ∧ HighlyExperiencedWithXS12_TSE(?y) ∧ HighestRatingSX12_TSE(?y) ∧ playsRole(?y, ?a) ∧ canResolve(?a, ?y) ∧ differentFrom(?z, ?y) ∧ TSE_Team(?b) → hasMember1(?b, ?z) ∧ hasMember2(?b, ?z) ∧ hasAcceptableTeamForResolution(?x, ?b)
MediumPriorityFault(?x) ∧ HighlyExperiencedWithXS12_TSE(?z) ∧ HighestRatingSX12_TSE(?z) ∧ playsRole(?z, ?a) ∧ canResolve(?a, ?x) ∧ BeginnerWithSX12_TSE(?y) ∧ playsRole(?y, ?a) ∧ canResolve(?a, ?y) ∧ TSE_Team(?b) → hasMember1(?b, ?z) ∧ hasMember2(?b, ?y) ∧ hasVeryAppropriateTeamForResolution(?x, ?b)	MediumPriorityFault(?x) ∧ BeginnerWithSX12_TSE(?z) ∧ playsRole(?z, ?a) ∧ canResolve(?a, ?x) ∧ BeginnerWithSX12_TSE(?y) ∧ differentFrom(?z, ?y) ∧ playsRole(?y, ?a) ∧ canResolve(?a, ?y) ∧ TSE_Team(?b) → hasMember1(?b, ?z) ∧ hasMember2(?b, ?y) ∧ hasLessAppropriateTeamForResolution(?x, ?b)

(BPMN) [22]. Activity proposal priorities are assigned by managers of the Electric power transmission department, the Transmission system operator department, the Telecommunications department and managers of the corresponding subdepartments. Priority assignment is a decision process in which activity proposals are evaluated based on the importance of the activity for the business system. Managers develop their own decision models based on the proposal description, deadline, their knowledge about the activity domain, and other relevant criteria. Priority assignments are implemented as human tasks. As an activity can be a matter for two or more departments, it is possible that it is assigned two or more different priorities. In this case final priorities have to be determined, which is an automated procedure.

When all the priorities are set, proposals with the highest priorities are chosen to be included in the weekly operational plan. Sometimes employees who should perform the chosen activities are already defined in the proposal. More often an employee responsible for an activity selectively decides who should perform it. In this case this is a decision-based human task assigned to the responsible person. As different departments have different requirements, distinct human tasks had to be implemented. Example decision rules for a selective assignment of employees for resolution of a medium priority fault are provided in Table II. A corresponding human task input is presented in Table III. It can be observed that the decision maker gives preference to quicker development of team members than to loading the most highly rated and experienced engineers.

After the employee assignment the activities to be included in the next week's operational plan are scheduled. An automated service is used for this purpose. In case of conflicts between activities, a fault is thrown and managers of departments responsible for the activity are informed in this case. The reason for the exception has to be identified and taken care of properly. Conflict resolutions are implemented as human tasks, but due to their unpredictable nature they are not automated.

There are several OWL reasoners and SWRL rule engines that can be used, for example RacerPro [14], Jess [15], Pellet [16], and FaCT++ [17]. For the human task service implementation we have chosen the RacerPro reasoner and Jess rule engine, because of their efficiency and support. For similar reasons and wide recognition JADE [13], [51] has been used as the agent platform. To prevent that the decision problem becomes too complex, we limited the maximum number of children nodes in the decision tree, maximum number of attribute values and the maximum depth of the tree. After implementing the "Create weekly operational plan" business process using the proposed human task service, the process has been included in the business activity monitoring. An important condition was that human tasks have to be performed effectively and with comparable outcomes to those without the human task service. Let us define task completion time as the time needed from human task service invocation until the results are received. Business activity monitoring results for the business process instances were taken for a time scale of 25 weeks, and for every week's process execution average human task completion times were determined.

TABLE III
EXAMPLE INPUT OF A HUMAN TASK

Process Data Input	/
People Query	TSOperationalManagementDepartmentManager(?x)
Task description	Assign the most appropriate transmission system maintenance team for resolution of fault Fault_IH_32.
Task output query	TSE_Team(?x) ∧ hasTeamForResolution(Fault_IH_32,?x)
Time Frame	From 2007-05-26T08:00:00 to 2007-05-27T14:00:00
Expected duration	0.5 h
Priority	3
Business Proc. Administr.	TransmissionSystemOperationalProcessesAdministrator(?x)

If a task is a non-automated task (nau), completion time (tct) is the sum of the time needed for the human participant to claim the task (tclt) and the actual execution time (tet)

$$\mathrm{tct} = \mathrm{tclt} + \mathrm{tet}. \tag{7}$$

Claim time is the time between task delegation to the participant and the time the participant actually acknowledges he

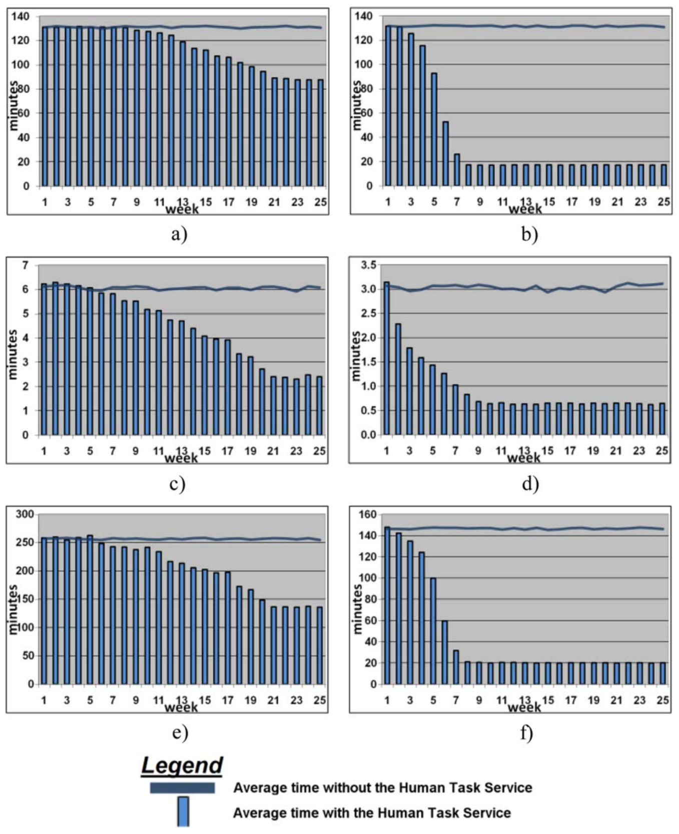

Fig. 4. Performance measurements. (a) Average task claim time for "Assign priorities to proposals" human tasks. (b) Average task claim time for "Determine employees to perform an activity" human tasks. (c) Average task execution time per proposal for "Assign priorities to proposals" human tasks. (d) Average task execution time per activity for "Determine employees to perform an activity" human tasks. e) Average task completion time for "Assign priorities to proposals" human tasks. f) Average task completion time for "Determine employees to perform an activity" human tasks.

would execute the task. If a task is automated (au), its claim time is zero because the system begins immediately with its execution and the task completion time equals the automated task execution time.

Fig. 4 represents the results for the "Create weekly operational plan" process activities for which the human task service improves their performance. Due to the similar nature of tasks, average times were calculated for all three types of human tasks involving priority assignment and for all three types of human tasks involving the employees for activities determination (Fig. 3). In the remainder of the section all the human tasks involving priority assignment, are characterized as "Assign priorities to proposals" human tasks, and all human tasks involving determining employees for activities as "Determine employees to perform an activity" human tasks. While being effective in achieving the desired outcomes, a trend of improving average task completion, claim and execution times followed by a stabilisation can be observed for both groups of human tasks.

In the first weeks of the human task service usage average execution times are slightly longer than without it (Fig. 4). The reason for this is that decision models are still being developed, while tasks are not yet automated. However, once they are built, the improvement in time is quickly perceivable.

Let us say that $\mathrm{p_{au}}$ indicates the percentage of human tasks which are automated and $\mathrm{p_{nau}}$ the percentage of human tasks which are not automated. Then the following can be stated:

$$\begin{aligned}\mathrm{tclt_{average}} &= \mathrm{p_{au}} * \mathrm{au\_tclt_{average}} \\ &\quad + \mathrm{p_{nau}} * \mathrm{nau\_tclt_{average}} \\ &= \mathrm{p_{au}} * 0 + \mathrm{p_{nau}} * \mathrm{nau\_tclt_{average}} \\ &= \mathrm{p_{nau}} * \mathrm{nau\_tclt_{average}}. \end{aligned} \tag{8}$$

Percentages of automated and non-automated tasks are

$$\mathrm{p_{nau}} = \frac{\mathrm{tclt_{average}}}{\mathrm{nau\_tclt_{average}}}, \quad \mathrm{p_{au}} = 1 - \mathrm{p_{nau}}. \tag{9}$$

Average claim time of non-automated tasks is the same with or without the use of human task service. This would not be true if average claim times differed considerably from week to week. However, among the human task properties claim time depends only on the timeframe. For human tasks instances originating from the same "Create weekly operational plan" activity, this property is always the same. Therefore, their claim times are comparable. For the "Assign priorities to proposals" human tasks, the average claim time without the human task service was 131.1 min. For the "Determine employees to perform an activity" human tasks, this time was 131.7 min. With the human task service these times were improved to 87.8 mi and 17.4, respectively. If "Assign priorities to proposals" human task is referred to as $t1$ and "Determine employees to perform an activity" human task is referred to as $t2$, then their percentages of automated and non-automated tasks in the stabilization phase are

$$\begin{aligned}\mathrm{p_{nau}}(t1) &= \frac{87.8}{131.1} = 67.0\% \\ \mathrm{p_{au}}(t1) &= 1 - 0.670 = 33.0\% \\ \mathrm{p_{nau}}(t2) &= \frac{17.4}{131.7} = 13.2\% \\ \mathrm{p_{au}}(t2) &= 1 - 0.132 = 86.8\%. \end{aligned} \tag{10}$$

The main reason for the difference in percentages between the tasks lies in the fact that most of the "Determine employees to perform an activity" human tasks belong to the first decision making process type as defined in section "Options Based Decision Making Process Types", while the "Assign priorities to proposals" human tasks mostly belong to the second type (base attributes of a decision depend on individual task owner).

Let $\mathrm{tct_{wth}}(t)$ indicate the average task t completion time without the human task service, and $\mathrm{tct_{st}}(t)$ the average task t completion time with the human task service (in the stabilisation phase). Then, the improvement $\Delta\mathrm{tct_{avg}}(\mathrm{t})$ is

$$\begin{aligned}\Delta\mathrm{tct_{avg}}(t1) &= \mathrm{tct_{wth}}(t1) - \mathrm{tct_{st}}(t1) \\ &= 256.7\mathrm{min} - 136.2\mathrm{min} = 120.5\mathrm{min} \\ \Delta\mathrm{tct_{avg}}(t2) &= \mathrm{tct_{wth}}(t2) - \mathrm{tct_{st}}(t2) \\ &= 146.8\mathrm{min} - 20.4\mathrm{min} = 126.4\mathrm{min}. \end{aligned} \tag{11}$$

The average completion time has been improved by $120.5/256.7 = 46.9\%$ for "Assign priorities to proposals" human tasks and by $126.4/146.8 = 86.1\%$ for "Determine employees to perform an activity" human tasks. The reason why the second percentage is higher is that the great majority of employee assignments to activities can be automated $(\mathrm{p_{au}}(t2) = 86.8\%)$, and approximately a third of priority assignments can be automated $(\mathrm{p_{au}}(t1) = 33.0\%)$, while the others are either semi-automated with confirmation or non-automated. However, both completion times have improved significantly.

Similar observations can be made about average task execution times. As the number of proposals and activity assignments can differ from week to week, a comparison of average task execution time per a decision process is more appropriate. Let $\mathrm{tet_{wth}}(t)$ indicate average task t execution time per a decision process without the human task service and $\mathrm{tet_{st}}(t)$ the average task t execution time per a decision process with the human task service (when the stabilization phase is reached). Then the execution time improvement per decision process $\Delta\mathrm{tet_{avg}}(t)$ is

$$\begin{aligned}\Delta\mathrm{tet_{avg}}(t1) &= \mathrm{tet_{wth}}(t1) - \mathrm{tet_{st}}(t1)\\ &= 6.1\ \mathrm{min} - 2.4\ \mathrm{min} = 3.7\ \mathrm{min}\\ \Delta\mathrm{tet_{avg}}(t2) &= \mathrm{tet_{wth}}(t2) - \mathrm{tet_{st}}(t2)\\ &= 3.0\ \mathrm{min} - 0.6\ \mathrm{min} = 2.4\ \mathrm{min}. \qquad (12)\end{aligned}$$

The average execution time has been improved by $3.7/6.1 = 60.7\%$ for "Assign priorities to proposals" human tasks and by $2.4/3.0 = 80.0\%$ for "Determine employees to perform an activity" human tasks.

We have demonstrated considerable time improvements for automated decisions and semi-automated decisions with confirmation of mental process tasks. This is important for the business process invoking the human task service, as well as for the task owners responsible for task execution. Besides the improved business process performance there is another advantage of the proposed system and organizational ontology framework. A very important characteristic of decision making is the ability to explain how a decision has been reached. Within the business process and SOA context this is especially important when studying the process execution. In the case of current human task implementations, a human task result is passed to the process and it cannot be ascertained directly how it has been reached. Even if a decision has been supported with an information system it is difficult to discover a direct connection. In the proposed system the human task output contains the ontology used for obtaining the result, which enables the traceability and explanation of how and why a decision has been reached. This means that the exact reason for a certain conclusion of a process instance (which depends on a decision taken) can be identified, reviewed and studied for possible improvements.

VI. Conclusion

In this paper a service-oriented architectural framework for automation and support of human tasks in business processes was proposed. We have determined what types of elementary tasks are relevant for such automation; i.e., mental process tasks. There are several important novelties and strengths of the approach. For the purpose of providing support and automation for mental process tasks, an organizational ontology comprising ontology-based decision models belonging to different organizational structure elements is proposed. As decision making is one of the most important activities, our approach provides support and automation for a large part of mental process human tasks. However, due to responsibility issues, sometimes semi-automation with confirmation is preferred over complete automation of human tasks. With extended possibilities of human task automation and improvement of their completion times the overall business process performance is improved and a step closer is made to the vision of end-to-end business process automation. Furthermore, our approach enables a high level of decision traceability, from explanation of how a decision has been reached to proactive identification of decision consequences for business process execution.

References

[1] M. Bohanec and V. Rajkovič, "Multi-attribute decision modeling: Industrial applications of DEX," *Informatica*, vol. 23, no. 4, pp. 487–491, Oct. 1999.

[2] A. Šaša, M. B. Jurič, and M. Krisper, "Agents and people activities in web-services based business processes," in *Proc. 2007 IEEE Int. Conf. Digital Ecosystems and Technologies (IEEE-DEST 2007)*, Cairns, Australia, 2007.

[3] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *IEEE Trans. Ind. Informat.*, vol. 1, no. 1, pp. 62–70, Feb. 2005.

[4] A. P. Kalogeras, J. V. Gialelis, C. E. Alexakos, M. J. Georgoudakis, and S. A. Koubias, "Vertical integration of enterprise industrial systems utilizing web services," *IEEE Trans. Ind. Informat.*, vol. 2, no. 2, pp. 120–128, May 2006.

[5] S. Ali, B. Soh, and T. Torabi, "A novel approach toward integration of rules into business process using an agent-oriented framework," *IEEE Trans. Ind. Informat.*, vol. 2, no. 3, pp. 145–154, Aug. 2006.

[6] R. Zurawski, "Integration technologies for industrial automated systems: Challenges and trends," in *Integration Technologies for Industrial Automated Systems (Industrial Information Technology)*. Boca Raton, FL: CRC Press, 2006.

[7] M. B. Jurič, B. Mathew, and P. Sarang, *Business Process Execution Language for Web Services*, 2nd ed. Birmingham, U.K.: Packt, 2006.

[8] L. F. Lai, "A knowledge engineering approach to knowledge management," *Inf. Sci.*, vol. 177, no. 19, pp. 4072–4094, Oct. 2007.

[9] Q. Xu, R. Qiu, and F. Xu, "Agent-based workflow coordination for supply chain management," *Trans. Nanjing Univ. Aeronaut. Astronaut.*, vol. 20, no. 1, pp. 112–117, Jun. 2003.

[10] S. Thatte, Microsoft Corporation, *XLANG: Web Services for Business Process Design*, Jun. 2001. [Online]. Available: http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

[11] F. Leymann, *Web Service Flow Language (WSFL 1.0)*, May 2001. [Online]. Available: http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[12] YAWL Team, *YAWL: Yet Another Workflow Language.* [Online]. Available: http://www.yawl-system.com/.

[13] Telecom Italia, *Java Agent DEvelopment Framework.* [Online]. Available: http://jade.tilab.com.

[14] Racer Systems GmbH & Co. KG, *Racer*. [Online]. Available: http://www.sts.tu-harburg.de/~r.f.moeller/racer/.

[15] Sandia National Laboratories, *Jess*. [Online]. Available: http://www.jessrules.com/jess/index.shtml.

[16] Clark & Parsia LLC, *Pellet*, [Online]. Available: http://pellet.owldl.com/.

[17] D. Tsarkov and I. Horrocks, FaCT++ [Online]. Available: http://owl.man.ac.uk/factplusplus/.

[18] Active Endpoints, Adobe, BEA, IBM, Oracle, and SAP AG, *WS-BPEL Extension for People, Version 1.0*, Jun. 2007. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people/.

[19] W3C, *OWL Web Ontology Language Overview*, Feb. 2004. [Online]. Available: http://www.w3.org/TR/owl-features.

[20] W3C, *SWRL: A Semantic Web Rule Language, Combining OWL and RuleML*, May 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/.

[21] OASIS Standard, *Web Services Business Process Execution Language Version 2.0*, Apr. 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

[22] OMG, Business Process Management Initiative OMG Final Adopted Specification, *Business Process Modeling Notation (BPMN) Specification*, Feb. 2006. [Online]. Available: http://www.bpmn.org/.

[23] Y. Li, K.-M. Chao, M. Younas, Y. Huang, and X. Lu, "Modeling e-marketplaces with multi-agents web services," in *Proc. 11th Int. Conf. Parallel and Distributed Systems*, Fukuoka, Japan, 2005, pp. 175–181.

[24] T. I. Zhang and H. Jiang, "A framework of incorporating software agents into SOA," in *Proc. Artificial Intelligence and Soft Computing (ASC 2005)*, Benidorm, Spain, 2005.

[25] I. Horrocks and C. Welty, "Digital libraries and web-based information systems," in *Description Logic Handbook: Theory, Implementation and Applications*, F. Baader, D. L. McGuiness, and D. Nardi, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[26] D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*. Upper Saddle River, NJ: Prentice-Hall PTR, 2004, pp. 1–11.

[27] E. A. Stohr and J. L. Zhao, "Workflow automation: Overview and research issues," *Inf. Syst. Front.*, vol. 3, no. 3, pp. 281–296, Sep. 2001.

[28] D. Georgakopoulos, M. Hornick, and A. Sheth, "An overview of workflow management: From process modeling to workflow automation infrastructure," *Distrib. Parallel Databases*, vol. 3, no. 2, pp. 119–153, Apr. 1995.

[29] C. Plesums, "Introduction to workflow," in *Workflow Handbook 2002*, L. Fischer, Ed. Lighthouse Point, FL: Future Strategies, 2002.

[30] M. B. Blake and H. Gomaa, "Agent-oriented compositional approaches to services-based cross-organizational workflow," *Dec. Sup. Syst.*, vol. 40, no. 1, pp. 31–50, Jul. 2005.

[31] M. B. Blake, "Agent-based workflow configuration and management of on-line services," in *Proc. Int. Conf. Electronic Commerce Research (ICECR-4)*, Dallas, TX, 2001, pp. 567–588.

[32] J. P. Walsh and G. R. Ungson, "Organizational memory," *Acad. Manage. Rev.*, vol. 16, no. 1, pp. 57–91, Jan. 1991.

[33] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek, "Toward a technology for organizational memories," *IEEE Intell. Syst.*, vol. 13, no. 3, pp. 40–48, 1998.

[34] A. Abecker and S. Decker, "Organizational memory: Knowledge acquisition, integration, and retrieval issues," in *Proc. XPS*, Würzburg, Germany, 1999, pp. 113–124.

[35] C.-C. Huang, T.-L. Tseng, and A. Kusiak, "XML-based modeling of corporate memory," *IEEE Trans. Syst., Man, Cybern.*, vol. 35, no. 5, pp. 629–640, Sep. 2005.

[36] R. Dieng, O. Corby, A. Giboin, and M. Ribière, "Methods and tools for corporate knowledge management," *Int. J. Human Comput. Stud.*, vol. 51, no. 3, pp. 567–598, Sep. 1999.

[37] A. S. Abrahams, D. M. Eyers, and J. M. Bacon, "An asynchronous rule-based approach for business process automation using obligations," in *Proc. 2002 ACM SIGPLAN Workshop Rule-Based Programming*, Pittsburgh, PA, 2002, pp. 93–103, ACM.

[38] R. van Kaathoven, M. A. Jeusfeld, M. Staudt, and U. Reimer, "Organizational memory supported workflow management," in *Proc. 4. Int. Tagung Wirtschaftsinformatik Electronic Business Engineering*, A.-W. Scheer and M. Nuttgens, Eds. Heidelberg, Germany: Physica-Verlag, 1999, pp. 543–563.

[39] U. Reimer, A. Magelish, and M. Staudt, "EULE: A knowledge-based system to support business processes," *Knowledge-Based Syst.*, vol. 13, no. 5, pp. 235–331, Oct. 2000.

[40] M. S. Fox, M. Barbuceanu, M. Gruninger, and J. Lin, "An organisation ontology for enterprise modeling," in *Simulating Organizations: Computational Models of Institutions and Groups*, M. Prietula, K. Carley, and L. Gasser, Eds. Menlo Park, CA: AAAI/MIT Press, 1998.

[41] S. Ba, K. R. Lang, and A. B. Whinston, "Compositional enterprise modeling and decision support," *Inf. Syst. E-Business Manage.*, vol. 6, no. 2, pp. 137–160, Mar. 2008.

[42] M. Wooldridge, *An Introduction to MultiAgent Systems*. Chichester, U.K.: Wiley, 2002.

[43] P. Giorgini and B. Henderson-Sellers, "Agent-oriented methodologies: An introduction," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Hershey, PA: Idea Group, 2005.

[44] K. Taveter and G. Wagner, "Towards radical agent-oriented software engineering process based on AOR modelling," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Hershey, PA: Idea Group, 2005.

[45] N. R. Jennings, T. J. Norman, and P. Faratin, "Autonomous agents for business process management," *Int. J. Appl. Artif. Intell.*, vol. 14, no. 2, pp. 145–189, 2000.

[46] F. Gandon, A. Poggi, G. Rimassa, and P. Turci, "Multi-agents corporate memory management system," *J. Appl. Artif. Intell.*, vol. 16, no. 9 and 10, pp. 699–720, Oct.–Dec. 2002.

[47] A. Abecker, A. Bernardi, and L. van Elst, "Agent technology for distributed organizational memories: The frodo project," in *Proc. 5th Int. Conf. Enterprise Information Systems*, Angers, France, 2003, pp. 3–10.

[48] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language," *Web Semant.: Sci., Serv., Agents World Wide Web*, vol. 1, no. 1, pp. 7–23, Dec. 2003.

[49] J. C. Giarratano and G. Riley, *Expert Systems, Principles and Programming*. Pacific Grove, CA: Brooks/Cole, 2005.

[50] P. H. Carstensen and K. Schmidt, "Computer supported cooperative work: New challenges to systems design," in *Handbook of Human Factors*, K. Itoh, Ed. Tokyo, Japan: Asakura, 2003.

[51] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Chichester, U.K.: Wiley, 2007.

Ana Sasa (M'07) received the B.Sc. degree in computer and information science from University of Ljubljana, Ljubljana, Slovenia, in 2005, where she is currently pursuing the Ph.D. degree in information systems and decision making.

She is employed as a researcher at the Information Systems Laboratory at the Faculty of Computer and Information Science, University of Ljubljana. Her primary research interests include service-oriented architectures, enterprise architectures, business process modeling and execution, and decision support systems.

Matjaz B. Juric received the B.Sc. degree in computer and information science and the Ph.D. degree from the University of Maribor, Maribor, Slovenia, in 1996 and 1999, respectively.

He is an Associate Professor and the Head of the Laboratory for Technologies of Communication at the Faculty of Electrical Engineering and Computer Science, University of Maribor. He is the Head of the SOA Competency Centre at the Science Park of the University of Maribor. He has authored several SOA books, such as *SOA Approach to Integration* (Birmingham, U.K.: Packt Publishing, 2007) and *BPEL Cookbook* (Birmingham, U.K.: Packt Publishing, 2006). The latter has been awarded as the best SOA book in 2007.

Dr. Juric is a member of the BPEL Advisory Board.

Marjan Krisper (M'05) received the M.Sc. degree in information systems engineering from the University of Ljubljana, Ljubljana, Slovenia, in 1977 and the Ph.D. degree in expert systems from the University of Belgrade, Belgrade, Yugoslavia, in 1989.

He is an Associate Professor, the Chair of Information Science, and the Head of the Information Systems Laboratory at the University of Ljubljana, Faculty of Computer and Information Science. His research interests include information systems, information systems development methodologies, information systems strategic planning, and electronic business.

Dr. Krisper is a member of the Association of Information Systems and a senior member of the Project Management Institute Slovenian Chapter.