UDC 004.414.2:005.4
Original Scientific Paper

Service-Oriented Architectural Framework for Support and Automation of Collaboration Tasks

Ana Šaša *ana.sasa@fri.uni-lj.si*
Information Systems Laboratory,
Faculty of Computer and Information Science,
University of Ljubljana, Ljubljana, Slovenia

Marjan Krisper *marjan.krisper@fri.uni-lj.si*
Information Systems Laboratory,
Faculty of Computer and Information Science,
University of Ljubljana, Ljubljana, Slovenia

Abstract

Due to more and more demanding requirements for business flexibility and agility, automation of end-to-end industrial processes has become an important topic. Systems supporting business process execution need to enable automated tasks execution as well as integrate human performed tasks (human tasks) into a business process. In this paper, we focus on collaboration tasks, which are an important type of composite human tasks. We propose a service-oriented architectural framework describing a service responsible for human task execution (Human task service), which not only implements collaboration tasks but also improves their execution by automated and semi-automated decision making and collaboration based on ontologies and agent technology. The approach is very generic and can be used for any type of business processes. A case study was performed for a human task intensive business process from an electric power transmission domain.

Keywords: business process automation, collaboration task, service-oriented architecture, ontology, multi-agent system

1. Introduction

Due to more and more demanding requirements for business flexibility and agility, automation of end-to-end industrial processes has become an important topic. Nowadays the prevailing approach 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 [11]. However, when striving towards complete business process automation it has to be taken into account that not all tasks can be automated. These are tasks which require human interaction (human tasks). They have to be performed by human participants and can represent very 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 a business process, SOA information systems supporting business process execution need to indicate a human participant responsible for task execution (task owner) when to perform a task, what is required to be done and after its completion task result needs to be passed back to the process. In this paper, we focus on collaboration tasks. A collaboration task is a human task which requires involvement of two or more human participants who have to collaborate in order to complete the task.

SOA systems can provide different levels of support for human participants performing human tasks. We have identified four levels of support for human tasks by such systems [20]:

- *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.

Current software solutions for SOA and business process execution implementation usually provide support for human tasks on the first level of automation. This paper demonstrates how a level three support can be achieved for collaboration tasks. The presented framework is an extension of our *Service-oriented framework for human task support and automation* that was discussed in [20], which discussed automation of elementary human tasks. In this paper, we extend the service-oriented architectural framework for human task execution (human task service) with a higher level of support and automation of collaboration tasks. We have performed a case study of our approach on a human task intensive business process from the domain of electric power transmission. The approach for development of this system was very generic and can be used for any type of industrial and industrial support business processes. In this paper we present this simple, but efficient and holistic approach to dealing with collaboration tasks in SOA systems and extending possibilities of their automation. Techniques to accomplish this are presented, among which one of the main mechanisms is ontology-based automation of protocol-based collaboration.

The rest of the paper is organised as follows. In the next section related work is presented. In Section three, human tasks and their main characteristics are introduced. Collaboration tasks are discussed as a special type of composite human tasks. In Section four, the architectural framework is discussed in detail. In the fifth Section, an example scenario from our case study is presented. Finally the last section contains the concluding remarks.

2. Related work

The approach presented in this paper is closely related several research and technical areas. However, the analysis of existing literature has shown that extending possibilities of automation of collaboration tasks as part of business process automation has not yet been discussed from our perspective.

In order to enable execution of business processes several workflow languages have emerged in the past decade among which the best known include XLANG [9], WSFL [10], YAWL [30] and BPEL (BPEL4WS 1.0, BPEL4WS1.1 and the latest version WS-BPEL 2.0) [15]. BPEL combines principles of XLANG and WSFL and is nowadays the most accepted and supported by a significant number of tools. It is an XML-based language designed primarily to support automated business processes based on Web services, which have been recognised as an important integration technology in industrial automation [2], [4]. BPEL covers many aspects of business processes, even though it does not cover human interactions. In order to provide standardized support for human tasks in BPEL, a pair of specifications has been developed and accepted by OASIS: BPEL4People [11] and WS-HumanTask specifications [12]. They cover various aspects of human interactions with a process, such as roles describing how people can interact with a process, interaction patterns, different ways of integrating human tasks into the process and operations for client applications. In our research all these aspects have been thoroughly considered and the proposed framework allows implementation of a system compliant with these specifications.

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 [25]. 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 [26]. In our framework, different human task owners are represented by agents. The objective is to enable support for human tasks with appropriate coordination characteristics and to automate collaboration tasks.

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 [3], [7], [27], [28]. Taveter and Wagner [29] have proposed the Radical Agent-Oriented Process (RAP) based on Agent-Object-Relationship (AOR) modelling, and the RAP/AOR methodology geared towards business process modelling, 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 [22] and [23].

In the proposed framework, ontologies are used as a means of enabling collaboration 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 [5]. There are several languages available for ontology representation, such as DAML, CGs, OIL, DAML+OIL, and OWL [5], [13], [24]. Our approach is based on the OWL (Web Ontology Language) 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 [13].

3. Human tasks in SOA systems

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 can be defined as all the work which needs to be accomplished in order to transform its input into the required output [18]. In order for a task to be accomplished, different resources may be required and input of a task has to contain everything that is necessary for its execution. Therefore a task can be represented as a function which transforms its inputs into an output or result; if t is a task, inp_t its input variable and r_t it's result variable then:

$$t(inp_t) = r_t. \tag{1}$$

Based on their composition tasks can be either elementary or composite. An elementary task is a task which cannot be decomposed into subtasks, while a composite task is a task which is composed of one or more subtasks which are required to be performed in order to come to a result. A subtask can be an elementary task or a composite task. Every composite task ct can be defined by a following composition tuple:

$$(st_{ct}, cf_{ct}) = ((st_{1ct}, st_{2ct} \dots st_{nct}), (cf_{1ct}, cf_{2ct} \dots cf_{(n+1)ct})), \tag{2}$$

where $n \in N$ is the number of ct's subtasks, st_{ct} a vector of ct's subtasks and cf_{ct} a vector of ct's composition functions. Composition functions are responsible for result to input translations between subtasks and for obtaining the final composite task's result. Let say that $inp_{ct} = r_{0ct}$ and $r_{ct} = inp_{(n+1)ct}$. If inp_{ict} and r_{ict} are task st_{ict}'s input and output variables respectively: $st_{ict}(inp_{ict}) = r_{ict}$, then for $\forall i \in 1..(n+1)$:

$$inp_{ict} = cf_{ict}(r_{m_0ct}, r_{m_1ct} \dots r_{m_kct}) = cf_{ict}\left(r_{1cf_{ict}}, r_{2cf_{ict}} \dots r_{m_{(k+1)}cf_{ict}}\right) = cf_{ict}(r_{cf_{ict}})$$

$$k \in 0..(i-1), \forall l \in 1..k\colon\ m_l \in 0..(i-1)\ \wedge\ r_{m_lct} = r_{(l+1)cf_{ict}},$$

$$\bigwedge_{l1 \neq l2} m_{l_1} \neq m_{l_2},\ l_1, l_2 \in 1..k,$$

$$\forall u \in 0..n\colon r_{uct}\colon \exists z, t \in 1..u\colon r_{tcf_{zct}} = r_{uct} \tag{3}$$

The last condition in (3) is necessary because without it a subtask could result in an output which is never used in order to obtain the composite task's result and thus based on the composite task definition it should not be part of the task.

In this manner a BPEL business process instance is an example of a composite task where every service it invokes represents a subtask and the BPEL code implements the composition functions.

Based on (1-3), a composition tuple of an elementary task et is: $(st_{et}, cf_{et}) = (\emptyset, (et))$. Let define a level $lvl(t)$ of a task t as follows:

$$lvl(t) = \begin{cases} 0, & t\ is\ an\ elementary\ task \\ max_i\{lvl(st_{it})\} + 1, & otherwise \end{cases} \tag{4}$$

Therefore, every subtask of a composite task is at least one level lower of the original task and the lower level limit is 0. This implies that every composite task can be decomposed into elementary tasks, and consequently if composition functions 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.

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.

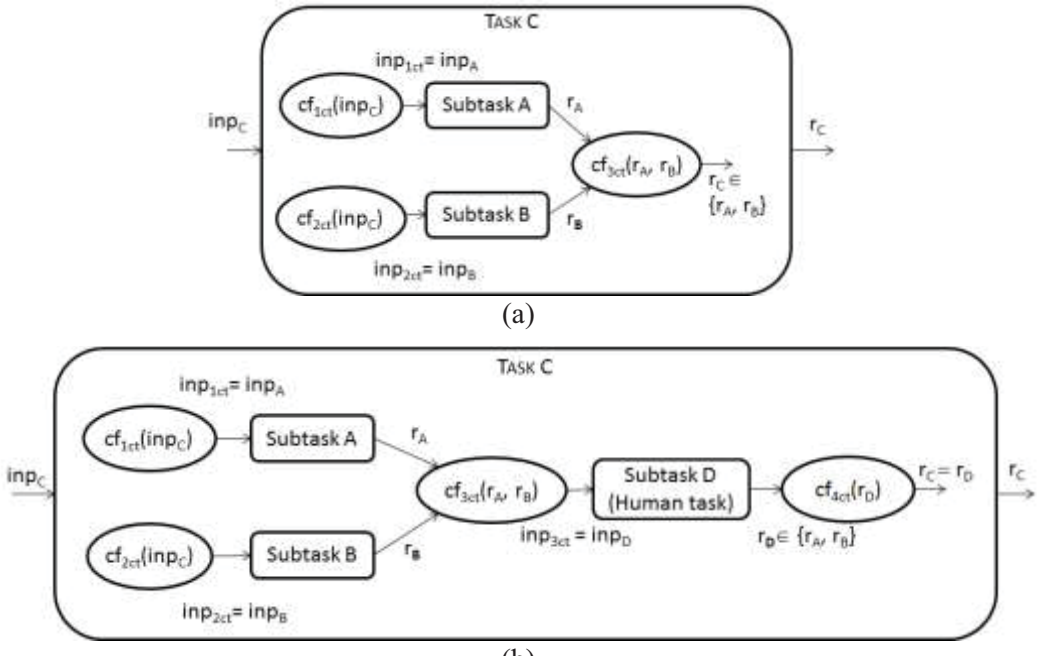

Figure 1. Restructuring of a composite task

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, 3) this choice has to be made by a human participant (Figure 1a, in which cf_{3ct} is the composition function that involves the choice). 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 (Figure 1b). 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.

However, due to the principles of service orientation, such as reusability, statelessness and loose coupling, an appropriate level of provided composition automation by the Human task service needs to be determined. On the one hand, if the Human task service implements only elementary human tasks, their composition would not differ from any other service composition and could be implemented with any of the workflow composition languages for example. On the other hand, this would not be appropriate for the following types of composite tasks:

i) Several very common and highly reusable task compositions typical for human tasks can be identified and should therefore be a part of the Human task service and not be the responsibility of a business process designer. They require passing a human task to different actors based on a certain pattern, such as sequential or parallel composition, for example. In the remainder of the paper they are referred to as basic workflow patterns. Based on (1-3) the following conditions describe composition tuples of basic workflow pattern tasks:

$$\forall i_1, i_2, \in 1..n\colon \mathrm{st}_{i_1 \mathrm{ct}} = \mathrm{st}_{i_2 \mathrm{ct}} \wedge \forall i_3 \in 1..n\colon cf_{i_3 ct}(x) = x \tag{5}$$

For example, composition functions of a sequential workflow pattern task (Figure 2) have the following form:

$$\forall i \in 1..(n+1)\colon inp_{ict} = cf_{ict}(r_{(i-1)ct}) = r_{(i-1)ct} = r_{1\,cf_{ict}} = r_{cf_{ict}} \tag{6}$$

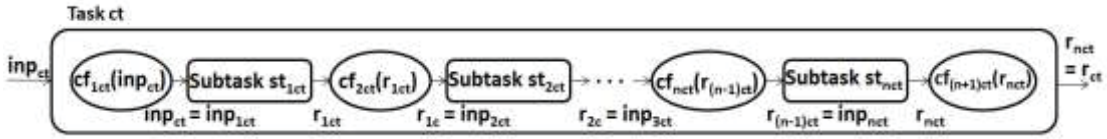

Figure 2: Sequential workflow pattern

One can observe that all the composition functions in this pattern are identity functions, for example in case of n = 2:

$$ct(inp_{ct}) = inp_{3ct} = cf_{3ct}(r_{2ct}) = r_{2ct} = st_{2ct}(inp_{2ct}) = st_{1ct}(inp_{2ct}) = st_{1ct}(r_{1ct}) = st_{1ct}(st_{1ct}(r_{0ct})) = st_{1ct}(st_{1ct}(inp_{ct})) = st_{1ct} \circ st_{1ct} \tag{7}$$

The basic workflow patterns are a concept that is already supported by many different commercial SOA systems. Therefore, in this paper we place them into the overall scope of composite human task types, but in the remainder of the paper we do not discuss them in detail.

ii) A collaboration task is composed of communication tasks (performative tasks) and message formation tasks during which collaboration participants perform everything necessary to compose the next message they will send. If st_{ict}, as defined in (2), is a message formation task which needs to be performed by a task owner to and st_{ict}'s input is defined as in (3), then for a collaboration task:

$$\forall i \in 1..(n+1): inp_{ict} = cf_{ict}\left(r_{m_0ct}, r_{m_1ct} \dots r_{m_kct}\right) = \left(r_{m_0ct}, r_{m_1ct} \dots r_{m_kct}\right) = r_{cf_{ict}}, \quad (8)$$

where $r_{cf_{ict}}$ is a vector containing all the previous results of the collaboration subtasks needed for st_{ict} accomplishment, including results of subtasks previously performed by to; for example when negotiating about a price, the price a participant proposes depends on all the prices they have already proposed and the responses they have received. If we would like to provide a loosely coupled service, it should be stateless [6]. This would require that a record of the collaboration history (or state) is not kept within the Human task service. When a human task is performed by a human collaboration participant this is not a problem because they would of course normally be able to remember the collaboration history. However as we strive to automate human tasks and our system acts on behalf of task owners this input has to be passed as a whole to the task from a business process execution instance. This implies that to would have to trust another party about their own preceding actions. Due to trust issues and better performance, collaboration tasks are implemented by autonomous agents representing different collaboration participants. Therefore collaboration tasks are part of the functionality of the proposed Human task service.

To sum up, there are two types of composite tasks that belong to the overall Human task service: basic workflow pattern human tasks and collaboration tasks. In the following section, we discuss our architectural framework and how its support and automation of collaboration tasks.

4. Architectural framework

Based on the principles of SOA [8] 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 human tasks, while providing user support and automation for certain types of human tasks. As established in the previous section human task automation is relevant for elementary human tasks, basic workflow pattern tasks and collaboration tasks. For more information about automation of elementary human tasks please refer to our previous work presented in [18].

Figure 3 illustrates the general structure of the proposed Human task service. Different components of the system can be classified into three main layers. The business process layer indicates how the Human task service can be used as a part of a business process, the human task execution layer is responsible for human task execution, and the ontology layer comprises all relevant organisational information needed for enabling this execution and possible automation.

Human task service is composed of the Human task workflow pattern manager and the Human task execution service. The Human task workflow pattern manager is responsible for basic workflow pattern composition of tasks. The Task recognition component takes care of initiating the correct pattern and the Pattern based processes component realises composition functions (2-3) for the patterns, for example as BPEL processes; for every non-pattern based task the Human task execution service is invoked and for any basic pattern subtask the Human task service is reinvoked.

Task properties required to be passed as input of the Human task service are: people link, task output query, process data input (optional), protocol (optional), timeframe, expected duration (optional), task description, priority, basic workflow pattern and business process administrator. Some properties require special explanations, which are given in Table 1. The protocol and people link property schema types are illustrated in Figure 4. In the remainder of the section, the Human task execution service and organizational ontology are discussed in more detail.

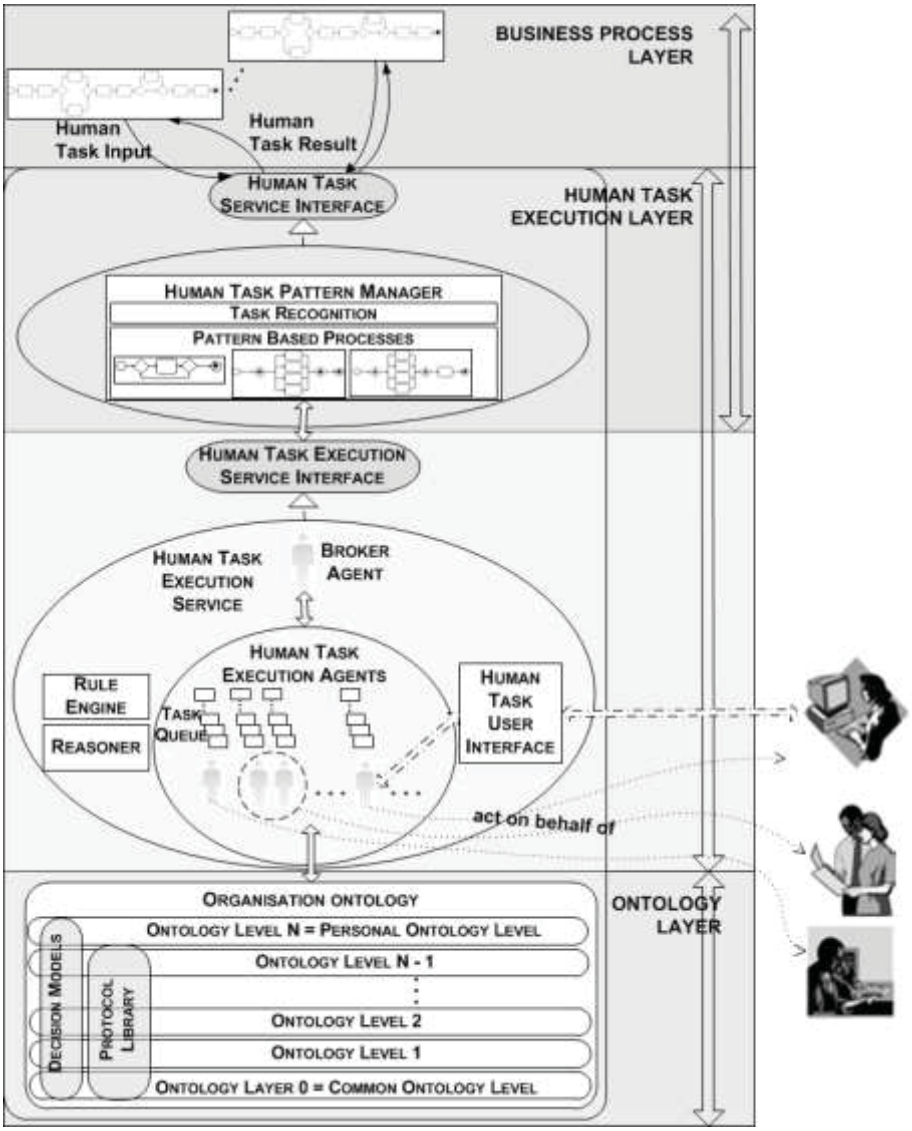

Figure 3. General structure of the proposed system

Property	Description
Process data input	Task execution may require data gathered during process execution. In this case process data contained in some process variable needs to be provided.
People link	It cannot be expected that a specific person or people for human task execution can be always known at design time. Therefore we use an approach based on a people link as defined in [19]. If a protocol is defined people link has to be provided for every role in the protocol. People links are bound to people queries which are in our case represented as queries upon the organisational structure ontology, which is discussed in more detail in section 4.2.
Protocol	If the human task is a collaboration task a protocol has to be provided. Protocol determines legal interactions between collaboration participants. A people link has to be provided for every role in the protocol together with a number of required individuals for a role. A process designer can choose among available protocols from the protocol library based on its unique name and description as discussed in section 4.2.1.
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 execution service returns a fault task properties together with the fault message are used to invoke another human task assigned to the business administrator who should take an appropriate action, for example terminate the process or tackle the problem and retry the task. An actual person or a people query can be specified.

Table 1: Description of some human task properties

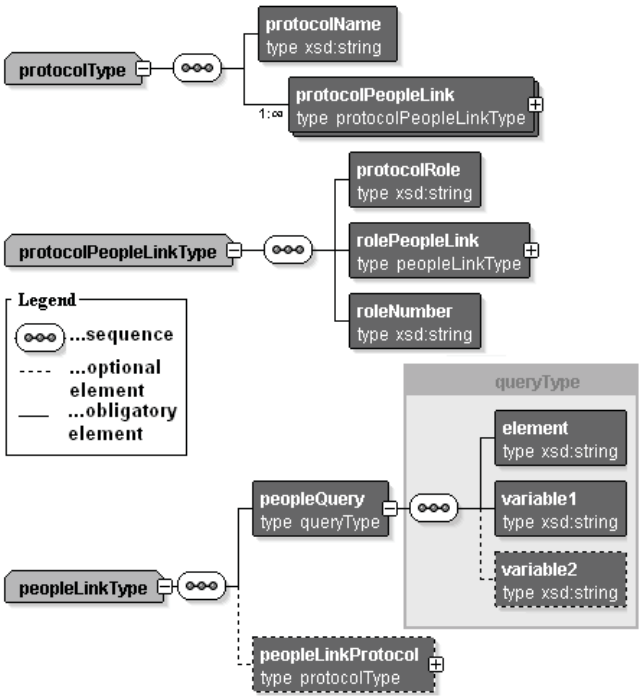

Figure 4: Protocol and people link input types

4.1. Human task execution service

Human task execution service is implemented by a multi-agent system comprising a broker agent and human task execution agents. When the Human task execution 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. Every human task agent has its task queue. Agents are assigned new tasks 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 thus responsible for enabling elementary human task and collaboration task execution. In both cases, before the execution process data input information is added to the ontology. An agent tries to find an elementary human task result in the ontology, using a reasoner. 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 and the available relevant information is gathered and passed together with the task description, deadline and priority to the task owner who needs to provide it.

When performing a collaboration task, the broker agent translates it into sequences of subtasks performed by each of its task owners during which participants, represented by human task agents, maintain the result history. These sequences are composed of message formation tasks (which do not differ from any other elementary human tasks or collaboration tasks) and performative tasks. When a performative task makes part of a collaboration task, its goal is not only to deliver a piece of information to another collaboration participant but to provide an input for the next task they have to perform. If this next task is automated by an agent there is no need for delivering the message to the actual person the agent is representing (even though it can be if desired). If it cannot be automated, it should be delegated to the human participant with the input for the task. Therefore in both cases, the human recipient of the performative task does not have to receive the message explicitly. For these reasons and strong support for collaboration provided with agent technologies , when performative tasks make part of a collaboration task, they are automated in the proposed system.

Determining a task owner is not different than any human task that human task execution agents need to perform: required output is determined by the people link, it may have a protocol, its timeframe and deadline are determined by the overall human task timeframe and priority is determined by its human task priority; the only difference is that there is no task owner. Therefore, to determine task owners the broker agents 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 cannot be delegated to the task owner and a fault is returned.

In case a message formation task of a collaboration task is a collaboration task itself the people resolution task will not return specific individuals but rather a group of individuals; in this case the human task execution agent acts on behalf of a group. Collaboration subtasks are performed as invocations of the Human task execution service. In this way task hierarchy is executed until a collaboration task ct' of level 1 is reached:

$$lvl(ct') = 1. \tag{9}$$

4.2. Organisational ontology

In order to be able to deal with different human tasks, the human task execution layer is generic and does not depend on a specific organisation. On the contrary, the organisational ontology is the organisation specific part of the system and has to be developed for every organisation separately. It provides all the necessary organisation dependent information for the human task execution layer to be able to enable execution and possibly automation of required tasks. In this section the organisational ontology framework is represented.

Ontology level illustrated in Figure 3 demonstrates the basic structure of an organisational ontology. Hierarchical structure is proposed due to some important advantages, such as reducing search complexity and promoting reuse of stored knowledge [14]. Different levels within the organisational ontology are used to represent this hierarchy. Let us say that organisational structure element (OSE) is a generic term for any kind of organisational structure element. It may be an organisation itself, a department, a role, a group or any other possible structural element used in an organisation. An organisational ontology comprises ontologies belonging to different OSEs. Every ontology of the overall organisational ontology belongs to exactly one OSE. If OSE1 is a part of OSE2, OSE1 can use the OSE2's ontology. The common ontology layer represents the organisation's common knowledge, such as its organisational 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. In order to develop the ontology, any of the existing ontology development methodologies can be used. We do not try to propose a new methodology, but rather allow the designer to choose the methodology that is most appropriate for a given environment. The decision model layer contains decision models, which are built upon the base ontology layer concepts and support decision processes required for executing human tasks. A detailed discussion of these processes can be found in [20].

The protocol library is the main mechanism that enables support and automation of collaboration tasks. In the remainder of the section, first ontology language and notation used are explained, and afterwards the protocol library is presented.

4.2.1. Language and notation

Due to the advances in the Semantic Web community, high level of support and its XML foundations, which make it appropriate to be used in conjunction with other Web technologies, our approach is based on OWL (Web Ontology Language) 2.0 [25], more specifically its description logics (DL) based sublanguage OWL DL enhanced with SWRL

(Semantic Web Rule Language) [26]. OWL ontology consists of Individuals, Properties and Classes. Individuals (also known as instances) represent objects in the domain that we are interested in. Properties are binary relations between individuals – i.e. properties link two individuals together. OWL classes are interpreted as sets that contain individuals. They are defined using formal descriptions that state precisely the requirements for membership of the class.

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 an OWL ontology. These rules are of the form of an implication between an antecedent and consequent, where the consequent holds if the antecedent holds [26]. In the proposed ontology SWRL rules are used for a specific kind of decision rules and for the protocol specification.

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 a SWRL rule are tied to the quantificator ∀. For every n-ary relation, for which n>2, at most (n-1) arguments in a relation can be fixed with a specific individual or datatype. An antecedent of a 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.

4.2.2. Protocol library

Different protocols may be needed to accomplish different collaboration tasks. If protocols were implemented in human task execution agents then adding a new protocol or changing an existing protocol would require substantial changes to the multi-agent system of the Human task execution service. As business systems need to be as agile as possible and adapt to changes quickly this would not be acceptable. For this, it is important that for a human task agent it is transparent what protocol they have to conform to and whether it changes or not. In order to solve this issue we introduce a protocol library, which is an important part of the organisational ontology.

Different approaches for protocol modelling exist; some are based on constraining the possible sequences of communication acts using transition diagrams [8], [5], while other approaches are based on logical representation of states, which can be derived from actions performed at the previous states [29], [21]. In the context of this paper communication act sequences constraints and state representations are used in order to propose modelling of protocols based on their role in a collaboration task as a subcategory of composite tasks. For this purpose some new terms need to be introduced. A protocol p determines: 1) a vector of constraint functions $c = (c_1, c_2 \dots c_r), \forall c_i: T \to R_{c_i}$, where T represents the domain containing tasks and R_{c_i} is the range of the constraint function c_i, and 2) a set:

$$TCP_p = \{(pr_{11}, pr_{12} \dots pr_{1r}), (pr_{21}, pr_{22} \dots pr_{2r}) \dots (pr_{s1}, pr_{s2} \dots pr_{sr})\}, \forall i \in 1..s, \forall j \in 1..r: pr_{ij} \subseteq R_{c_j}^m. \quad (10)$$

A collaboration subtask type can be one of the performative task types or of a message formation task type. A protocol only concerns performative collaboration subtasks. Every communication act involves two parties; therefore there are two performative task types for every communication act type. Let $PT = \{t_1, t_2 \dots t_{n_{PT}}\}$ be the set of all performative task types, let $st_{ct} = (st_{1ct}, st_{2ct} \dots st_{nct})$ be a collaboration subtask vector defined as in (1-3) and (2) and tt a mapping from st_{ict} to its type. Then every st_{ct} has a corresponding performative task vector $pst_{ct} = (pst_{1ct}, pst_{2ct} \dots pst_{mct}), m < n$ defined as follows:

$$\exists i_1 \in 1..n: st_{i_1ct} = pst_{1ct} \wedge tt(st_{i_1ct}) \in PT \wedge \forall i_2 \in 1..(i_1 - 1): tt(st_{i_1ct}) \notin PT$$

$$\forall i \in 2..m: \exists j \in i..n: tt(st_{jct}) = pst_{ict} \wedge \exists k \in 1..(j-1): tt(st_{kct}) = pst_{i-1} \wedge \nexists o: k < o < j: tt(st_{oct}) \in PT. \quad (11)$$

If mapping of vector's $v = (v_1, v_1 \dots v_{n_v})$ arguments by a function f is characterised by $f_{arg}(v) = (f(v_1), f(v_2) \dots f(v_{n_v}))$, then st_{ct} conforms to the protocol p if the following condition is satisfied for its corresponding vector pst_{ct}: $(c_{1_{arg}}(pst_{ct}), c_{2_{arg}}(pst_{ct}) \dots c_{r_{arg}}(pst_{ct})) \in TCP_p$.

Different protocols may constrain different subtask properties. However, in our case, the minimal constraints are collaboration participant roles representing task owners and second party collaboration participants (for example from a send performative task point of view task owner is the sender and the second party is the receiver), and the corresponding legal performative task types. Let presume they are characterised as c_1, c_2 and c_3 respectively. Then $R_{c_1} = R_{c_2} = CPR_p$ is the set of all collaboration participant roles and $R_{c_3} \subseteq PT$.

A collaboration subtask vector can be mapped into vectors st_{cp_ict} comprising only tasks performed by individual collaboration participants, i.e.:

$$\forall i \in 1..|R_{c_1}|: st_{cp_ict} = (st_{i_1ct}, st_{i_2ct} \dots st_{i_{n'}ct}), \forall j \in 1..n': c_2(st_{i_jct}) = cp_i \wedge \nexists k \notin \{i_1, i_2 \dots i_{m'}\}: c_2(st_{kct}) = cp_i, cp_i \in R_{c_2}, l1 < l2 \rightarrow i_{l1} < i_{l2}. \quad (12)$$

Similarly TCP_p can be translated into:

$$TCP_p(cp_i) = \{(pr_{cp_i12} \dots pr_{cp_i1r}), (pr_{cp_i22} \dots pr_{cp_i2r}) \dots (pr_{cp_is2} \dots pr_{cp_isr})\}, \forall k \in 1..s, \forall j \in 1..r: pr_{cp_ikj} \subseteq R_{c_j}^{m'}, \quad (13)$$

where vectors pr_{cp_iij} comprise only constraints for performative tasks performed by cp_i. Therefore from cp_i's point of view a collaboration task conforms to the protocol p if:

$$(c_{1_{arg}}(pst_{cp_ict}), c_{2_{arg}}(pst_{cp_ict}) \dots c_{r_{arg}}(pst_{cp_ict})) \in TCP_p(cp_i). \quad (14)$$

Now a function $lt(ft) = lt_{ft}$, can be determined as follows:

$$\forall j \in 1..|lt_{ft}|: \exists z \in 1..s: \forall k \in 1..|ft|: \forall l \in 1..r: c_l(ft_k) = pr_{kcp_izl} \wedge lt_{jft} = (pr_{(k+1)cp_iz1}, pr_{(k+1)cp_iz2} \dots pr_{(k+1)cp_izr}), a \neq b \rightarrow lt_{aft} \neq lt_{bft}, \quad (15)$$

where $ft_k \in T$, $lt_{jft} \in T$ and $pr_{kcp_izl} \in R_{c_l}^{m'}$ are the k-th arguments of vectors ft, lt_{ft} and pr_{cp_izl} respectively. Thus for every individual participant a protocol determines a function lt which returns a vector of legal performative tasks they can perform based on a vector ft comprising performative tasks they have already finished during the collaboration.

The protocol library implements the minimal constraints and supports their extensions. In the protocol library the set PT containing performative task types is represented with the performative task type ontology. As it is used by all protocols this ontology is a common protocol base. Each protocol p has a corresponding ontology which determines protocol sets CPR_p and functions $lt(ft) = lt_{ft}$ for every collaboration participant. CPR_p is represented by a class for every collaboration participant role, while functions lt are implemented by SWRL rules (legal task rules). Besides the collaboration roles ontology and legal task rules every protocol has a name and a description. The description serves for the business process designer to understand protocols which are available allowing them to choose the appropriate one. Protocol name is used for specifying the protocol property of a Human task service's input.

This approach does not require a specific agent communication language (ACL) or a protocol specification to be used and can thus be implemented for any ACL, such as FIPA ACL [8] or KQML [7], and an arbitrary interaction protocol built upon them.

5. An example scenario

In this section, an example scenario from our case study is presented. In the scenario, the most appropriate transmission system engineer for a given elementary human task needs to be determined through collaboration based on the contract net protocol [8][1]. The protocol is required due to the nonworking hours specified in the timeframe when the task must be performed. It is performed between the transmission system operator and engineers.

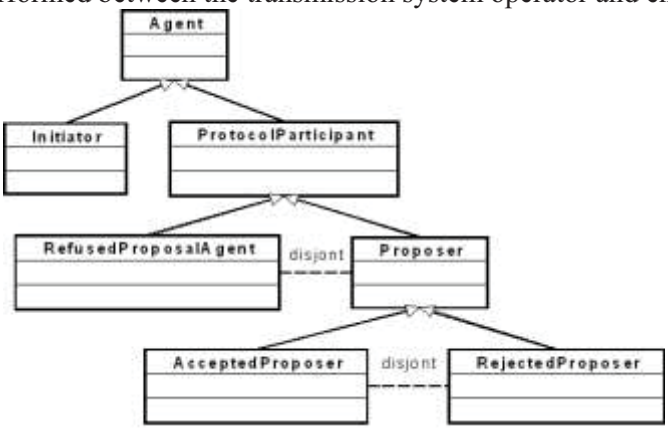

Figure 5: Contract Net Protocol roles ontology

Figure 5 illustrates contract net protocol roles ontology. The initial individual roles that the protocol translates to are the agent class child nodes, i.e. Initiator and ProtocolParticipant. Their subclasses are defined because their roles specialise during the collaboration task. Example of contract net protocol SWRL rules for the initiator role is:

```
Initiator(?x) ∧ InitiationAction(?y) ∧ isInState(?y,NotStarted) ∧
ProtocolParticipant(?z) → SendCFPMessageAction (?y) ∧ startAction(?x,?y) ∧
hasReceiver(?y,?z)
finishedAction(?x,?y) ∧ hasReceiver(?y,?z) ∧ SendCFPMessageAction(?y) ∧
ReceiveProposeMessageAction(?a) ∧ isInState(?a,NotStarted)→ startAction(?x,?a) ∧
hasSender(?a,?z) ∧hasDeadline(?a,"2007-05-15T09:00:00")
finishedAction(?x,?y) ∧ hasReceiver(?y,?z) ∧ SendCFPMessageAction(?y) ∧
ReceiveRefuseMessageAction(?a) ∧ isInState(?a,NotStarted) → startAction(?x,?a) ∧
hasSender(?a,?z)∧ hasDeadline(?a,"2007-05-15T09:00:00")
AcceptedProposer(?a) ∧ SendAcceptProposalMessageAction(?y) ∧ isInState(?y,NotStarted)
∧ hasReceiver(?y,?a) ∧ Initiator(?z) → startAction(?z,?y)
RejectedProposer(?x) ∧ SendRejectProposalMessageAction(?y) ∧ isInState(?y,NotStarted)
∧ hasReceiver(?y,?x) ∧ Initiator(?z) → startAction(?z,?y)
finishedAction(?x,?y) ∧ ReceiveProposeMessageAction(?y) ∧ hasSender(?y,?z) ∧
ReceiveRefuseMessageAction(?a) ∧ hasSender(?a, ?z) → cancelAction(?x,?a) ∧
Proposer(?z)
finishedAction(?x,?y) ∧ ReceiveRefuseMessageAction(?y) ∧ hasSender(?y,?z) ∧
ReceiveProposeMessageAction(?a) ∧ hasSender(?a,?z) → cancelAction(?x,?a)
```

It can be observed that the minimal constraints are extended with task deadline constraints. Properties specifying this task are given in Table 2.

Process Data Input	/	Number	1
Protocol	/	Protocol People Link	
Protocol Name	/	Role	Participant
Protocol People Link		Query	TransmissionSystemEngineer
Protocol Role	/	Protocol	/
Role People Link		Number	All

[1] Within the context of this paper the finishing messages (inform/failure) of the contract net protocol as defined in [8] are not a part of this collaboration task based on composite task definition in Section3. This would be implemented in another way, such as through a separate human task service call from a business process.

People Query	SystemX12Check TransmissionSystemEngineer(?x)	Task description	Perform system X12 unit check and report states during nonworking hours.
People Link Protocol		Task output query	X12SystemUnitState(?x) ∧ onDate(?x, "2011-05-26")
Protocol Name	Contract Net Protocol	Time Frame	From 2011-05-26T15:00:00 to 2011-05-27T00:00:00
Protocol People Link		Expected duration	1 to 3 hours
Role	Initiator	Priority	3
Query	TransmissionSystemOperator(?x)	Workflow Pattern	/
Protocol	/	Business Administr.	TransmissionSystemProcessesAdministrator(?x)

Table 2: Example input of a human task

When invoked with these properties the Human task service passes the task, without the business administrator property, to the Human task execution service. As the people resolution task is a collaboration task it itself comprises resolution subtasks needed to be performed for both protocol people links, i.e. finding a TransmissionSystemOperator employee for the Initiator role and TransmissionSystemEngineer employees for the ProtocolParticipant roles. After determining who should the perform contract net protocol roles the SystemCheckTransmissionSystemEngineer people query is transformed into a query looking for transmission system engineers available for 3 hours in the specified time frame. This query represents the output query property of the contract net protocol based people resolution task and is fairly simple for the purpose of this example. The human task agent representing the Initiator performs the first action as specified with the contract net protocol and sends calls for proposals to participant human task agents. Content of the message is the output query with an additional condition, stating that the SystemCheckTransmissionSystemEngineer is the individual to which this proposal is sent to. After receiving the call for proposal message every participant agent tries satisfy the query conditions. The query is satisfiable in their personal ontology if they are willing to perform the task and if they are available at the specified time. In that case the response contains an ontology satisfying the query, in which they state the time when they propose to perform the task. If it is not satisfiable a refusal is sent.

After receiving the proposals and refusals the Initiator must send acceptance and rejection messages and thus performs a message formation task, in which they add proposals to the ontology and reason upon it in order to determine whether the output query is now satisfiable, i.e. if the most appropriate transmission system engineer for the task is found. After obtaining the collaboration resolution task result the broker agent assigns a human task agent to perform the human task that they have been chosen for through the collaboration protocol.

In this way, the protocol-based collaboration is automated by the human task service. Due to different nature of the required work for message formation tasks, they can be either automated or not, however the part of actual composition of the collaboration part and performative acts are always automated by the system. If a protocol changes or if a new protocol is added, the protocol library can be updated, whereas the multi-agent system that implements the actual collaboration does not require any changes. The proposed solution provides a simple, but flexible and powerful mechanism for implementation of collaboration tasks in SOA systems.

6. Conclusion

In this paper a service-oriented architectural framework for automation and support of collaboration tasks in business processes was proposed. The main components of the framework that enable this are (1) - an organisational ontology comprising ontology-based decision models belonging to different organisational structure elements and a protocol

library, and (2) - a multi-agent system using the ontology. There are several important novelties and strengths of the approach. As collaboration is one of the essential activities in organizations, the framework provides support and automation for a large part of human tasks. However, due to responsibility issues, sometimes semi-automation with confirmation is preferred over complete automation of human tasks. The main advantage are semi-automated or completely automated collaboration tasks, which strive towards to the vision of end-to-end business process automation. Another important advantage is loose coupling between a collaboration protocol and its implementation. This allows for a greater flexibility, because it requires only to change the ontology based protocol definition and does not require any changes in the implementation of the system, which is an especially important advantage in the rapidly changing business environments.

References

[1] Aalst, W.M.P. van der; Hofstede, A.H.M. ter. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245-275, 2005.

[2] Ali, S; Soh, B; Torabi T. A Novel Approach Toward Integration of Rules Into Business Process Using An Agent-Oriented Framework. *IEEE Transactions on Industrial Informatics*, 2(3):145-154, 2006.

[3] Bermudez, J; Goni, A; Illarramendi, A; Bagues, M.I. Interoperation among agent-based information systems through a communication acts ontology. *Information Systems*, 32(8):1121-1144, 2007.

[4] Blake, M.B; Gomaa H. Agent-oriented compositional approaches to services-based cross-organizational workflow. *Decision Support Systems*, 40(1): 31-50, 2005.

[5] Bradshaw, J.M; Dutfield, S; Benoit, P; Woolley, J.D. KAoS: toward an industrial-strength open agent architecture. In: Bradshaw, J.M., editor. *Software Agents*. AAAI Press/The MIT Press, Cambridge, MA, 1997.

[6] Erl, T. *Service-Oriented Architecture: Concepts, Technology and Design*, Prentice Hall PTR, Upper Saddle River, NJ, 2005.

[7] Finin, T; Labrou, Y; Mayfield, J. KQML as an agent communication language, *Proc. of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages. 456-463, Gaithersburg, MD, USA, 1994.

[8] Foundation for Intelligent Physical Agents. Interaction Protocol Library Specification, 2003. Available on: http://www.fipa.org/repository/ips.php3.

[9] Giorgini P; Henderson-Sellers, B. Agent-Oriented Methodologies: An Introduction. In: Henderson-Sellers, B; Giorgini P., editors. *Agent-oriented Methodologies*, Idea Group Inc., Hershey, PA, 2005.

[10] Horrocks, I; McGuiness, D.L; Welty, C. Digital Libraries and Web-Based Information Systems. In: Baader, F; McGuiness, D.L; Nardi D., editors. Description *Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, Cambridge, UK, 2003.

[11] Jammes, F; Smit, H. Service-Oriented Paradigms in Industrial Automation. *IEEE Transactions on Industrial Informatics*, 1(1):62-70, 2005.

[12] Jennings, N.R; Norman, T.J; Faratin, P. Autonomous Agents for Business Process Management. *International Journal of Applied Artificial Intelligence*, 14(2):145-189, 2000.

[13] Kalogeras, A.P; Gialelis, J.V; Alexakos, C.E; Georgoudakis, M.J; Koubias, S.A. Vertical Integration of Enterprise Industrial Systems Utilizing Web Services. *IEEE Transactions on Industrial Informatics*, 2(2):120-128, 2006.

[14] Lai, L. F. A knowledge engineering approach to knowledge management. *Information Sciences*, 177(19): 4072-4094, 2007.

[15] Leymann, F. Web Service Flow Language (WSFL 1.0), IBM, 2001.

[16] Li, Y; Chao, K-M; Younas, M; Huang, Y; Lu, X. Modeling e-marketplaces with multi-agents Web services. *Proc. 11th Int. Conf. on Parallel and Distributed Systems,* pages 175-181, *Fukuoka*, Japan, 2005.

[17] OASIS. Web Services – Human Task (WS-HumanTask) Specification Version 1.1, Committee Specification 01, 2010. Available on: http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html.

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

[19] OASIS. WS-BPEL Extension for People (BPEL4People) Specification Version 1.1, Committee Specification, 2010. Available on: http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html.

[20] Šaša, A; Jurič, M.B; Krisper, M. Service-oriented framework for human task support and automation. *IEEE transactions on industrial informatics*, 4(4):292-302, 2008.

[21] Singh, M.P. A social semantics for agent communication languages. *Issues in Agent Communication*, Springer, Berlin, 2000.

[22] T.I. Zhang, H. Jiang, "A Framework of Incorporating Software Agents into SOA", Proc. Artificial Intelligence and Soft Computing (ASC 2005), Benidorm, Spain, 2005.

[23] Taveter, K; Wagner, G. Towards Radical Agent-Oriented Software Engineering Process Based on AOR Modelling. In: Henderson-Sellers, B; Giorgini, P., editors. *Agent-oriented Methodologies*, Idea Group Inc., Hershey, PA, 2005.

[24] Thatte, S. XLANG: Web Services for Business Process Design, Microsoft Corporation, 2001.

[25] W3C. OWL 2 Web Ontology Language", Document Overview, 2009. Available on: http://www.w3.org/TR/owl2-overview/.

[26] W3C. SWRL: A Semantic Web Rule Language, Combining OWL and RuleML, W3C Member Submission, 2004. Available on:http://www.w3.org/Submission/SWRL/.

[27] Wooldridge M. *An Introduction to MultiAgent Systems*, John Wiley & Sons Ltd, Chichester, UK, 2002.

[28] Xu, Q; Qiu, R; Xu, F. Agent-Based Workflow Coordination for Supply Chain Management. *Transactions of Nanjing University of Aeronautics & Astronautics*, 20(1):112-117, 2003.

[29] Yolum, P; Singh, M.P. Commitment-based enhancement of e-commerce protocols. *Proc. IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 278–283, Gaithersburg, MD, USA, 2000.

[30] Zurawski, R. Integration Technologies for Indstrial Automated Systems: Challenges and Trends. In: R. Zurawski, editor. *Integration Technologies for Industrial Automated Systems* (Industrial Information Technology), CRC Press, Boca Raton, FL, Jul. 2006.