Event-driven adaptive collaboration using semantically-enriched patterns

Nikos Papageorgiou a, Yiannis Verginadis a, Dimitris Apostolou b, Gregoris Mentzas a,b,*

a Institute of Communications and Computer Systems, National Technical University of Athens, 9, Iiron Polytechniou Str., Athens, Greece
b Informatics Department University of Piraeus, 80, Karosi & Dimitriou Str., Piraeus, Athens, Greece

ARTICLE INFO

Keywords:
Collaboration patterns
Semantic web
Ontology
Virtual organizations

ABSTRACT

Collaboration is essential for value creation in the modern business environment, may span across organizational and geographical boundaries and is often used for mission critical tasks. Collaborative environments are subject to continuous changes because participation is dynamic and business goals may be evolving. In such dynamic environments there is a need for adapting the ways of collaboration to reflect the current conditions. By focusing on collaboration in dynamic environments, we explore the utilization of collaboration patterns as models for recurring high-value collaborative tasks, which can be intelligently identified, retrieved and enacted when needed. In this paper, we propose Collaboration Pattern Assistant, an information system which is built around the concept of collaboration patterns and is based on an innovative coupling of ontologies with an event driven architecture. The advantages of using ontologies to represent collaboration patterns lie in their ability to model effectively related concepts and interrelations, to map to collaboration services provided by different suppliers, to use reasoning to check values for validity and consistency, as well as to derive new facts based on the existing ones. The adoption of an event driven architecture enables collaboration support which can respond to continuously changing circumstances by processing effectively and reacting to events generated by on-going collaborations.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Collaboration is a recursive process where two or more people or organizations work together in an intersection of common goals – for example, an intellectual endeavor1,2 that is creative in nature3 – by sharing knowledge, learning and building consensus. Collaboration is essential for value creation in the modern business environment, may span across organizational and geographical boundaries and is often used for mission critical tasks (Hlupic & Qureshi, 2002, 2003). Examples of collaborative tasks include strategic planning, engineering design and problem solving. While collaboration can be creative and can lead to new value offerings, it is also fraught with challenges that can lead to unproductive processes and failed efforts (Dennis & Wixom, 2002; Nunamaker, Dennis, Valacich, Vogel, & George, 1993).

Over the last two decades, many organizations and individuals have relied on electronic collaboration between distributed teams to achieve higher productivity and produce joint products. Technology has evolved from standalone tools, to open systems supporting collaboration in both intra- (Dennis, George, Jessup, Nunamaker, & Vogel, 1988) and inter-organizational settings (Malone, Yates, & Benjamain, 1987; Roth, 1996; Srinivasan, Kekre, & Mukhopadhyay, 1994), and from general purpose platforms to specialized collaboration tools. Today’s computer-assisted collaboration encompass a broad range of tools that enable groups of people to work together including social software such as wikis and blogs, instant messaging, team spaces, web sharing, audio and video conferencing. Various languages for modeling collaborative processes, including BPEL4People, have been proposed in an effort to provide computer support to reliably repeatable sequences of collaborative processes. However, even for carefully planned processes with human participation, ad-hoc adaptation and intervention is required due to the complexity of human tasks, people’s individual understanding and unpredictable events. This limits the applicability of existing tools in virtual collaboration. Collaborative environments are subject to continuous changes because participation is dynamic and business goals may be evolving. In such dynamic environments there is a need for adapting the ways of collaboration to reflect the current conditions.

By focusing on collaboration in dynamic environments, we explore the utilization of collaboration patterns (CPats) as models for recurring high-value collaborative tasks (Barros, Dumas, & ter Hofstede, 2005), which can be intelligently identified, retrieved and enacted when needed. CPats can be a means to capture best
practices about solutions to recurring collaborative problems. In previous work we have introduced CPats as a way to balance flexibility and reusability in mixed human and software-enabled collaborative environments (Verginadis, Apostolou, Papageorgiou, & Mentzas, 2009a, 2009b).

The research question we investigate in this paper is how to take advantage of the concept of collaboration patterns by means of a collaborative information system that exploits the conceptual advantages of patterns and addresses the need for adapting collaboration support to reflect changing conditions. We propose Collaboration Pattern Assistant (CPA), a collaborative information system which is build around the concept of CPats and is based on an innovative coupling of semantic technologies with an event driven architecture to enable collaboration support which can respond to continuously changing circumstances. We demonstrate that CPA can facilitate collaborations in real-life, dynamic collaborative environments.

The remaining of this paper is organized as follows: Section 2 describes the theoretical underpinnings of our work which is built on the concept of patterns. Section 3 presents our proposed CPat model, its structure and schematic representation while Section 4 outlines the use of ontologies for formally representing CPats. Section 5 describes CPA, the corresponding software implementation. Section 6 illustrates how CPA can support an example collaboration scenario. Section 7 presents the system evaluation results while Section 8 discusses related work and our contribution in the field. We conclude with Section 9, highlighting our plans for future work.

2. Patterns

2.1. Concept of patterns

The concept of patterns, inspired by the way experts tackle work on a particular problem without re-inventing the wheel, has been studied first in the field of engineering by Christopher Alexander, a professor of Architecture at University of California, Berkley. His book, where he describes a language for architectural patterns (Alexander et al., 1977), is seen as the prototype for patterns in many other domains. During this first use of patterns in architecture, he defined a pattern as a “morphological law that explains how to design an artefact in order to solve a problem in a specific context”. In computer science and software engineering, design patterns have been adopted in order to capture best practices for creating software and using common concepts to solve recurring problems (Gamma, Helm, Johnson, & Vlissides, 1995). The idea of patterns has also been introduced in the field of Human Computer Interaction. Tidwell (2005) has extended the use of patterns to describe interfaces for desktop applications, web applications, web sites and mobile devices. Moreover, the application of patterns in other domains, such as process, workflow and event-based systems has been studied.

2.2. Collaboration patterns

During business collaborations, recurring segments of work can be identified and introduced as CPats for future use. The vision behind CPats is that the solution can be described as an encapsulated component that can be reused whenever a situation occurs (Schümmmer, 2002). CPats can be regarded as prescriptive means for modeling collaborative tasks and protocols of cooperation (Molina & Bell, 1999), while they can guide the configuration of collaborative working environments to meet the requirements of the participants (eAce, 2005; Slagter, Biemans, & Jones, 2005).

A CPat determines the basic structure of the solution to a particular collaborative problem. It does not always specify a fully detailed solution, however. It provides a scheme for a generic solution to a family of problems, rather than a prefabricated module that can be used ‘as is’. The implementation of this scheme must be done according to the specific needs of the situation at hand. Moreover, CPats are not only useful to users within the same domain as the pattern, but frequently a pattern is useful in other domains as well.

Towards the direction of having a common understanding about what a CPat is, including at the same time the main elements of the, so far, expressed definitions of a pattern (Anderson, 2000; Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996; Dyson & Longshaw, 2004; Schümmer & Lukosch, 2007), we propose the following definition:

“A collaboration pattern is a prescription which addresses a collaborative problem that may occur repeatedly in the environment. It describes the forms of collaboration and the proven solutions to a collaboration problem and appears as a recurring group of actions that enable efficiency in both the communication and the implementation of a successful solution. The collaboration pattern can be used as is in the same application domain or it can be abstracted and used as a primitive building block beyond its original domain.”

3. A model for collaboration patterns

3.1. Overview of collaboration pattern model

For modeling CPats following the definition introduced in Section 2, we need attributes to represent the situation and its solution that the pattern can address in a specific context and under specific pre-conditions that must be satisfied. Moreover, a CPat model should encapsulate key findings of existing research in patterns (Verginadis, Papageorgiou, Apostolou, & Mentzas, 2010), such as: (a) CPats should serve different collaboration goals and can be of different levels of granularity; (b) CPats should trigger human-based and machine-enabled processes when certain events occur and conditions hold (de Moor, 2006); (c) patterns may prescribe a structure (in the form of a series of steps and the applicable user roles), resources (describing the activities to be done and content to be used), and methods for accessing IT resources to get things done (Henninger & Ashokkumar, 2006); (d) CPats should contain a diagrammatic description of the proposed solution.

As a starting point, a CPat needs a certain Name as an identifier, a Problem that describes the issue the CPat has addressed before or it is expected to address in the future, a Context that describes the space in which the CPat can be embedded and be applicable, Pre-conditions, that dictate the states and conditions that must be satisfied before the CPat can be considered applicable and a Category that assigns a granularity level to it.

Regarding CPat categories, we propose a three-level categorization of CPats based on the collaboration goal or function they support. Specifically: (i) the Strategic CPat category contains CPats that facilitate strategic goals which may refer to, e.g., group formation and structure, the addition or removal of a partner, actions that determine the group goals, etc., (ii) the Business CPat category contains CPats that serve operational functions of the collaborating groups (e.g., proposal preparation, conflict resolution, scheduling a meeting, etc.), and (iii) the Simple CPat category which includes CPats that focus on elementary collaborative activities (e.g., completing a task) and are expected to propose a solution that involves a small number of simple activities. The proposed category hierarchy can be extended to support additional levels of abstraction.

An important attribute of a CPat description is the specification of the trigger for its execution. In an event-driven approach, simple
environments. Another attribute of a CPat description is the need for reactivity and attention amplification in collaborative or complex events can trigger the execution of a CPat satisfying the Collaboration pattern model structure.

Table 1 compiles the CPat attributes described in Section 3.1 into a comprehensive structure which specifies what a CPat does, where it is applicable and under which circumstances it may be used. Table 1 contains also the structure of “Schedule a meeting”, an example CPat, which takes place in the context of a Virtual Organization (VO) consisting of pharmaceutical companies collaborating in order to develop a drug according to a new set of drug administration regulations. In this scenario, a meeting needs to be scheduled either periodically or upon the delay of a specific deliverable. To schedule a meeting, at least three collaborators should be available along with the coordinator and the necessary budget. When the aforementioned pre-conditions and triggers are satisfied, the specific CPat is applied and the solution it prescribes, in terms of actions to be taken and tools to be used, can be followed. Specifically, the work-package leader should be notified in order to send an e-mail to the VO coordinator asking him or her to notify the VO members about the need of organizing a meeting. Further, a web tool for agreeing on the meeting date is identified and possibly automatically configured (e.g., with partner names and possible meeting dates) and invoked. When the meeting takes place, the meeting minutes are produced. In case the delayed deliverable that triggered this meeting is submitted by the responsible partner before the actual meeting takes place, an exception event is triggered and the CPat is stopped; moreover, a CPat for postponing the meeting is invoked.

3.2. Collaboration patterns structure

Table 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name and No.</td>
<td>A name and a number for quick referencing</td>
<td>&lt;Schedule a Meeting&gt; - CPat 1</td>
</tr>
<tr>
<td>Category</td>
<td>Positioning in one of the proposed categories (strategic/business/simple)</td>
<td>Business Pattern</td>
</tr>
<tr>
<td>Problem</td>
<td>A description of the problem(s) the CPat has addressed before or it is expected to address in the future</td>
<td>Check the work progress for a specific deliverable</td>
</tr>
<tr>
<td>Group lifecycle phase</td>
<td>One or more group lifecycle phases where it can be applicable (pre-creation, creation, operation or termination)</td>
<td>VO Operation</td>
</tr>
<tr>
<td>Application area</td>
<td>Declares the sector (e.g., Manufacturing) where it is applicable</td>
<td>Pharmaceutical Industry</td>
</tr>
<tr>
<td>Pre-Conditions</td>
<td>The list of the states and conditions that must be satisfied before the specific CPat can be considered applicable</td>
<td>(No. of collaborators &gt; 3) AND (VO coordinator available) AND (Budget available)</td>
</tr>
<tr>
<td>Triggers:</td>
<td>Events and event patterns that can trigger the execution of the CPat</td>
<td>(The last progress meeting was held 3 months ago) OR (Deliverable Derma1 delayed)</td>
</tr>
<tr>
<td>Triggers of exceptions</td>
<td>Events that can raise an exception during the implementation of CPat</td>
<td>Deliverable Derma1 has just been sent</td>
</tr>
<tr>
<td>Roles:</td>
<td>Includes the collaboration roles that are to be involved in the CPat</td>
<td>VO Coordinator, WP leader, VO member</td>
</tr>
<tr>
<td>Input information</td>
<td>Documents or data that will be used in terms of this CPat</td>
<td>Project’s DoW, VO members’ contact details</td>
</tr>
<tr>
<td>Output information</td>
<td>Documents or data that will be produced in terms of this CPat</td>
<td>Meeting minutes document</td>
</tr>
<tr>
<td>Duration</td>
<td>The acceptable time frame in which the proposed by the CPat solution can be successfully implemented</td>
<td>2 Weeks (this can be a variable)</td>
</tr>
<tr>
<td>Exception</td>
<td>A description of an exception to the pattern (e.g., termination of the specific CPat and execution of another one)</td>
<td>&lt;Postpone Scheduled Meeting CPat&gt;</td>
</tr>
<tr>
<td>Post-Conditions</td>
<td>Conditions and states that hold after the successful termination of the CPat</td>
<td>Meeting took place AND Agreed minutes stored in system</td>
</tr>
<tr>
<td>Related CPats</td>
<td>– CPats that optionally can be executed in parallel or after its termination</td>
<td>– CPat that optionally: &lt;Schedule conference call CPat&gt;</td>
</tr>
<tr>
<td></td>
<td>– Alternative CPats that can be used instead of the described one</td>
<td>– CPat that can be executed subsequently: &lt;Postpone Scheduled Meeting&gt;</td>
</tr>
<tr>
<td></td>
<td>– Conflicting CPats that cannot be executed concurrently with the described one</td>
<td>– Alternative CPat: &lt;Schedule an Online Meeting CPat&gt;</td>
</tr>
<tr>
<td>Solution</td>
<td>Comprises prescriptions of solutions to the designated problem in the form of action lists, workflows or even instructions for tool usage</td>
<td>Usage of tools: – Engage with <a href="http://www.doodle.ch">www.doodle.ch</a> to find date based on collaborators’ availabilities</td>
</tr>
</tbody>
</table>

3.3. Collaboration patterns diagrammatic description

Fig. 1 depicts the diagrammatic description of the “Schedule a Meeting” CPat following the BPMN4 notation. BPMN is typically used for business process modeling and contains symbols that are able to visualize the collaborative process of the CPat solution, CPat triggers, pre and post-conditions, exceptions, input and output information. The capability of BPMN to model abstract processes is also very important because it can be used in cases where the solution of a CPat should have the form of a loose action list and thus allow the CPat designer to decide about the rigidity of the solution that a specific CPat proposes.

4. Ontology based representation of collaboration patterns

4.1. The Companion collaboration pattern ontology

The aim of Companion (Papageorgiou, Verginadis, Apostolou, & Mentzas, 2010), the proposed ontology for representing CPats is to: (i) provide a formal representation of the CPat model concepts and interrelation; (ii) capture the requirements of prominent collaborative processes, (iii) allow for mapping to collaboration services provided by different suppliers and (iv) provide a flexible structure that can be easily refined, updated, extended andinstantiated. As stated by Henninger and Ashokkumar (2006), a
major weakness of most existing pattern representations is the lack of semantics, i.e. typed relationships between pattern attributes. While other formal media such as UML can be used to model and represent patterns none has the combination of both formal representation and distributed accessibility that ontologies provide. Following Henninger’s argumentation about the benefits of ontologies for the formal representation of usability design patterns (Henninger & Ashokkumar, 2006), mapped certainly to the requirements of collaboration patterns, ontologies provide a computational medium that can: (a) intelligently match collaboration contexts and collaborative requirements to collaboration patterns, (b) make intelligent inferences about applying patterns to solve problems at successive levels of abstraction and (c) check the consistency of patterns. These capabilities are provided as a result of the conjunction of our ontology-based paradigm for the design and instantiation of CPats, which we will describe in the following sections, with the general capabilities of an OWL reasoner in TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, conjunctive query answering).

Companion is developed in OWL Description Logic (OWL-DL), a highly expressive yet computable ontology language. For the development of the CPat ontology we have used the Protégé 3 (Stanford, 2008) ontology editor with OntoViz (Sintek, 2003) plug-in for visualization and the Pellet reasoner (Sirin, Parsia, Grau, Kalyanpur, & Katz, 2007) for validation.

Figs. 1 and 2 depict the pattern concept along with its object properties. Starting from the statement that a Collaboration Pattern is subclass of a Pattern, as described in CPat model, we state that a CPat has Pre-Conditions, Post-Conditions, category (CPatCategory), Application Area, Triggers which are Complex Events and related CPat(s). These relationships are depicted in Fig. 2.

Fig. 3 shows that a CPat has Input and Output information (CPatInformation), problem(s) (CPatProblem), corresponds to Group lifecycle phase (GroupPhase), has Participants (CPatParticipant), has exception(s) (CPatException), solution(s) (CPatSolution), hasInvitation (CPatInvitation) and hasRecommendation (CPatRecommendation).

Beyond object properties, the class CollaborationPattern has some data properties (CPatDuration, CPatName, CPatNo). All properties are declared as functional, i.e. they can have only one, unique value y for each instance x. CPats may be related with other CPats. The object property hasRelatedCPat describes those relationships. Special cases of pattern relationships such as canBeExecutedInSequence, hasAlternativeCPat, hasConflictCPat and canBeExecutedInParallel are modeled with sub-properties.

Fig. 4 depicts the restrictions of class CollaborationPattern. Every instance of the CPat class:

- is subclass of Pattern and belongs to one of StrategicPattern, BusinessPattern or SimplePattern classes,
- has at least one Access Right, Application Area,
- has exactly one CPatCategory,
- has at least one Participant who is the CPat Initiator,
- has two or more Participants that undertake specific roles in terms of CPat’s solution implementation,
- has at least one CPatProblem, CPatSolution, ComplexEvent and GroupPhase.
Companion continues with the description of other related CPat concepts. Pre-Conditions and Post-Conditions are of type Condition. A Condition is composed of one or more expressions which evaluate one or more facts. Facts are the elements of a knowledge-base that contain information about the state of collaboration. CPat categories are represented as instances of the CPatCategory class. CPats are also associated with application areas for which we use the North American Industry Classification System (NAICS, 2007). Every CPat and CPat exception can be triggered by a ComplexEvent. A CPat exception involves at least one of the following: the proposed usage of a Collaboration Tool, the execution of an Action List, the execution of another CPat, the execution of a Workflow or a combination of them.

In our approach, every CPat is represented as a subclass of the CollaborationPattern class and accordingly every instance of a specific CPat is represented as an instance of the corresponding OWL class. For example in a situation where the collaboration aims to plan, execute and evaluate a kick-off meeting of 'Project1', we could create an instance of the 'KickoffMeetingCPat' (owl class) with name 'Project1_KickOff_Meeting' (individual). CPat classes can be created with any generic ontology editor or by using a custom editor created for this purpose. CPats, in order to express specific concepts found in CPat attributes, have the possibility to import the corresponding domain ontologies, e.g., Medical, Geographical, or Virtual Organizations ontologies (Fig. 5). In such a way CPats and the tools that process them are not limited to the terminology of Companion but they are free to use any domain ontology in an open and transparent manner.

CPats attributes (e.g., Problem, Phase, PreCondition, Trigger,) are described in detail with the CPat model (Section 3). There are many ways to convert them in OWL constructs, but if we want to maintain the CPat ontology’s computability (the ability to perform ontology classification and realizations within a reasonable time) and efficiency, we must restrict CPat ontology to the OWL-DL dialect. Since CPats are OWL classes this choice sets some restrictions on the kind of relationships that we can have among CPats and other OWL constructs such as individuals, properties, etc. A solution to this problem is to use OWL property restrictions in order to relate the CPat classes to their attributes, when attributes take values in a dynamic manner (such as PreConditions, Triggers and Participants) and annotation properties for the simple static CPat attributes (e.g. CPats creation date, number, name, etc.). An example of the KickOffMeeting CPat, if we use the TURTLE serialization syntax for RDF/OWL is shown in Fig. 6.

In a similar way we describe the rest of the CPat into OWL-DL and allow the CPat Assistant to read it and generate CPat logic, rules and actions. In the following section we will describe the CPat triggering and execution logic.

4.2. Reasoning with Companion

The Companion ontology along with other imported domain-specific ontologies are used in order to provide semantics-based recommendation and decision facilities during CPat execution. CPat elements such as preconditions, triggers, participants, input information, tools and solutions are inferred and recommended in runtime based on the subsumption and instance checking capabilities of OWL-DL. The most common inferences that are expected to be performed with Companion are the class/subclass (e.g., subclasses used in existential quantification, subproperty assertions, etc.) and class/individual inferences (e.g., interactions between complete and partial definitions or between an inverse relationship and domain/range constraints on a property, etc.).

We give an indicative reasoning example using the extension of Companion shown in Fig. 7. With this ontology we define that: (i) the class NotRepliedEmail is subclass of class CEvent (represents complex events), (ii) it has two object properties (”to” and “from”) denoting accordingly the sender and the recipient of an email and a datatype property named “timeout” that denotes the duration elapsed before the NotRepliedEmail complex event is generated, and (iii) the ontology contains a class ProjectManager that is subclass of VOMember. Using these concepts we can declare a CPat trigger named ProjectManagementProblem with OWL axioms (Fig. 8).

This definition enables an OWL reasoner to infer that the NotRepliedEmail events that have sender (pointed by the

http://www.w3.org/TeamSubmission/turtle/.

property “from”) a ProjectManager belong to the class ProjectManagementProblem.

By inserting the triples shown in Fig. 9 we define two NoteRepliedEmail instances (NotRepliedEmail_16,17) that come from different senders. NotRepliedEmail_17 comes from an individual that belongs to the class ProjectManager (because ProjectManager_15 has rdf:type ProjectManager) but NotRepliedEmail_16 comes from a sender that is not a ProjectManager. As a consequence we expect that only NotRepliedEmail_17 will be inferred as a ProjectManagementProblem. A test with Protege and Pellet
gives indeed the expected results (Fig. 10). Similarly, reasoning with Companion can enable CPat classification, e.g., the CPat_Kick-OffMeeting is subclass of CPat_ProjectMeeting because the preconditions, triggers, etc., of the first are subclasses of corresponding properties of the second.

4.3. Responding to changing circumstances

Collaborations are subject to changes because participation is dynamic and business goals may be evolving. In this section we describe how CPats can be used as a mechanism to enable collaboration adaptation in order to reflect the current collaboration circumstances. To this end, we explore the capabilities of event-driven systems to react to changing conditions of the environment as these are signaled by generated events.

In our approach, CPats are recommended automatically upon the arrival of complex events, if relevant conditions hold. CPat conditions and triggers are represented using OWL-DL classes. Classes in OWL-DL can have instances, either set explicitly (asserted) or inferred by their OWL-DL property descriptions. The main idea behind the evaluation of CPat conditions and triggers is that, when the corresponding classes that define the conditions or the triggers have instances, the condition is true or the trigger exists. The evaluation of class instances is based on the execution of an OWL reasoner. The reasoner reads the CPat OWL classes and their property restrictions, evaluates new statements (triples) upon their insertion into the knowledge base and produces inferred triples by translating OWL semantics to rules. By implementing a mechanism that inserts new collaboration knowledge into a CPat knowledge base in the form of OWL statements and combining it with OWL CPats we enable the automatic execution of CPat recommendations and corresponding solutions. The pseudo-code in Table 2 illustrates a CPat triggering logic.

The starting point for recommending a CPat is the arrival of a complex event. The event is written to the knowledge base in order to be processed by the reasoner and all CPat triggers are examined. For each CPat whose trigger (i.e. the class that is related to the CPat with the hasTrigger property) has some instance, the corresponding CPat PreCondition class is examined. If the pre-condition is true (has at least one instance) then a CPat recommendation is generated (Table 2 – step 1). The CPat Recommendation is presented to all candidate CPat initiators (Fig. 11, see also Section 5).

CPat initiators are discovered by means of retrieving the instances of the class that is related to the specific CPat with the property hasInititator (Table 2 – step 2). CPat initiators have the option to accept or reject a CPat. If a CPat initiator decides to accept a recommended CPat, the CPat Assistant presents a GUI that enables the user to configure the CPat in terms of candidate Participants, Input Information, and Solution. All invitation and recommendation concepts are expressed with Companion (Fig. 12).

Similarly to CPat initiators, CPat participants and input information are discovered by retrieving from the knowledge base the instances of the classes that are related to the CPat class with corresponding instances. CPats may be linked to one or more roles. Participants are related to the CPat with properties named "role<role_name>" (e.g., "roleMeetingOrganizer") according to their desired role. The property hasInputInformation points to the class that describes a CPat candidate input information. Input information, documents or other data in the knowledge base, are referenced by using a unique URI (e.g., the url of an html document). This URI is an OWL/RDF individual, too. This implies...
that it can be associated with other individuals or literals with OWL properties according to its content. CPat participants may be assigned to a CPat with a specific role either upon accepting an invitation or upon direct assignment by the CPat initiator.

In step 3 of Table 2, we define that, when the CPat initiator that accepted the recommendation chooses to initiate the execution of the CPat solution, the system validates that the created CPat instance is complete (in terms of required participants, input information and solution configuration). As a CPat instance is being created all information is stored in the CPat knowledge base in the form of statements using the corresponding properties (hasInputInformation, role<?role_name>, etc.) (Fig. 13).

In step 4, during the execution of the solution and upon the completion of each step the post condition class of the CPat is examined (Table 2 – step 4). If it has instances, CPat participants are informed that the CPat goals have been reached and the CPat is terminated.

Finally (Table 2 – step 5), for each CPat instance that is currently active and whenever a new event is written to the knowledge base, the corresponding CPat Exception Trigger class is being examined for instances. If it is true (because there are some individuals belonging to the relevant class) then the system recommends the termination of the CPat and, in case that an exception CPat has been defined, proposes to start a new CPat in order to handle the exception.

5. Collaboration Pattern Assistant

Collaboration Patterns Assistant (CPA) is a software system that facilitates collaboration by providing functionalities for supporting

<table>
<thead>
<tr>
<th>Table 2</th>
<th>CPA logic.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WHEN &lt;Trigger&gt; IF &lt;PreCondition&gt; THEN &lt;Recommend CPat&gt;</td>
</tr>
<tr>
<td>2</td>
<td>IF &lt;CPat Recommendation Accepted&gt; THEN &lt;Begin CPat Configuration (by the CPat initiator)&gt;</td>
</tr>
<tr>
<td>3</td>
<td>IF &lt;CPat Recommendation Accepted&gt; AND &lt;CPat Configuration Completed&gt; THEN &lt;Execute CPat Solution&gt;</td>
</tr>
<tr>
<td>4</td>
<td>IF &lt;CPat Solution Executed&gt; AND &lt;PostCondition True&gt; THEN &lt;Terminate CPat&gt;</td>
</tr>
<tr>
<td>5</td>
<td>WHEN &lt;Exception Trigger&gt; IF &lt;CPat Solution Executed&gt; THEN &lt;Terminate CPat&gt; AND &lt;Recommend Alternative CPat&gt;</td>
</tr>
</tbody>
</table>

Fig. 11. CPat recommendation.

Fig. 12. CPat recommendations and invitations.
the creation, recommendation, instantiation and execution of CPats. The system employs an Event-Driven Architecture (EDA) which allows CPats to be recommended automatically upon the arrival of events, if the context is suitable as described below.

5.1. Functionalities

CPA functionalities can be distinguished in those that are used when designing CPat-based collaborations and those used when executing collaborations. Design time functionalities involve the creation, editing and validation of CPats. The CPA can model recurring segments of collaborations as CPats. The functional tool that facilitates this work is the Collaboration Patterns Editor (CPE) that is responsible for defining, editing, searching and simulating collaboration patterns. It produces abstract CPats which can be instantiated and adapted during run-time. In run time, CPA provides functionalities for helping users collaborate according to the selected or recommended patterns. CPA involves the following run-time functionalities:

- **Recommender service**, which recommends CPats and CPat solution elements (actions, workflows, tools). It has as input the collaboration state that is composed by what is currently being executed and what just happened; what events or complex events were detected. It has as output the recommendation for initiating a new CPat (in parallel, sequential, or by terminating the old one). For example, when a VO participant is not active for a given period, it recommends mitigating CPats such as scheduling a project management meeting to address the issue.

- **Awareness service**, which provides information about the state of collaborators and the state of on-going collaborative work. It takes input from the knowledge base regarding the several aspects of on-going collaborations and provides as output new or updated documents involved in each CPat as well as notifications about the possible next steps as specified in the CPat. Moreover, it reports about people in charge of tasks, or updated documents involved in each CPat as well as notifications about the possible next steps as specified in the CPat. For example, it can report that “all collaborators have downloaded the proposal”.

CPA distinguishes two user roles: (i) CPat initiators and (ii) CPat participants. CPat initiators are responsible for the initiation and the termination of a new collaboration according to the selected CPat. CPat participants use the CPA in order to get information about the current activities inside their group, to participate in collaborations performed within a planned CPat and to get informed about their assigned tasks. CPat initiators may also be CPat participants. The candidate CPat initiators are determined by the system according to the specification of each CPat. CPats are stored in OWL in a dedicated CPat knowledge base. The initiator of a CPat decides the form of collaboration by choosing a loose schema (i.e. adopting an action list) or triggering the execution of a strict workflow.

5.2. CPA architecture and implementation

The architecture of CPA is shown in Fig. 14. The CPA user interface assists users in collaborating by e.g., providing information about their assigned tasks and tools to use. CPat specifications are inserted into the system by a dedicated CPat Editor. CPats are stored in OWL in the CPat knowledge base. Knowledge about the ongoing collaborations is maintained also in knowledge bases. Information about events that occur during collaboration arrives through the Complex Event Processing component and its associated event patterns store. The CPat Logic component implements the CPat triggering and execution logic described in Table 1 using an OWL reasoner and drives the execution of the CPat solution (workflow or action list). In order to provide more fine-grained results, besides the OWL property restrictions which are being executed automatically by the reasoner, we have implemented a SPARQL filter evaluation mechanism. The classes which define CPat conditions, triggers, or recommenders may be associated also with SPARQL filters. The system automatically adds the filter to the instance retrieval SPARQL queries.

The CPA prototype was implemented using the Adobe Flex/AIR framework\(^7\) for the client and the open source Flash server Red5\(^8\) (Fig. 15). This combination gives us the ability to build on an open source platform which provides facilities very useful in the development of collaborative applications. The communication between the server and the client is done using the RTMP/AMF3 protocols. The use of these protocols allows real-time bidirectional communication between the clients and the server in order to transfer data, commands, events or even streaming audio/video.

The Red5 server is written in Java and runs inside the Tomcat servlet container. The fact that it is written in Java allows easy integration with the existing vast collection of open-source tools and services. All the business logic of CPats is written in Java in the form of a servlet that plugs in the Red5 RTMP servlet. In order to store knowledge we use the Sesame triple-store with the SWF OWLIM plugin for OWL. This combination is very efficient and allows both automatic OWL reasoning (or else ontology materialization) and querying through SPARQL. The incoming events are going to be processed by the Esper\(^9\) Complex Event Processing (CEP) engine. This CEP Engine has the capability to process simple events (e.g., coming from groupware applications), combine them and produce complex constructs of events that may trigger CPats. Workflow execution is managed by Intalio,\(^10\) an open source workflow management system which executes workflows written in BPEL, provides human task management and BPEL4People services and a BPMN Editor.

6. Illustrative scenario

In this section we present an illustrative scenario that involves the process for carrying out a collaborative drug development project, involving a VO structure of collaborating partners with expertise in the design, identification and experimental confirmation of designing and developing drug candidates. The pharmaceutical industry is considered a typical example of knowledge-intensive sector where the problem of dealing with heterogeneous and vast number of information appears to be insurmountable. The VO of our scenario follows the Star Alliance VO topology which involves

---

\(^7\) http://labs.adobe.com/technologies/flex/
\(^8\) http://www.red5server.org/
\(^9\) http://esper.codehaus.org/
\(^10\) http://www.intalio.com/bpms/workflow
the grouping of independent organizations, with a core organization taking the lead management role (Lethbridge, 2001). The VO comprises four pharmaceutical companies with expertise in drug design and development. The common goal for this collaboration has been agreed to be the development of drugs according to the regulations and ethics taking into account the profit maximization.

According to Investigational New Drug Application Process (IND, 2010), the process of developing a new drug involves several different stages starting from pre-clinic studies (testing the drug in the lab, use it on guinea pigs, etc.) and continuing with the four phases imposed by Foods and Drug Administration (FDA, 2011) and the European Medicines Agency (EMEA, 2011). During these phases a formal proposal is introduced to the FDA or EMEA with all the details of the new drug. Upon approval, phase one starts with the testing on a group of healthy people in order to decide on the drug toxicity, liver and spleen reaction, the best dose amount, the best way to administer the new drug (oral, patch, intravenous, intradermal). The next two phases involve the testing on a group of sick people in order to decide on the new drug effectiveness. Phase two involves 100–300 sick people while phase three involves the testing on an extensive group with ethnographic differences that takes place in different hospitals. Since only a 5% of new drugs are approved to be circulated in the public, not many efforts continue with phase four where the approved drugs continue to be tested for side effects for many years after their first circulation.

Among the collaboration patterns that can be applied at the initial phase of our scenario is the CPat “Selecting Design Methods”
that involves collaboration on identifying, discussing and evaluating alternative design methods for a specific drug design. In this CPat, a VO member undertakes the role of the initial idea creator (IIC) while the other VO members comment and collaborate on producing an applicable new drug design method.

Relevant roles for the specific CPat are: VO Coordinator (3C HealthCare Ltd.), VO member IIC (Archimedes SA), VO member 1 (Acadic Deutschland KG) and VO member 2 (Progression SA) that correspond to European pharmaceutical companies that collaborate inside the VO. At some point of this VO operation, CPA evaluates the collaboration state (burst of H1N1 epidemic while there are not any design computations available for the group) and recommends the CPat to the VO Coordinator and to the VO members. CPA can recommend patterns to all possible initiators but accepts only the first one who will take up on this certain recommendation. In Fig. 16, the VO Coordinator who has already logged in to the CPA sees the recommendation to initiate the “Selecting Design Methods” CPat (see Fig. 16, area 1) and accepts this recommendation. VO Coordinator initiates the specific CPat by describing its context and in addition he/she uploads any necessary documents. He/She also defines the desired solution for the specific CPat by choosing the already designated template action list which can be further enriched with new collaboration activities or proposed tools that will facilitate the collaboration (see Fig. 16, area 2). In Fig. 16, areas 3 and 4 the VO Coordinator can find invitations to participate in other collaborations and can also be navigated across all active CPats, respectively.

All partners accept their invitation to participate in the specific CPat and Archimedes SA becomes the VO member IIC as he/she completes the first two activities by sharing a new design method with the group (see Fig. 16, area 2). Fig. 17 is a screenshot of the CPA client for the rest of the members where in the bottom section of the right pane (area 1) they can see in which collaborative activities they must be engaged in with respect to the specific CPat. Their engagement to the collaborative process consists of updating/adding/using items (information, comments, documents, tools, etc.) which can be found in the center pane below their assigned activity (see Fig. 17, area 2). We note here that since all these are collaborative activities, there can be more than one assignee to an activity. In addition any participant can contribute regardless of whether he/she has been assigned to do so. Collaboration participants are aware of the way that collaboration work proceeds, as they can search through all the related to CPats events (see Fig. 17, area 3). Moreover, CPA users can see details, e.g., due date, about a selected collaboration activity (see Fig. 17, area 4), or he/she can use the presence info pane (see Fig. 17, area 5) in order to find out who else is currently using CPA tool. Through this pane participants can communicate using the video conference call functionality. Once all collaborative activities have been completed the CPat is considered to be successfully completed and CPA notifies all participants.

In our case, when the new drug design idea becomes a formal proposal that is approved by FDA and EMEA, the VO must evolve to cope with the critical phases – drug testing in different labs and hospitals. Hence, the process of developing a new drug can continue with the initiation of another CPat such as “Add new VO member” for extending the VO with the addition of hospitals and/or testing labs with their own assets (testing knowledge, doctors supervising and volunteers) that will be used during the drug testing phases.

7. Evaluation

Collaborative systems present unique evaluation challenges because (a) they represent feature-rich environments, (b) their use is distributed over space and time, and (c) their contributions are perceived differently by people with varying backgrounds, goals, and priorities (Grudin, 1988). The range of approaches on evaluation demonstrated in prior work can be grouped into two broad categories. The first involves focused analysis of specific features or capabilities of a collaborative system using either experts or users as informants (Steves, Morse, Gutwin, & Greenberg, 2001). Such approaches are sometimes favored because they are inexpensive relative to laboratory or field methods (Drury, 1999), but are

![Fig. 16. CPat initiator accepts CPat recommendation and initiates it.](image-url)
limited because they do not account for the context that characterizes collaborative system use in situ. (Damianos et al., 2000). The same category includes questionnaires and experiments which have shown to be effective for identifying and measuring specific system capabilities (Haynes, Purao, & Skattebo, 2009). The second broad category of approaches to collaborative system evaluation involves field studies of deployed systems. With these approaches, real users are allowed to interact with the system in real-life situations. Experts investigate group interactions and transcripts that highlight critical, generally negative incidents (Ramage, 1999).

To demonstrate and evaluate our collaboration patterns based approach and tool, we combined scenario-based evaluation (Carroll, 2000) with field observations to determine users’ needs and reactions, followed by post-deployment surveys and interviews to assess specific system capabilities in the given deployments (Sokolov, 1999). Our evaluation objective was to determine whether the proposed pattern-based approach and system fulfills our research objectives. Specifically, we investigated the following aspects:

- System capability to guide effectively participants in successful collaborations.
- System ability to adapt collaborations by recommending resources and actions in response to changing circumstances.
- System ability to keep participants aware of collaboration activities.

We have conducted two studies: one at an inter-organizational and one at an intra-organizational setting. Both studies are based on the application domain of research and development project work, where benefits are expected from carrying them out by means of virtual e-science teams. The first study refers to a research laboratory active in the field of computer and communication systems comprising mostly co-located researchers. The second study refers to a consortium of industrial and research organizations embarked jointly in new product development in the domain of software engineering. In the two studies, we examined the performed collaborative activities with reference to specific collaborative scenarios. The scenarios described what the users were expected to do and specified the characteristics of the group that should carry out and the social protocols which should be in place.

The inter-organizational study involved four research institutions and four companies across Europe. These organizations took part in our investigation for a period of approximately two months. The users were allowed to use and configure freely available collaboration patterns. Table 3 presents an overview of collaboration-related data taken from the system log. Participants from each participating organization were interviewed in addition to the questionnaires that were distributed to gauge their reactions to the system.

The questionnaire results, depicted in Fig. 18, indicate that in both studies users perceived highly beneficial the system event-triggered recommendation functionalities, in particular the ability to adapt their collaboration by means of dynamically adopting a new collaboration pattern (75% responded positively, i.e. agree and strongly agree in the intra-organizational one). Sixty-five percent and 61% answered positively with respect to the system ability to guide users, 60% and 54% answered positively with respect to the system ability to keep users informed, 55% and 61% answered positively with respect to the system ability to make collaboration easier than before while 50% and 43% answered positively with respect to the system ability to make collaborations faster. Finally, users stated that through CPA it was easy to be aware of the actions of other participants by 65% and 72%, participant status by 60% and 71% and available resources by 55% and 57%.

Fig. 17. VO initiator views the CPat progress.
In the interviews that we performed, in both studies users agreed that the system was useful. The intra-organizational team perceived that the primary value was that it provided timely and relevant recommendations. The system ability to react to events generated by ongoing activities enabled users to adapt their collaboration by using a new pattern or inviting a new participant, and to identify relevant content, tools and other resources. The structured nature of patterns also helped to organize information pertinent to the collaboration in a way that made it easier to find information, as compared with searching through e-mail in order to find the most recent update. Finally, users reported they found beneficial being aware of the contributions of other collaborators in the activities of interest because it reduced redundancies and made their contributions visible to the team.

The inter-organizational team consisted of members who did not normally work together; CPA provided a way for these diverse members to collaborate with each other effectively, particularly by enabling the assignment of responsibilities, by guiding participants and informing them about pending and upcoming activities. For instance, someone working partially in the joint effort could check the status of an activity to see whether it had been completed, without needing to e-mail or call the person responsible for the activity. Another useful aspect of the structured representation was the ability to transfer knowledge from one participant to another. This was considered highly important because member replacement is typical in cases of VOs as individuals or even participating organizations may be replaced. It was possible to make the new member current because all the pertinent information was recorded in the activity.

One limitation of the system that was revealed during the evaluation was its lack of an access control mechanism. This allowed users to modify or delete arbitrarily collaboration patterns, data and data structures. This limits the applicability of the system in collaboration contexts where policies for access control need to be in place. A further limitation is that the system did not produce summary or aggregated reports about user activities that would allow management to view the overall status of the collaboration and to meet the needs of users who play different roles in collaborative processes. Further, participants would welcome better integration of the system with third-party tools (e.g., Google docs) as well as formal business processes.

8. Related work

Our work focuses on supporting pattern-based collaborations within service-oriented architectures comprising software-enabled, user-enriched services. In such an environment,
collaborations typically involve both humans providing their skills and experiences as services, as well as software services thus creating highly dynamic and complex interactions.

Supporting human tasks in SOA has been leveraged with standardized technologies such as BPEL4People (Agrawal et al., 2007) that target the support of human interactions as part of business processes (i.e., workflows) by designing and executing a set of human tasks, see e.g., WS-HumanTask (Amend et al., 2007). Production-quality platforms such as Oracle Beehive exploit such technologies to provide a range of integrated collaboration services including conferencing, instant messaging, email, calendar, and team workspaces. Such systems provide a robust infrastructure for enterprise users but they rely on the use of administrator-defined BPEL processes that do not target ad-hoc collaboration in which users can configure their collaboration activities and interactions on the fly. In another effort Dustdar (2004) introduced Caramba, a tool aiming at ad-hoc collaboration in virtual teams. Caramba organizes work items of individuals as activities that can be used to manage collaboration. The system enables virtual teams to participate in ad hoc collaborative work with some combination of modeled process templates. Schall, Dorn, Truong, and Dustdar (2009) have developed a method for designing service interfaces embodying human activities as actions offered by Web users, as well as a recommendation algorithm that is based on collaborative tagging of SOA resources (e.g., human activities) which helps to determine suitable resources drawn from properties of user preferences and measured similarity of human activities and actions.

Research in patterns has focused on various areas related to collaboration, such as workflow patterns which formalize recurrent problems and proven solutions related to the development of workflow applications (van der Aalst et al., 2003) or service interaction patterns which focus on collaborative business processes and apply to the service composition layer (i.e., orchestration, behavioral interface, and choreography) and to lower layers dealing with message handling and protocols (Barros et al., 2005). Other important efforts introduce activity patterns which model recurring human activities performed in the context of collaborative work (Harrison, Cozzi, & Moran, 2005; Moody, Green, Muller, Tang, & Moran, 2006) and task patterns as templates aiming at supporting users in executing either personal or collaborative work (Hu et al., 2009). Moreover, there are works targeting the analysis of patterns in collaborative environments, e.g., offline pattern mining (Dustdar & Hoffmann, 2007), runtime analysis of ad-hoc collaboration processes (Truong & Dustdar, 2009). Among the software tools that exploit the concept of patterns for supporting collaborative work, Lotus Activities is arguably the most prominent. Lotus Activities is a production-quality system built around the Activity-Centric Computing approach (Geyer et al., 2006) which is loosely inspired by Activity Theory and aims to address work fragmentation by allowing users to structure their work around the concept of an activity. In practice, activities help users organize and share their work, plan next steps and execute them. Similar to Lotus Activities in which patterns are used as aids for humans to realize collaboration activities, Kolschoten and Veen (2005) proposed tools that assist novice practitioners of GSS-based facilitation in order to better design and execute a collaboration process based on thinkLets. The suite has four tools, which each support a specific part of the process. The toolset is focused on design and preparation of a collaboration process but it does not support the execution. In the same direction of work, Keen et al. (2007) extend the artful business processes of Hill, Yates, Jones, and Kogan (2006), in an effort to integrate people with processes by mixing back-end automation with human interaction. Specifically, they focus on human-centric business process management, where they build business spaces for human workflows, facilitated by a number of human task management widgets (e.g., task/process definition list, escalation/process/task list, human workflow diagram, etc.). These widgets as browser-based rich internet applications (RIA) are configurable across 3 aspects: content that users work with, actions that users perform (e.g., user interaction patterns available) and display modes that users view.

Recently de Moor (2009) examined the use of collaboration patterns as conceptual building blocks for Community Informatics (CI). Socio-technical patterns work at a different level of detail from more technical design patterns and are especially useful at the beginning of social software projects to describe in a broad way – at the application domain level - the complex nature of the interactions between the social and technical systems that need to be built. Moreover, other notions similar to the patterns idea have been proposed. In Hamid et al. (2010) the authors propose a supporting tool (i.e. IT Support Conversation Manager) that tries to bridge business processes, collaboration tools and enterprise applications. They introduce the notion of conversation and conversation template as a logical container for capturing the interactions of people around the definition, refinement and enactment of a best practice process or a set of related processes. The elements of Conversations are Participants, Events that are records of actions taken by the participants and Tasks that are defined on-the-fly during a conversation. The generic patterns of compositions of tasks with dependencies among them, constitute the conversation templates.

In addition, we found efforts on patterns that use in some degree ontologies. Henninger and Ashokkumar (2006) proposed an ontology based (OWL) model for Usability Patterns. Although this pattern meta-model is not designed for collaboration patterns it is significant work that proves the benefits of ontology based modeling of patterns and gives concrete usage scenarios for a pattern ontology. Biuk-Aghai (2003) proposes the use of an ontology in order to describe and map between different levels of information, expressed in the Information Pyramid of Virtual Collaboration. Moreover ontologies for virtual collaboration patterns are mainly used to communicate meaning, and to reuse and organize knowledge. The e-Ace Project (2005) proposed an ontology structure that implements a “collaboration stack”. This ontology maps the various levels of pattern abstraction, ranging from abstract collaboration patterns to collaborative services and communication technologies. In that way it serves as a pattern hierarchy, allowing the automatic selection of lower-level patterns upon the selection of specific abstract patterns. Unified Activity Methodology has introduced an activity meta-model in the form of ontology (Moran, 2005). According to this, an activity is represented as an association of properties and as relationships to other entities.

Similarly to the aforementioned systems our approach leverages collaboration patterns for supporting ad-hoc collaboration in mixed software- and human-enabled service environments. Moreover, similar to Truong and Dustdar (2009), our system considers interactions and events that arise in collaborations, which act as sources for the recommendation of ‘collaboration solutions’, i.e., distributed resources including Web services, human services, activities and tools. The recommendation process is triggered by events generated dynamically within the on-going collaboration and processes with a Complex Event Processing (Luckham, 2001) framework such as Esper. In contrast to the previous works, our system utilizes events and collaboration patterns to perform run-time adaptation of collaboration. Moreover, our system uses ontologies to describe collaboration patterns and employs a reasoning process to match triggering events and conditions with collaboration solutions.

9. Conclusions

Collaboration is a complex human activity that involves both relatively stable routines and unplanned, spontaneous tasks.
Process-centric models allow for reusability because they can be applied several times as well as for automation because with technologies such as workflow management systems they can be enacted automatically. On the other hand, ad-hoc collaboration requires means for participants to easily define and customize their collaborative actions, at runtime.

CPats describe loosely collaborative problems and related facilities for CPats. They provide a mechanism for representing and executing process-centric models as well as ad-hoc collaboration by supporting both workflow-based solutions as well as lists of freely configurable actions. Workflows are useful for process-centric collaborative work. Action lists help users collaborate by informing them about the tasks they have to perform, the resources they may need in each step while allowing users to modify them in order to adequately support the specificities of the on-going collaboration. By employing the dual logic of CPat, the CPA system can support the fusion of process-centric with ad-hoc collaboration in an effort to balance reusability and flexibility. Moreover, by adopting an EDA, CPA can respond to changing circumstances by recommending relevant CPats.

The advantages of using ontologies to represent CPats lie in their ability to model effectively related concepts and interrelations, to map to collaboration services provided by different suppliers, to use reasoning to check values for validity and consistency, as well as to derive new facts based on the existing ones.

Our field trials with the system demonstrated that it had concrete benefits for the people collaborating on these activities, including better guidance of users, ability to adapt collaboration activities, easier knowledge transfer for people joining the team midstream, and a reduced need for tracking the status of activities and collaborators’ contributions. Nevertheless, field methods require typically an extended observation and analysis time frame (Grudin, 1994), and their results may still not provide for a comprehensive evaluation (Hughes, King, Rodden, & Andersen, 1994) but may also bias results towards that particular frame of reference. Hence, a long term evaluation of the system would lead to more conclusive findings. We plan to conduct more user evaluations of the system in order to determine which aspects of a CPat-based system are most important to users and how we should grow the system to best meet users’ needs.

Our current CPat representation approach allows for deterministic reasoning based on OWL-defined classes (i.e. OWL classes fully defined by using necessary and sufficient conditions). However, in reality there are cases where collaboration-related data are uncertain and hence reasoning becomes inexact. Uncertainty can be for example on whether a triggering event actually occurred, or on the matching between participants with specific capabilities and a role, or on the matching between the condition that needs to hold and the CPat that is to be used for this condition. All these circumstances of collaboration uncertainty can be mapped to uncertainty about ontology classes, relations and individuals. Our future work will exploit the use of probabilistic reasoners in combination with the ontology-based description of CPats and contrast them with pure OWL-based reasoning in terms of expressive power, ease of authoring, and robustness to changes in the schema. We are currently exploring the use of Markov logic networks (Richardson & Domingos, 2006) as a computation medium able to provide probabilistic reasoning and learning facilities for CPats.

References

Hlupic, V., & Qureshi, S. (2002). What causes value to be created when it did not exist before? A research model for value creation. In R. H. Sprague, Jr. (Ed.), Proceedings of the 36th Hawaii international conference on system sciences (HICSS-36), Big Island, Hawaii (pp. 1–10). IEEE.