Collaboration Patterns in Event-Driven Environments for Virtual Organizations

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

*Institute of Communications and Computer Systems, National Technical University of Athens
9 Iroon Polytechniou Str., Athens, Greece, Tel. +302107723895
** Informatics Department, University of Piraeus
80 Karaoli & Dimitriou Str., Piraeus, Athens, Greece, Tel. +302104142314

Abstract

Modern enterprises tend to form virtual but legally consolidated schemas for collaborating in order to function and survive in constantly changing grounds, where competition always grows and the emergence of new technologies keeps on posing new challenges. They collaborate in virtual organizations, where several segments of work or parts of collaboration can be often identified as recurring and be reused. In order to take advantage of this repetition, we consider the introduction of Collaboration Patterns in VOs. We propose a combined schema for using Collaboration Patterns along with the initiatives of the Event driven technology, in order to capture the ad hoc nature of these collaborations and cope with the demanding and rapidly changing environments of virtual organizations.

1. Introduction

Enterprises in the 21st century are operating in an environment where market constantly changes, new technologies continually emerge and competition is considered to be fierce at the global scale [Canavesio & Martinez, 2007]. They cope with critical challenges in order to survive, in such a competitive market environment, that the single enterprise model is not sufficient to cover all aspects of their needs. Nowadays, they try to establish a virtual but legally consolidated schema for collaborating with main objective to deal with these critical challenges, by expanding their influence area and widening their profits, in terms of a virtual organization (VO). A virtual organization is defined as: *"A temporary alliance of independent enterprises that come together to share skills, core competencies and resources in order to better respond to business opportunities and whose cooperation is supported by computer networks."* [Luczak & Hauser, 2005].

Since one of the virtual organizations' main characteristics is the continuous change, their efficiency is determined by the speed and accuracy with which information can be managed and exchanged among the business partners [Sandakly et al., 2001]. The several cooperating nodes in virtual organizations are geographically distributed while their coordination takes place through electronic communications. So, the integration of business partners using information and communication technology poses several challenges that may be handled by the usage of the appropriate technology.

Such a technological approach that can be very valuable for supporting the demanding and rapidly changing environment of VOs, is considered to be the Event-driven architecture (EDA). EDA is a software architecture approach for promoting the production, detection, consumption of, and reaction to events [Luckham, 2002]. This kind of architectural approach may be applied for the design and implementation of applications and systems which transmit events among loosely coupled software components and services [Chakravarty & Singh, 2008] that sounds exactly like the complicated collaboration environment of a VO.

Furthermore, we explore the concept of Collaboration Patterns in VOs, as segments of work or parts of collaboration, which can be identified as recurring and can be reused. The reuse of Collaboration Patterns can constitute an advantage in collaborative environments such as VOs, where there is an increased need for modeling, executing, monitoring and supporting the dynamic nature of collaborations. The overall aim of using Collaboration Patterns in the VOs domain is to enhance support of the networked enterprises in a successful and fast way, by providing an approach for capturing and reusing relevant to collaboration operations.

In this paper, we present our preliminary work regarding the combination of an event driven architecture and collaboration pattern services for coping with the critical challenges that a VO environment poses. In section 2 we refer to the notion of patterns and the potential combination of Collaboration Patterns with complex events. We continue in section 3, by analyzing collaboration pattern services that can help the collaborations inside the VOs, while in section 4 we present an architectural proposition for supporting their implementation. In section 5, we give an illustrative scenario of how the event-based architecture can be used to

support collaboration based on collaboration patterns. Finally, this paper concludes with the related work and a discussion regarding the future work on this effort.

2. Collaboration Patterns

2.1 Concept of Patterns

In the modern competitive business environment the notion of collaboration involves the complex nature of vast number of processes that may span across organizational/geographical boundaries, through the world wide web. The need for coping with iterative problems or reusing iterative segments of collaboration processes (*patterns*) seems more imperative than ever before. It is often said that patterns are discovered rather than invented. This is true in the sense that models turn into patterns only when it is realized that they may have a common usefulness. When experts work on a particular problem, it is unusual for them to tackle it by inventing a new solution that is completely distinct from existing ones. They often recall a similar problem they have already solved, and reuse the essence of its solution to solve the new problem. This kind of 'expert behavior' is a natural way of coping with any kind of problem or social interaction [Buschmann et al., 1996].

For many people, the word *pattern* has appeared almost entirely due to the work of Christopher Alexander, a professor of architecture at the University of California at Berkeley. His pattern language book [Alexander et al., 1977], is seen as the prototype to patterns books in software. During this first use of patterns in architecture, he defined a pattern as a "*morphological law that explains how to design an artifact in order to solve a problem in a specific context*".

This concept has also been adopted in the context of software engineering, in order to capture best practices for creating software and using common concepts to solve recurring problems. The first notable publication in this area was the book "Design Patterns: Elements of Reusable Object-Oriented Software" by the so-called Gang of Four [Gamma et al., 1995], that advanced the popularity of patterns in computer science.

2.2 Collaboration Patterns

An organization encounters many problems in its day-to-day operations, collaborations and in its strategic positioning against or besides other organizations. In general, collaboration participants need to be collocated either physically or virtually in order to make a decision or work together. During such collaborations, recurring segments of problems or actions can be identified and introduced as Collaboration Patterns for future use. These patterns in terms of collaboration can also be regarded as prescriptive, providing means for modeling collaborative tasks and protocols of cooperation [Molina & Bell, 1999], while they can guide the configuration of Collaborative Working Environments (CWEs) to meet the requirements of the participants [Slagter et al., 2005], [eAce, 2005].

However, although a collaboration pattern determines the basic structure of the solution to a particular collaborative problem, it does not always specify a fully detailed solution. A pattern 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 design problem at hand. The CPs are not only useful to users within the same domain as the pattern, but frequently a pattern is useful in other domains as well.

In direction of having a common understanding about what a collaboration pattern (CP) is , including at the same time the main elements of the, so far, expressed descriptions of a pattern [Buschmann et al. 1996], [Schummer & Lukosch, 2007], [Anderson, 2000], [Dyson & Longshaw, 2004], 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."

The vision in Collaboration Patterns is that the solution can be described as an encapsulated component that can be reused whenever a collaborative situation/problem occurs [Schümmer, 2002]. For a solid and unambiguous description of CP we need attributes that present the description of the problem and its solution that the CP can address in a specific context and under specific pre-conditions that must be satisfied. So, a CP needs a certain *Name* as a collaboration pattern identifier, a *Category* that positions the specific pattern in terms of VO collaborations, a *Problem* that describes the issue the CP has addressed before or it is expected to address in the future, a *Context* that describes the space in which the pattern can be embedded and be applicable and *Pre-conditions,* that dictate the states and conditions that must be satisfied before the specific CP can be considered applicable. One of the most important elements of a CP's description is the *triggering* part of its execution. In an event-driven approach simple or complex events can trigger the execution of a specific collaboration pattern satisfying the need for reactivity and attention amplification in the environment of VOs. Another attribute that is needed for describing a CP is the *Solution* that comprises of prescriptions of solutions to the designated problem and usually involves actions, workflows or even collaborative tools. This solution will create a set of *Post-conditions* that are conditions and states that hold after the

successful termination of the collaboration pattern. In addition *Exceptions* must be stated once describing a CP as different course of action that may involve the execution of one of the *Related Patterns* that can be executed sequential, in parallel or after its termination.

2.3 Collaboration Patterns and Complex Events

Events are data elements that capture the state (or changes to state) of real-world or computer-based objects. Events consist of data and temporal attributes that represent the "what" and "when" of an object - the state of an object or the interaction of objects at a particular time [Kobetz, 1993], [Heemels & Sandee 2006]. Nowadays, the complex event processing (CEP) has become a hot research topic as it handles the extraction and process of complex events from an event cloud [Chakravarty & Singh, 2008]. [Luckham, 2002]. Here, only a partial temporal order of events exists while the «publishers» of events don't know anything about their corresponding «subscribers».

Event-driven approaches were primarily used for handling business transactions that involve vast number of events (thousands per second). But event driven approaches are considered to be quite valuable also for systems that don't have to deal with such vast numbers of events. This is because these approaches:

- introduce a kind of reactive dynamics in the system
- enable active responding on signals sensed/derived within a given VO (attention amplifier)
- solve interoperability issues among the dispersed collaborating organizations, as the system perceives everything as events

In terms of a system for handling knowledge based collaborations inside VOs, services may produce many events that might be relevant for other services. It is clear that all these influences, due to their ad-hoc nature, cannot be defined in advance explicitly, but they can be categorized and used as triggering mechanisms of reoccurring segments of collaborative work (Collaboration Patterns).

3. Collaboration Patterns Services in VOs

In this section we define our proposed approach needed for handling VOs in an easier and more effective way by introducing Collaboration Patterns. In such an approach we need to identify both design time and run time functionalities.

Design time functionalities involve the creation, editing and validation of Collaboration Patterns according to a predefined collaboration goal which constitutes the goal of the VO. The editor can model recurring segments of collaborations as CPs or he can use data mining techniques to capture them from previous collaborations. The functional tool that facilitates this work is the Collaboration Patterns Editor (CPE) that is responsible for defining, editing, searching and simulating Collaboration Patterns.

The actual handling of events becomes an issue to be supported through run time functionalities. The idea here is to use the Collaboration Patterns Assistant (CPA) in order to help users in executing and monitoring collaboration services according to the selected/triggered collaboration pattern. We can distinguish three kinds of services during run-time:

1. *Recommender Services*

This kind of service provides the recommendation of actions and collaboration tools to be used. Firstly, it recommends actions to continue the collaboration. 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 CP (in parallel, sequential, by terminating the old one), or an action / action list. Secondly, it recommends the usage of collaboration tools (shared virtual space, workflow, e-mail, wiki etc.), related to the current collaboration. In terms of a VO an example of a recommender service it could be: When a collaborator systematically (in a given time period) does not participate" the system provides the choice between different Collaboration Patterns: a) Schedule a project management meeting to discuss it, b) Automatically terminate collaboration after 2 months.

2. *Awareness Services*

These services provide awareness of the state of collaborators and the state of collaboration work. Firstly, they take input from the Collaboration Knowledge Base regarding the several aspects of the on-going collaboration and provides as output, in a user friendly way, new or updated documents related to the current collaboration, visualization of the previous steps of collaboration or CP and notification about the possible next steps of collaboration. Secondly, it takes input from the Collaboration Knowledge Base regarding the several information of the on-going collaboration and provides as output reports about people in charge of tasks, which users are on-line collaborating and what are the results of specific collaborations. An example could be the update of the user dashboard (e.g. "all collaborators have downloaded the proposal").

3. *Analytics Services*

Analytics services can provide a statistical analysis based on previous and ongoing collaborations. It uses social networking analysis techniques in order to present indicators and metrics associated to the past / ongoing collaborations and to the past or potential collaboration partners (e.g. partner X involved successfully in 10 VOs about automobile manufacturing or Y number of documents were exchanged between collaborators / per day, week, month).

4. Event-Based Architecture for Collaboration Patterns

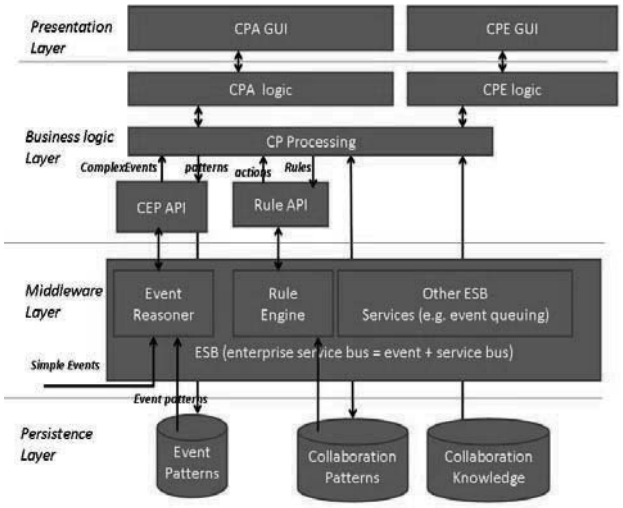
This proposed architecture articulates a conceptual view of a system that supports the description, identification and

usage of CPs and events during the life cycle support of VOs.

In order to support the services mentioned in the previous section the following components are required (figure 1):

- Event service bus which belongs to the middleware level. It acts as a lightweight, ubiquitous integration backbone through which events flow through intelligent routing mechanisms. Specifically it is the actual bridge between the several event sensors that perceive events from an event cloud and the complex event processing engine.
- Event reasoner which belongs to the middleware layer. It has the capability to process simple events, combine them and produce complex constructs of events that may trigger Collaboration Patterns.
- Rule engine as another component of the middleware level is triggered by a (complex) event and uses facts (from Collaboration Knowledge Base) to derive recommendations. Examples of such recommendations include: recommend a new CP, enact a new CP, terminate a CP and substitute a CP by another.

Figure 1: Architecture for Collaboration Patterns



The rationale of our approach is based on the event-condition-action (ECA) rules [Behrends, 2006] for patterns. These types of rules are understood as follows: whenever an event occurs, the rule engine performs the actions of all matching rules whose conditions hold. Simple events arrive at the Event Reasoner, which maintains (partially detected) complex events in its pattern base. Event patterns remain in the store until they reduce to true (occurred) or false (impossible). As soon as an event pattern reduces to true, the Event Reasoner notifies the rule engine about the corresponding complex event's occurrence, and the engine applies the matching CP rules (preconditions and triggers), in order to derive recommendations.

In the persistence level three information sources are required:

- A collaboration knowledge base. The knowledge base contains domain models as well as facts about the collaboration that may change as the collaboration proceeds. It provides information about the state of collaborators and the state of collaboration work, while it aggregates events and their consequences for the ongoing collaborations.
- Collaboration Pattern base which stores Collaboration Patterns.
- Event patterns base which stored event patterns.

CPs that have been created by the CP editor can be combined and instantiated in order to create a collaboration workflow combined with human action. When a collaboration workflow is executed it interacts with the other components by generating events. Moreover, during execution, the workflow generates events which are used by the system in order to provide awareness and analytics services to users. Finally, the system monitors the on-going collaboration and modifies its execution by recommending new CPs or performing changes in CPs. The appropriate interaction takes place through the presentation layer and the corresponding graphical user interfaces.

5. Illustrative Scenario

In this section, we give an example of how the event-based architecture can be used to support collaboration based on a specific collaboration pattern. The virtual organization of our example comprises a number of pharmaceutical companies and intends to develop and test a series of new dermatological drugs. The recurring collaboration activities that can take place in the context of this VO are those related to the organization of a joint laboratory experiment. In order to schedule such an experiment, the VO coordinator notifies the collaborating organizations about the need for a joint experiment. The coordinator then uses a web tool for reaching consensus on the date of the experiment and for agreeing on the experiment specific objectives. These activities are part of a pattern which we call "OrganizeExperiment". The experiment should be performed periodically (e.g., every three months) or can be imposed by an external event such as the release of a new regulation from the Foods and Drug Administration which requires some additional joint laboratory experiments before the new dermatological drugs enter the clinical testing phase. In both cases, it is expected that at least three partners agree and also that the necessary budget for the experiment exists. Once the experiment is organized, it takes place and the results are captured and stored in a shared repository. In this collaboration pattern, it is also important to include cases, such as the revision of the regulation, in which the execution of this collaboration pattern must be stopped or postponed. The actual process of postponing the experiment can be the subject of a different collaboration pattern.

The aforementioned collaboration pattern can be triggered when a specific pattern of simple events occurs. For example, when the last experiment was conducted more than three months ago, or when it was conducted less than three months ago and a new regulation has been published and the partner responsible for monitoring the Federal Register notifies the coordinator that the new regulation affects one of the new drugs. We call this complex event "ExperimentTrigger".

We use a simple language based on event-condition-action (ECA) rules for Collaboration Patterns. This type of rule is understood as follows: whenever an event occurs, the rule interpreter performs the actions of all matching rules whose conditions hold. The following pseudocode snippets are typical for collaboration activities which can constrain how activities and events interrelate – for example an ExperimentTrigger event should lead to the organization of a new experiment. The IF part includes zero or more gating requirements (a "?" indicates a variable):

```
WHEN ExperimentTrigger( ?Partner, ?Drug,
        ?DateofLastExperiment)

IF ExperimentConditions( ?PartnersAgreed > 3
        AND ?BudgetAvailability = TRUE)

THEN OrganizeExperiment( ?Partner, ?Drug,
        ?PossibleDatesofNewExperiment)
```

Event-driven architectures typically provide means to deal with exceptions. Exceptions can be anticipated or unanticipated [Chakravarty & Singh, 2008]— an anticipated exception, such as a delayed shipment, is one that business analysts have considered but that the IT system doesn't properly handle, whereas an unanticipated exception is one that business analysts have failed to model. Similarly to [Chakravarty & Singh, 2008], our approach shows how to address anticipated exceptions in modeling or during configuration. An anticipated exception in the aforementioned scenario that would cancel the enactment of the specific collaboration partner can be that a revision to the FDA regulation has just been announced on Federal Register.

6. Related Work

Within the domain of cross-organizational collaboration, patterns have been mainly used to facilitate the modeling and enactment of inter-organizational business processes. Existing research has examined patterns in the area of control flow [van der Aalst et al 2003], data flow [Russell et al., 2004a], and resources [Russell et al, 2004b]. For example, [Norta et al, 2006] developed a pattern-knowledge base for supporting inter-organizational collaboration.

Our work, however, is different from cross-organizational business process management. We focus on the use of patterns as a vehicle for supporting knowledge-based collaboration [Biuk-Aghai, 2004]. Our goal is to address the specificity frontier of collaboration support systems [Bernstein, 2000], i.e. to support a range of collaboration types, from fixed to ad-hoc. In this respect, our work extends research on activity patterns [Moody et al, 2006] and proposes Collaboration Patterns as a type of design patterns applicable on event-based systems [Paschke, 2008].

Regarding our architectural proposition, we extend previous research on event-based architectures for cross-organizational coordination [Chakravarty & Singh, 2008], as well as efforts that adopt patterns for knowledge workflow management [Sarnikar and Zhao, 2008].

7. Conclusions and Future Work

In this paper we have discussed the introduction and combination of Collaboration Patterns and event driven architectures to the challenging environments of virtual organizations. These environments present an ad-hoc nature that cannot always be defined in advance explicitly and embody several cooperating nodes that are geographically distributed, posing in such way technological challenges regarding the support of collaborations and their respective coordination.

The virtual organizations as collaborative environments with increased needs for modeling, executing, monitoring and supporting the dynamic nature of collaborations that are involved can be favored by the combined strength of collaboration Patterns and events that may act as triggering mechanisms for reoccurring segments of collaborative work. We believe that this proposal can cope with the critical challenges that a VO environment poses. Further work will be carried out in terms of implementing this combination of event driven architecture along with Collaboration Patterns and validating it across real case scenarios in VOs lifecycles.

Acknowledgements

This work has been partially funded by the European Commission regarding the strep project SYNERGY (Supporting highlY-adaptive Network Enterprise collaboration thRouGh semanticallY-enabled knowledge services), ICT No 63631. The authors would like to thank the project team for comments and suggestions.

References

Alexander, C., Ishikawa, S. And Silverstein, M., 1977. A Pattern Language: Towns – Buildings – Construction. *Oxford University Press*, London.

Anderson, F., 2000. *A collection of history patterns, Pattern Languages of Program Design*. 263–297, Reading, MA, USA: Addison-Wesley.

Behrends, E., Fritzen, O., May, W. and Schubert, D., 2006. An ECA Engine for Deploying Heterogeneous Component Languages in the Semantic Web. *In Web Reactivity (EDBT Workshop)*, Springer LNCS 4254.

Bernstein, A., 2000. How Can Cooperative Work Tools Support Dynamic Group Processes? *Bridging the Specificity Frontier, CSCW'00*, Philadelphia, PA.

Biuk-Aghai, R., 2004. Patterns of Virtual Collaboration in Online Collaboration Systems. *Proceedings of the IASTED International Conference Knowledge Sharing and Collaborative Engineering*, ACTA Press: 57-62.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. 1996. *Pattern-oriented software architecture. A system of patterns*. John Wiley & Sons Ltd, Chichester, UK.

Canavesio, M. M. and Martinez, E., 2007. Enterprise modelling of a project oriented fractal company for SMEs networking. *Computers in Industry* 58: 86-92.

Chakravarty, P. and Singh, M. P., 2008. Incorporating Events into Cross-Organizational Business Processes. *IEEE Internet Computing,* 46-53.

Dyson, P. and Longshaw, A., 2004. *Architecting Enterprise Solutions: Patterns for High- Capability Internet-based Systems*. John Wiley & Sons Ltd, Chichester, UK.

eAce 2005. *eAce project*. Available online at: http://www.eace-project.org/index.jsp .

Gamma, E., Helm, R., Johnson, R. and Vlissides, J., 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.

Heemels, W.P.M.H., and Sandee, J.H., 2006. Practical stability of perturbed event-driven controlled linear systems. *Proceedings of the American Control Conference*, 4379–4386, Minneapolis, Minnesota, USA,.

Kopetz, H., 1993. Should responsive systems be event-triggered or time-triggered? *Transactions on Information and Systems* E76-D(11): 1325–1332.

Luckham, D.C., 2002. *The Power of Events*. Addison-Wesley.

Luczak, H. and Hauser, A., 2005. Knowledge management in virtual organizations. *Proceedings of International Conference on Services Systems and Services Management*, 898.

Molina, A. and Bell, R. 1999. A Manufacturing Model representation of a flexible manufacturing facility. *Journal of Engineering Manufacture: Proceedings of the Institution of Mechanical Engineers* 213(B): 225-246.

Moody, P., Gruen, D., Muller, M., J., Tang, J. and Moran, T., P., 2006. Business activities patterns: a new model for collaborative business applications. *IBM Systems Journal* 45(4): 693-694.

Norta, A., Hendrix, M. and Grefen, P., 2006. A Pattern-Knowledge Base Supported Establishment of Inter-organizational Business Processes. *R. Meersman, Z. Tari, P. Herrero et al. (Eds.): OTM Workshops 2006*, LNCS 4277: 834–843.

Paschke, A., 2008. Design Patterns for Complex Event Processing. *2nd International Conference on Distributed Event-Based Systems (DEBS'08)*.

Russell, N., ter Hofstede, A., H., M., Edmond, D. and van der Aalst, W., M., P., 2004a. *Workflow Data Patterns. QUT Technical report, (FIT-TR-2004-01)*.

Russell, N., ter Hofstede, A., H., M., Edmond, D. and van der Aalst, W., M., P., 2004b. Workflow Resource Patterns. *BETA Working Paper Series*, WP 127, Eindhoven University of Technology.

Sandakly, F., Garcia, J., Ferreira, P. and Poyet, P., 2001. Distributed shared memory infrastructure for virtual enterprise in building and construction. *Journal of intelligent manufacturing*, 199.

Sarnikar, S. and L. Zhao, 2008. Pattern-based knowledge workflow automation: concepts and issues. *Information Systems and E-Business Management* 6(4).

Schümmer, T., 2002. Constructing a Groupware Pattern Language. *Workshop on Socio-Technical Pattern Languages, part of CSCW* 2002.

Schümmer, T. and Lukosch, S., 2007. *Patterns for computer-mediated interaction.* John Wiley & Sons Ltd, Chichester, UK.

Slagter, R., Biemans, M. and Jones, V., M., 2005. *Service Orchestration for Collaboration Patterns*. Technical Report TR-CTIT-05-71, Centre for Telematics and Information Technology, University of Twente, Enschede, ISSN 1381-3625.

van der Aalst, W., M., P., ter Hofstede, A., H., M., Kiepuszewski, B. and Barros A., P., 2003. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51.