A Review of Patterns in Collaborative Work

Yiannis Verginadis¹, Nikos Papageorgiou¹, Dimitris Apostolou², Gregoris Mentzas¹

¹Institute of Communications and Computer Systems
National Technical University of Athens
9, Iroon Polytechniou Str., Athens, Greece
Tel. +302107723895
{jverg, npapag, gmentzas}@mail.ntua.gr

²Informatics Department University of Piraeus
80, Karaoli & Dimitriou Str., Piraeus, Athens, Greece
Tel. +302104142314
dapost@unipi.gr

ABSTRACT
Patterns, repeatable processes for recurring high-value tasks, have great potential for assisting computer-mediated collaboration. In this paper we focus on patterns for collaborative work as a means to capture best practices about recurring collaborative problems and solutions amongst dispersed groups. We present a comparative review of relevant research and commercial efforts related to patterns that can be used to facilitate collaboration.

Categories and Subject Descriptors
A.1 [Introductory and Survey]. H.5.3 [Information Systems Applications]: Group and Organization Interfaces – collaborative computing, computer-supported cooperative work, web-based-interaction.

General Terms: Management, Design.

Keywords: Collaboration, Patterns, Survey

1. INTRODUCTION
The concept of patterns was inspired by the way experts tackle work on a particular problem without re-inventing the wheel. In computer science and engineering, design patterns are predefined solutions for problems that may occur in many repeatedly. Patterns were first studied systematically in the field of engineering by C. Alexander, a professor of Architecture in University of California, Berkley. His book, where he describes a language for architectural patterns [1], 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 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 [2], that advanced the popularity of patterns in computer science. The idea of patterns has also been introduced in the field of Human Computer Interaction. Tidwell [3] 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 [8, 12, 24].

In this paper, we focus on the role that patterns can play in the context of collaboration. Collaboration is a critical business function that demands skills and knowledge, spanning a wide range of domains (including social, business and technical domains). Collaboration patterns can be a means to capture best practices about recurring collaborative problems and solutions. In section 2 we discuss the role of patterns in facilitating collaboration. In section 3 we present our survey of prominent pattern approaches relevant to collaboration and we conclude in section 4 by discussing on the implications of our survey.

2. THE ROLE OF PATTERNS IN FACILITATING COLLABORATION
Although organizations seem to invest increasingly in collaboration tools, they still lack knowledge on how to apply such tools effectively [47]. Collaborating partners may find the collaboration tools easy to operate, but they cannot typically use their full potential [38]. Moreover, they cannot readily identify what the best collaboration practice is, when to embark on such a practice and how to facilitate it using available tools [18]. To alleviate problems in collaborative work and to encourage the sustained use of collaboration tools in organizations, patterns have been exploited as models for repeatable processes for recurring high-value collaborative tasks. Similar to software engineering, patterns have influenced how e-collaboration is realized. Patterns provide a vocabulary for expressing collaboration best practices and concise representative designs and detailed implementations.

Approaches, like Collaboration Engineering [4], suggest human-based methods for designing, modeling, and deploying repeatable collaboration processes using patterns. Nevertheless, we argue that there are more ways for e-collaboration to benefit from patterns. Taking into account the various ways in which patterns have been used in other domains such as for mining best practices, for benchmarking practices and for recommending relevant practices, it is worth examining the potential of such ways within the context of e-collaboration. In this study, we set out to survey a broad spectrum of patterns approaches, identify their main characteristics and assess their potential benefits and implications if applied in the context of collaborative work.

3. SURVEY OF PATTERNS APPROACHES RELEVANT TO COLLABORATION
In this section we outline prominent approaches that use patterns to facilitate collaboration directly or indirectly. For each of the
approaches, we present a short overview along with their key underpinnings.

3.1 Collaboration Patterns in Communities of Practice

One of the research efforts that we reviewed was the work on patterns in Communities of Practice (CoPs) [5]. CoPs are living organisms that build up a collective history of information created, discussions conducted, tasks performed, and goals accomplished. Traces of collaborative activities can be found in the many technologies used, such as mailing list archives, web pages, and document repositories. However, due to the longitudinal and distributed nature of virtual communities, it is very hard to recollect and recombine relevant information in detail. According to de Moor [5] collaboration patterns are “a particular class of patterns that capture socio-technical lessons learnt in optimizing the effectiveness and efficiency of collaboration processes”.

De Moor [5] shows how formalized collaboration patterns can be used to cultivate CoPs by improving their collective, distributed memory of communicative interactions and information. A methodology that in a structured way analyzes how collaboration patterns in CoPs can be elicited, represented, analyzed, and applied is outlined in his work. Collaboration patterns are used to create a conceptual layer on top of a community information system consisting of data resources, information and communication technologies and human users. The patterns defined in this layer helptrigger human and machine processes when certain states and events in the information system occur. Given that collaborative community has two main types of community memory resources, computerized knowledge bases and community members, these resources are described by memory patterns, capturing their essential properties from a collaboration point of view.

3.2 Collaboration Engineering

Briggs [4] has suggested Collaboration Engineering (CE) to refer to a research-based approach that designs, models, and deploys repeatable collaboration processes for recurring collaborative tasks. Two kinds of patterns are pertinent to CE: ThinkLets and Collaboration Patterns in Knowledge Management.

3.2.1 Thinklets

CE processes are modeled with standardized and codified components called thinkLets, which yield predictable, repeatable patterns of collaboration among people working together toward a goal. According to [4] “a thinkLet constitutes the smallest unit of intellectual capital required to create one repeatable, predictable pattern of collaboration among people working toward a goal”. A thinkLet describes an elementary group process from a leader’s point of view by providing explicit, scripted prompts for the group, and by guiding the practitioner through the decisions that must be made based on the group’s behavior. In addition, a thinkLet defines which Group Support Systems (GSS) to use and how to configure each of them. Briggs [4] has introduced a formal documentation method for thinkLets named ThinkLet Description Documents. The knowledge encapsulated in a thinkLet can reduce the conceptual load of using GSS. By following a prescription for the tool, the configuration and the script, one may be able to recreate the stimulus used by others to produce a pattern of collaboration without having to comprehend the complete range of support a GSS may offer.

ThinkLets comprise five general patterns: diverge, converge, organize, evaluate, and build consensus. For each of the general patterns, specific thinkLets have been codified capturing the best practices of highly experienced meeting facilitators [4], [6]. When they are put together to form a coherent whole, the resulting CE processes provide ready-to-apply ‘recipes’ for successful meetings to be conducted by other facilitators, or even by novices and practitioners themselves using facilitation techniques and technology.

ThinkLets are mostly distributed through books in a textual form. [6] presented a conceptual object oriented model for thinkLets and collaboration processes. ThinkLets can be combined with other thinkLets yielding compound thinkLets that evoke more complex patterns of collaboration, or a sequence in the pattern of collaboration.

3.2.2 Collaboration Patterns in Knowledge Management

Quereshi at al. [7] introduced a framework that illustrates the interdependence of knowledge management and team collaboration. They explain how specific knowledge management activities correspond to the five collaboration patterns introduced by Briggs (diverge, converge, etc.). Regardless of which knowledge management activities a team performs, the activity will require that the team engages in one or more of the five collaboration patterns. Their work illustrated the interdependence of knowledge management and collaboration by mapping collaboration technology to knowledge management activity by the pattern of collaboration that the technology can foster.

3.3 Event Patterns

In the domain of information systems architectures, Chakravarty et al. [8] outline the role of event patterns in Event-driven Architectures (EDAs), and propose an architecture which aim to address the engineering challenge of incorporating events in a Web process in a reusable manner - that is, without hard coding events into a process model. An EDA provides a way of organizing systems that sense, analyze, and respond to events. For business processes, sensing involves receiving events from multiple sources, analyzing involves deciding a response typically by aggregating such events and responding involves updating expectations and modifying executions. Events may be simple or complex. A simple event instance is specified through its name and parameter values; typical parameters include a transaction ID, when and where the event occurred, and other domain-specific content. A complex event is expressed as a pattern over simple events.

Barros et al. [9] distinguish four categories of event patterns: co-occurrence, time relation, data dependency and consumption. Each comes with a short description, examples and an assessment of its support by Business Process Execution Language (BPEL) and Business Process Modeling Notation (BPMN). In [10], Business Event Modeling Notation (BEMN) introduced as a formal language for business events and event patterns. BEMN serves to define event rules. Each rule consists of an event pattern description, specifying combinations of events that can be matched, as well as a set of output event declarations indicating what kind of events are to be produced as a result of the firing of a rule.

3.4 Workflow and Process Patterns

In the domain of workflow and business process management, six pattern-based approaches have been identified and outlined below.
3.4.1 Workflow Patterns
Van der Aalst et al. [11] proposed a set of workflow patterns, initially to characterize only the kinds of control flow that appear frequently in workflow processes. Workflow patterns are expressed as a means to indicate business requirements. Workflow patterns range from fairly simple constructs presented in any workflow language to complex routing primitives not supported by many workflow management systems. In process-aware information systems various perspectives such as control-flow, data, resource and operational can be distinguished [12].

The following set of attributes is used to describe workflow patterns: description, synonyms, examples, problem and potential implementation strategies. Initially, five categories of workflow patterns have been identified: Basic Control Flow Patterns, Advanced Branching and Synchronization Patterns, Patterns involving Multiple Instances, State-based Patterns and Cancellation Patterns.

3.4.2 Patterns for Business Process Management
Dan Atwood [24] describes BPM process patterns as constructs that connect activities together to solve a common problem. BPM process patterns serve as a simple technique to shorten the learning curve and improve productivity and quality of process design. BPM process patterns can be combined to form complete processes. BPM process patterns follow the structure and categorization of workflow patterns [12]. Hence, they are broken into the six categories: Basic Control, Advanced Branching, Structural, Multiple Instances, State Based and Cancellation. For each van der Aalst’s control flow pattern Atwood gives a business process example which includes a short scenario and a graphical example along with guidelines for its usage in business process modeling.

3.4.3 Patterns for Business Object Frameworks
The aim of the work of Barros [13] was to accomplish the encapsulation of high level knowledge and business logic in the design of Business Object Frameworks (BOF). BOFs are derived from formal and explicit Business Process Patterns. The purpose of a BOF is to provide a generalized solution to a business process design problem that can be used to develop an object-based software application for any particular real-life problem in a specific business domain. It has been demonstrated that a pattern and a framework derived from it can be applied to the design of a process for a given business in a domain or to the development of an Information System in support of such a process. In other words, business process patterns are generalized designs that include best practices for businesses in a given application domain. For a given domain, beyond best practices usually expressed in a specific business logic, business process patterns share a common structure of activities and flows. Thus, products or service provision processes, such as manufactured goods, health services, justice services, financial services, etc., share a common structure. Business process patterns are represented with an activity-based modeling scheme that uses IDEF0, a method designed to model the decisions, actions, and activities of an organization or system.

3.4.4 Patterns for Inter-Organizational Business Processes
Norta et al. [14] suggest the use of patterns in inter-organizational business processes. The concept of Dynamic Inter-organizational Business Process Management (DIBPM) was introduced in [15] as a model and framework that addresses the need of organizations for dynamically bringing together a service consumer and a service provider over web-based infrastructures where the service is a business process. It is defined that “a dynamic inter-organizational business process is formed dynamically by the automatic integration of the sub-processes of the involved organizations”. Here dynamically means that during process enactment collaborator organizations are found by searching business process market places and the sub-processes are integrated with the running process.

A pattern meta-model is being used for patterns in order to assist the users to comprehend better the differences, commonalities, relationships of patterns. The meta-model is organized in four detailed packages: (i) The Package Pattern contains all classes that capture information for specifying patterns. (ii) In the Taxonomy package, classes are contained that capture information about DIBPM perspectives; this package contains classes that create a taxonomy into which patterns can be embedded. (iii) The Support package encapsulates classes for managing information about technologies that support patterns. (iv) The User Management package captures information of different users of the pattern repository, e.g., administrator, reviewer, pattern submitter, and so on.

3.4.5 MIT Process Handbook pattern model
The Process Handbook (PH) has been founded on a set of key patterns that support the business process redesign [16]. The primary goal of PH is to be a comprehensive framework for organizing large amounts of useful knowledge about business processes in a richly interconnected, consistent, and powerful way. The PH has been shown [17] to be useful in a variety of domains such as business process reengineering [16], business process automation [18] and software design.

PH is a large repository of business processes and available to the public over the web: http://ccs.mit.edu/ph. PH business processes can be used as patterns to help people in redesigning existing business processes, inventing new processes especially those that take advantage of information technology, and organizing as well as sharing knowledge about organizational practices.

3.4.6 Action Patterns in Business Process Models
Smirnov et al. [19] specified two classes of business process-related patterns: co-occurrence and behavioral action patterns. The co-occurrence action patterns capture sets of actions, which often co-occur together in business processes, ignoring any ordering relations between them, while behavioral action patterns describe how co-occurring actions are related to each other in terms of behavioral constraints. Their main contribution is a formal description of action patterns and an approach for identification of patterns in existing process model collections based on association rules mining. The mined action patterns can be used to suggest additional activities to the modeler during a modeling act.

3.5 Activity Patterns
Activity Theory has been introduced as a framework for Human Computer Interaction design and research [20]. Based on Activity Theory, the Unified Activity Management (UAM) project [21] aimed to define an organizing framework for supporting collaborative work around the concept of human activity by creating a unified activity representation, architecture, and user experience. According to UAM, an activity is any coherent set of actions a human takes towards some end, be it specific or vague. Activities
vary from small single-person tasks to large-scale collaborative projects. Further, UAM aims to embed best practices in activity patterns that guide people as they perform an activity and simplify access to the resources they need. Activity patterns are templates for repetitive activities, ad-hoc or not [22]. Moreover, an activity pattern formalizes the structure and content of an activity and the integration methods it depends on, thereby making it reusable as a template in future activities. By sharing activity patterns, users can socialize best practices and reusable processes. IBM has developed Activity Explorer part of IBM Workplace 2.6 in which a component that utilizes activity patterns has been integrated. With this component, activity patterns can be created manually by experienced users or by ‘cleaning up’ actual instances of activities that have been carried out successfully.

### 3.6 Interaction Patterns

In this section we describe several approaches that can be grouped under the title interaction patterns and which aim to capture optimal solutions to common interaction problems. The efforts that we reviewed cover patterns for service, collaboration and business interactions. Notably, interaction patterns have been applied to new collaboration domains, such as the Second Life Virtual Gaming Environment [23]; nevertheless, their examination is beyond the scope of our study.

#### 3.6.1 Service Interaction Patterns

The aim of service interaction patterns [25] is to present a collection of patterns for service interactions in collaborative business processes, where a number of parties, each with its own internal processes, need to interact with one another according to certain pre-agreed rules. The proposed patterns apply primarily to the service composition layer (i.e. orchestration, behavioral interface, and choreography) but also to lower layers dealing with message handling and protocols. According to Barros et al. the value of patterns lies in their independence of specific languages or techniques. In particular, the proposed service interaction patterns have been used to analyze the scope and capabilities of BPEL and to some extent, of related specifications such as Web Services Description Language and WS-Addressing. The focus of service interaction patterns is on capturing: (i) interactions between services and (ii) direct dependencies between interactions and internal actions (e.g. processing of an incoming message or preparation of an outgoing message).

Each service interaction pattern is made up of a description, a set of synonyms, examples, a discussion of the ‘forces’ (i.e. issues and design choices) involved, a discussion of how the pattern can be approached using existing web services technology and a discussion of related patterns and/or notions. The solutions are described at a high level of abstraction and do not include code fragments.

#### 3.6.2 Collaborative Interactive Applications Methodology

Molina et al. [26] use pattern-based techniques to design groupware systems. Their process for groupware design is guided by a methodology, named Collaborative Interactive Applications Methodology (CIAM) which adopts different viewpoints for creating conceptual models of collaborative systems. Molina et al. suggest the usage of patterns in the ‘inter-action modeling’ and in the ‘interaction modeling’ phases. The Inter-Action Modeling describes the main tasks (or processes), which define the group work (and collaboration states) in defined organizations while the Interaction Modeling covers all the interactive aspects of applications used. The patterns used in these phases cover aspects of the workflow between tasks along with the context in which the group works by means of the roles and the objects of the involved domain model. The detected patterns in this context imply generalizing repetitive behaviors related to work organization and action protocols. Molina et al. [26] introduce a notation for interaction modeling named Collaborative Interactive Applications Notation (CIAN). The models generated with CIAN specify the main tasks and processes that must be supported by the Computer Supported Cooperative Work system. In these models the workflow and its context is shown graphically by means of the roles and the objects of the domain model involved. CIAN diagrams consist of graphs whose nodes are the states and the arcs are the transitions among states.

#### 3.6.3 Interaction Patterns from Social Network Analysis

Dustdar et al. [27] introduce Social Network Analysis [28] as a method to detect interaction patterns. They apply three patterns (proxy, broker, and master-slave) from the software architecture domain in the domain of business process modeling, and build an algorithm capable to detect these patterns using the logs of Caramba, a process-aware collaboration system with ad-hoc process support [29]. One of the latest additions to the interaction patterns from social network analysis is the introduction of trust aspects in this context [30].

#### 3.6.4 Action Patterns in Virtual Collaboration

Biuk-Aghai et al. [31] have worked to obtain basic action patterns using data mining, facilitated by information visualization. The results of this work have been consolidated in a framework that presents all the components of the process derivation cycle, a model of information and patterns of virtual collaboration in virtual workspace systems, and a process derivation method, along with information visualization tools that have been developed to aid in this step.

To represent essential aspects of action patterns, a graphical notation called Extended MOO diagrams has been proposed. The notation is based on the MOO diagram notation proposed by Hawryszkiewycz [32]. It depicts action, subject, referent, and location of an action pattern.

### 3.7 Patterns for Collaborative Working Environments

The eAce project [33] studied the technical, organizational, economic, market, legal and regulatory factors for the adoption of new collaborative tools in business environments and implemented the Collaborative e-Working Environment platform for the collaboration and knowledge sharing of communities. During the project a multitude of collaboration patterns and collaboration pattern languages was collected from the literature, research projects and domain-specific scenarios. An ontology structure [34] was proposed that implements a ‘collaboration stack’ that identifies the various levels of abstraction, ranging from abstract collaboration patterns on the one side to collaborative services and communication technologies on the other side.
3.8 Knowledge Flow Patterns
Samikar et al. [35] proposed pattern-based knowledge workflows for enabling the automation of knowledge flows across an organization. This approach addresses the problem of orchestrating various collaboration, communication and information retrieval components such as search engines, recommender systems and chat rooms in order to satisfy the knowledge needs in corporations. They divide knowledge flow patterns into elementary patterns and advanced patterns. Elementary knowledge flow patterns are the most basic patterns of knowledge flow. All knowledge flows can be eventually decomposed into elementary knowledge flows. Advanced knowledge flows are common knowledge flows that occur frequently in e-business. They are further divided into user-based patterns, service-based patterns and management patterns.

The elementary knowledge flow patterns can be combined to form advanced knowledge flow patterns, i.e. User-based knowledge flow patterns that are initiated by a user and are composed of elementary knowledge flow patterns; Service-based knowledge flow patterns that refer to knowledge flows between a user and a machine-based service; and Management patterns that help analyze, monitor and control knowledge flows across an organization.

3.9 E-Business Patterns
IBM has introduced a pattern-based approach for e-business software design [36]. E-business patterns describe the relationships between users, the business organizations or applications, and the data to be accessed. This approach enables IT architects to implement successful e-business solutions through the reuse of components and solution elements from proven successful experiences.

E-business patterns follow a layered model that structures them according to their function into Business, Integration, Composite, Application and Runtime patterns. Business patterns describe the relationships between users, the business organizations or applications, and the data to be accessed. There are defined four basic business patterns according the corresponding archetypical business interactions (User-to-business, User-to-User, User-to-Data, Business-to-Business). Integration patterns are orthogonal to business patterns in the sense that they tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern. The two general integration pattern categories are Access Integration, which refer to the integration of several services through a common entry point and is for front-end or customer integration, and Application Integration, which refer to the integration of multiple applications and data sources without the user directly invoking them. Composite patterns represent commonly occurring combinations of Business patterns and Integration patterns. Composite patterns are made up of several individual Business patterns and at least one Integration pattern. Application patterns provide a conceptual layout describing how the application components and data within a Business pattern or Integration pattern interact. Runtime patterns define the logical middleware structure that supports an Application pattern. They depict the major middleware nodes, their roles, and the interfaces between these nodes.

3.10 Collaboration Patterns
Collaboration Patterns (CPats) have been introduced for describing reoccurring segments of collaborative work that handle knowledge-based collaborations in dynamically changing collaborative environments [37]. A CPat 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.

A model for collaboration patterns has been developed for formally describing the important aspects of a CPat (e.g. problem, solution, context, category, precondition, event triggers etc.). The specific approach took into account the aspects of collaboration states that may induce a recommendation for executing a specific CPat. One important element of a CPat description is ‘triggering event’ because CPats aim to be triggered automatically upon detection of relevant events generated within dynamically changing collaborative environments.

3.11 Pattern Languages & Ontologies
In reviewing pattern-based approaches, we noted that several approaches represent patterns using ontologies. For example, researchers have bootstrapped an OWL ontology based on PH patterns creating an OWLized version of the MIT PH [16]. Moreover, OWL has been used to formally specify the ontology of Activity Patterns [21]. Further, an ontology has been developed in OWL-DL to formalize CPats in a way that can be easily refined, updated, extended and instantiated [38]. In this section we present approaches that introduce and use other languages for representing patterns.

3.11.1 Pattern Neighborhoods, ConAn Engine
Arevalo [46] applies Formal Concept Analysis [39] detect and recognize recurring collaborations among software artifacts. Formal concept analysis is a branch of lattice theory that allows identifying meaningful groupings of ‘elements’ that have common ‘properties’. In order to achieve their goal, they take the source-code of an object-oriented application as the main information source and extract structural relationships between classes. With their method they manage to detect both classical and non-classical patterns in the different applications using simple structural relationships between classes, not limited to known design patterns but extended to any recurring collaboration between classes in the analyzed applications. They also establish relationships, called pattern neighborhoods, over detected patterns and enable the incremental construction of a pattern library.

3.11.2 Pattern Language Mark-up Language (PLML)
PLML is an XML-based pattern language for human-computer interfaces [40]. Some important PLML elements are: the ‘pattern id’ that declares the unique identifier of the pattern, the ‘problem’ element that describes the design situation that the pattern will address, the ‘context’ element that characterizes situations in which this pattern can be most naturally applied and the ‘forces’ element that describes the forces in the environment that use of the pattern will resolve. The ‘solution’ element, addresses the problem, and should generalize from the examples that the pattern contains. Usually, the solution is expressed in the form of an instruction. The ‘synopsis’ element acts as a summary of the pattern and ‘diagram’
communicates to the user of the pattern details that are more readily expressed and apprehensible in schematic form.

3.11.3 Pattern Relations in Collaborative Pattern Environment
Schuemmer [41] presented Collaborative Pattern Environment, a collaborative environment for assisting authors and readers in generating patterns languages. Collaborative Pattern Environment assists the author in two ways: first, it allows the author to represent the patterns in a semantic network, which helps to capture dependencies between the patterns. Second, it helps the author to keep track of unfinished parts of the pattern by indicating those parts in the graphical pattern map. Here, the concept of pattern language is used as a collection of related patterns that address a specific problem, not as a formal language. Schummer [41] built on top of PLML and elaborated on pattern relations. According to the authors pattern relations can be distinguished in either labeled edges between two pattern nodes X and Y or as relations between patterns and other artefacts.

3.11.4 GAMA, a pattern language for computer supported dynamic collaboration
Based on Collaborative Pattern Environment and the method for pattern language creation that it supports, a pattern language for computer supported dynamic collaboration named GAMA has been proposed [41], [42]. The focus of the GAMA pattern language is to provide design solutions to end users and software developers that help to build collaborative technology which supports the needs of teams in a frequently changing environment. The basic structure of the language is made up of three layers, each of which addresses different questions, namely: Questions that arise during the design of community support, Questions that address issues that relate to group support for specific tasks and Questions related to the base technology.

3.11.5 Task Pattern Markup Language
Gaffar et al. [43] introduced the concept of task and pattern models to leverage the process of task modeling. Task models are used to effectively represent complex software artefacts. A user task model specifies what the user does, or wants to do, and why. It describes the tasks that users perform using an application, as well as how the tasks are related to each other. In other words, it captures the user task and system’s behavior with respect to the task-set. Patterns describe generic reusable task fragments, which can be used to establish the task model. Patterns for task models contain variables, which can act as placeholders for each particular context of use. During the process of pattern adaptation, these placeholders will be replaced by concrete values representing the particular instances of the current context of use. In order to model task patterns in a usable manner, the Task Pattern Markup Language (TPML) was developed and proposed. TPML is an XML-based language that consists of five descriptor elements (Name, Problem, Context, Solution and Rational) and the pattern body containing the formalized solution.

3.11.6 An Ontology-based Model in Patterns for Usability Design
Hemninger and Ashokkumar [44] proposed an ontology based model for patterns used in software interface design named usability patterns. Usability patterns represent knowledge about known ways to design graphical user interfaces that are usable and meet the needs and expectations of users. The aim of this work is to present an approach using Semantic Web concepts that turn informal patterns into formal representations capable of supporting systematic design methods.

The OWL and OWL-DL ontology languages have been used to model usability patterns. The OWL standard defines a frame-based knowledge representation language with axiomatic constructs for logic-based expressivity. OWL includes vocabulary for describing properties and classes that support the construction of class taxonomies and relationships between class properties and class instances. They also developed a PatternCore meta-model that defines the original pattern properties that are included for all pattern definitions. Some of the properties include: ‘hasProblem’, ‘hasSolution’, ‘hasContext’, ‘hasRationale’ and ‘hasForces’. Describing pattern attributes using this formal medium facilitates automated inferences of relationships between patterns.

3.11.7 OMG UML Profile for Patterns
OMG specified a UML profile for patterns [45], which is part of the UML profile for Enterprise Distributed Object Computing. A profile in UML provides a generic extension mechanism for customizing UML models for particular domains and platforms. Profiles are defined using stereotypes, tagged values, and constraints that are applied to specific model elements, such as classes, attributes, operations, and activities. A profile is a collection of such extensions that collectively customize UML for a particular domain (e.g., aerospace, healthcare, financial) or platform. The OMG patterns profile defines how to use UML and relevant parts of the Enterprise Collaboration Architecture profile to express object models such as Business Function Object Patterns using pattern application mechanisms. A meta-model has been defined in three main model elements in order to describe a Business Pattern: Business Pattern Name, Business Pattern Package, and Business Pattern Binding. Business Pattern Names are to identify patterns defined with Business Pattern Packages and are used to invoke patterns with those pattern names. Business Pattern Package is used to specify patterns and handle them as design elements. Business Pattern Binding indicates applying patterns and also represents a parameterized collaboration.

4. DISCUSSION
In this section, we compare the twenty seven examined pattern approaches across ten characteristics (Table 1). The ten characteristics have been identified in order to cover important aspects relevant for collaboration.

The first characteristic is ‘Model’. A model is an abstraction of phenomena in the real world; with this characteristic, we portray whether each approach uses a formal or informal modeling mechanism for patterns or not. In table 1, we have used the following notation regarding the model characteristic: ‘T’, when an effort describes patterns by using only text; ‘S’, when a structured format was used (e.g. tabular format); ‘O’, when on ontology has been developed. We found that the majority of researchers and commercial developers use a structured descriptive mechanism (e.g. tabular format) for imprinting patterns. Just three efforts use only text for describing patterns (i.e. Business Process Patterns for BOF [13]) and consequently their automatic processing by information systems becomes difficult, while only six out of the twenty seven efforts present a complete model using both a structured descriptive mechanism combined with an ontology (e.g. CPats [37]).
We use the ‘Language’ characteristic for capturing the approach taken for encoding the description of patterns. We have used the following notation regarding the language characteristic: ‘C’, when a specific effort focuses on just collecting groups of patterns and presenting them in a directory; ‘F’, when a formal language is used that can be interpreted and used by information systems; ‘G’, when a graphical description of patterns is given, providing a visual aid to the facilitators or end-users of the collaboration. Five of the approaches examined do not provide a formal language (e.g. Thinklets [4]) while the majority introduces a formal language (e.g. Pattern Language Mark-up Language [40]) often based on an ontology (e.g. Activity Patterns [21]). In addition, less than half of the approaches use a graphical notation to give a visual overview of the designed patterns (e.g. Business Function Object Patterns [45]) to collaboration designers, facilitators and users. Fincher et al. [40] introduced a complete pattern language (PLML) accompanied by UML diagrams.

From our survey we have identified two distinct approaches used to categorize patterns: based on pattern objective/function and based on pattern granularity. The characteristic ‘Categorization’ is used in order to unveil the categorization approach followed. The following notation is used: ‘G’, when the examined effort presents a categorization of the patterns based on their granularity; ‘O’, when the examined effort presents a categorization based on the objective / functional level of the collaboration that patterns address; ‘No’, when no clear categorization is present. Categorization is considered valuable because it helps to identify patterns based on the objectives and the abstraction level of different collaborations. We found that in most approaches categorization is based on collaboration objective or function (e.g. Collaboration Patterns in Virtual Communities [5]). Seven approaches present a categorization based on pattern granularity (e.g. Usability Patterns [44]). Just four approaches try to adopt both styles of categorization (e.g. Patterns for Collaborative Working Environments [33]). We believe that categorization of patterns based on collaboration objective or function has added value because it can be used to further analyze patterns in run time and identify that two or more collaboration patterns present the same functional behavior. The latter can be the starting point for deciding which one of them can be used in a specific collaborative situation.

The capability to support ‘Collaboration’ is a key consideration of our survey. We have used the following notation regarding the ‘Collaboration’ characteristic: ‘M’ for approaches that have been developed with the primary aim to support collaboration; ‘S’ for approaches that address some aspects of collaboration; ‘No’ when collaboration cannot be supported in any way using the specific kind of patterns. The majority of the examined approaches consider collaboration as their main focus; in some cases collaboration may even span across organizational boundaries. Nine of them are considered to be able to support collaborations (e.g. Workflow Patterns [12]) but have not been motivated by the collaboration needs at least in their early phases of their work. So, in our view workflow and process patterns are a special case of collaboration patterns, given that workflow patterns are patterns which refer to the structured collaboration of humans and software systems. Even if not all collaborative problems within a collaborative environment can have prescribed solutions in the form of workflows, it is sensible to have at least a basic predefined structure for specific parts of collaboration support.

We have used the ‘Interaction’ characteristic to depict whether an approach supports human tasks, services or both. We have used the following notation regarding this characteristic: ‘H’ when an approach involves only human tasks in the collaboration framework; ‘S’ when an approach mainly involves web services. Most of the examined approaches focus on patterns on a service interaction level (e.g. Service Interaction Patterns [25]). Nevertheless, we have identified seven approaches that supplement efforts like Norta’s et al. [14] by focusing on human-to-human interaction (e.g. Patterns in CoPE [41]) without taking under consideration the services that may be involved in a collaborative work. Only six approaches try to cover human and service interactions simultaneously (e.g. e-Business Patterns [36]) in an effort to establish a complete supporting mechanism for facilitating both humans and services in the context of collaboration.

The ‘Target User’ characteristic focuses on the actual beneficiary of the pattern. We have used the following notation regarding this characteristic: ‘E’ when patterns are mainly developed / detected for assisting the collaboration participant; ‘D’ for patterns that assist the designer or the facilitator of collaboration. We found that the focus of the majority of the approaches is on the designer or the facilitator of the collaboration (e.g. Patterns for Inter-Organisational Business Processes [14]). Only five of the pattern approaches that were examined, focus both on the designer and the participant (e.g. CPats [37]).

We have used the ‘Contribution’ characteristic in order to portray the vision on how patterns should be used. We have used the following notation: ‘M’ when an approach aims to be used to mine / detect patterns by examining details of several collaborations; ‘E’ when the pattern is in an executable format by software systems. This execution results in the monitoring and orchestration of automatic or manual collaboration actions; ‘Ra’, when patterns serve as recommendation mechanisms for automatic actions; ‘Rm’, when patterns serve as recommendation mechanisms for manual actions; ‘Rs’, for patterns that recommend actions based on the interaction / collaboration state. The ‘contribution’ column of Table 1 reveals that the majority of efforts aim to support pattern mining as ‘guide of best practices’ or as comparative elements between the design and the actual collaborative work (e.g. MIT Process Handbook [16]). Only five approaches involve patterns that can actually be executed by some kind of engine (e.g., Workflow Patterns [12]).

Another characteristic that we have used in our analysis is ‘Distribution’, which captures the boundaries of collaboration. We have used the ‘Intra’ and ‘Inter’ keywords to characterize patterns that support collaboration taking place inside the borders of teams/enterprises/organizations and collaboration that spans across them. As seen in Table 1, most approaches cover interactions or collaborations that are located both inside a group or an organization and can also span these limited boundaries (e.g. Activity Patterns [21]). Just five efforts are focused mainly on intra-organisational/group interactions (e.g. Patterns in CoPE [41]).

The ‘Knowledge’ characteristic has been used to position efforts that their main focus (‘M’) is to acquire, distribute and manage knowledge based aspects of collaboration (e.g. knowledge flows, knowledge artefacts) with the use of patterns (that may use meta-models for capturing their differences, commonalities and relationships) or others that may support knowledge (‘S’) by making some additional assumptions or extensions (i.e. in their models or ontologies).
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Design Patterns, [2]</td>
<td>S</td>
<td>F</td>
<td>O</td>
<td>No</td>
<td>S</td>
<td>D</td>
<td>M</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Thinklets, [4]</td>
<td>S</td>
<td>C</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>D</td>
<td>Rm</td>
<td>Intr</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Collaboration patterns in knowledge management, [7]</td>
<td>S</td>
<td>C</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>D</td>
<td>Rm</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Event Patterns, [8]</td>
<td>S</td>
<td>F&amp;G</td>
<td>No</td>
<td>S</td>
<td>S</td>
<td>D &amp; E</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Workflow Patterns, [12]</td>
<td>S</td>
<td>F&amp;G</td>
<td>G&amp;O</td>
<td>S</td>
<td>S</td>
<td>D &amp; E</td>
<td>E&amp;M</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Process Patterns, [14]</td>
<td>S</td>
<td>C&amp;G</td>
<td>G&amp;O</td>
<td>S</td>
<td>S</td>
<td>D</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Patterns for inter-organizational business processes, [14]</td>
<td>S&amp;O</td>
<td>C&amp;G</td>
<td>O</td>
<td>M</td>
<td>S</td>
<td>D</td>
<td>Rm</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>11</td>
<td>Action Patterns in Business Process Models, [19]</td>
<td>S</td>
<td>F</td>
<td>O</td>
<td>S</td>
<td>S</td>
<td>D</td>
<td>M&amp;&amp;Rm</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>12</td>
<td>Activity Patterns, [21]</td>
<td>S&amp;O</td>
<td>F</td>
<td>G</td>
<td>M</td>
<td>H</td>
<td>E</td>
<td>E&amp;M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>13</td>
<td>Service Interaction Patterns, [25]</td>
<td>S</td>
<td>C</td>
<td>O</td>
<td>M</td>
<td>S</td>
<td>D</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>Collaborative Interactive Applications Patterns, [26]</td>
<td>S</td>
<td>G</td>
<td>O</td>
<td>M</td>
<td>S</td>
<td>D</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>15</td>
<td>Interaction Patterns from SNA, [27]</td>
<td>S</td>
<td>F&amp;G</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>D &amp; E</td>
<td>M</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>16</td>
<td>Action Patterns in virtual collaboration, [31]</td>
<td>S&amp;O</td>
<td>F&amp;G</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>D</td>
<td>M</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>17</td>
<td>Patterns for collaborative working environment, [33]</td>
<td>O</td>
<td>F</td>
<td>G&amp;O</td>
<td>M</td>
<td>H&amp;S</td>
<td>D</td>
<td>M</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>18</td>
<td>Knowledge workflow Patterns, [33]</td>
<td>T</td>
<td>C</td>
<td>G&amp;O</td>
<td>M</td>
<td>S</td>
<td>E</td>
<td>M&amp;Rm</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>19</td>
<td>e-Business Patterns [35]</td>
<td>S</td>
<td>C&amp;G</td>
<td>O</td>
<td>M</td>
<td>H&amp;S</td>
<td>D</td>
<td>Rm</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>21</td>
<td>Pattern Language Made-up Language (PLML), [46]</td>
<td>S&amp;O</td>
<td>F&amp;G</td>
<td>No</td>
<td>S</td>
<td>H&amp;S</td>
<td>D &amp; E</td>
<td>E&amp;Rm</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>22</td>
<td>Patterns in CoPE, [47]</td>
<td>S</td>
<td>C&amp;G</td>
<td>No</td>
<td>M</td>
<td>H</td>
<td>D</td>
<td>M</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>23</td>
<td>CAMA, [42]</td>
<td>S</td>
<td>F&amp;G</td>
<td>O</td>
<td>M</td>
<td>S</td>
<td>D</td>
<td>E&amp;Rm</td>
<td>Intr</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>24</td>
<td>Task Patterns, [43]</td>
<td>S</td>
<td>F</td>
<td>O</td>
<td>S</td>
<td>H</td>
<td>D</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>25</td>
<td>Usability Patterns [44]</td>
<td>S&amp;O</td>
<td>F&amp;G</td>
<td>G</td>
<td>S</td>
<td>S</td>
<td>D</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>26</td>
<td>Business Function Object Patterns, [45]</td>
<td>S</td>
<td>G</td>
<td>G</td>
<td>M</td>
<td>S</td>
<td>D</td>
<td>M</td>
<td>Intr/Inte</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>27</td>
<td>CP patterns [47]</td>
<td>S&amp;O</td>
<td>F&amp;G</td>
<td>O</td>
<td>M</td>
<td>H&amp;S</td>
<td>D &amp; E</td>
<td>E&amp;Rm&amp;Rm&amp;Rm&amp;Rm</td>
<td>Intr/Inte</td>
<td>M</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Comparative Review of Pattern Efforts
One third of the examined approaches (e.g. CPats [37]) is able to capture knowledge related to collaborations. We found that another one third of the approaches can support aspects of knowledge based collaborations (but it was not their main focus) using their proposed patterns. The remaining one third cannot integrate knowledge elements in their approach (e.g. Service Interaction Patterns 25)).

Finally, we have used the ‘Event-Aware’ characteristic for capturing whether the approach can be aware of the collaboration state by means of event detection and processing. Collaborative systems typically produce events that determine the collaboration state and might influence other services. The majority of approaches do not provide an event-aware mechanism. Only four of them take under consideration the change of states in order to recommend or detect a pattern (e.g. CPats [37]).

These ten characteristics that were used for this comparative review of pattern approaches were carefully selected for revealing and evaluating artefacts that can truly assist virtual, computer-mediated communities. In these dynamic collaborative environments (e.g. we evaluated artefacts that can truly assist virtual, computer-mediated communities. In such environments capturing best practices about recurring collaborative problems and solutions amongst dispersed groups becomes a necessity. According to de Moor [5] and Henninger [44] model, ontologies and formal languages are essential in order to capture and reason about patterns, hence we have used the ‘model’ and ‘language’ characteristics. The characteristics ‘interaction, target user and contribution’ are also considered critical [11], [37] for assisting us in evaluating the pattern approaches based on who they try to facilitate and under what scope and technology.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the results of a survey on pattern approaches, models and languages which are relevant to collaboration. We have detected two broad dimensions of work in existing efforts. The first dimension distinguishes between approaches that detect/mine patterns in order to identify differentiations from established best practices and propose corrective actions, mainly to the designer/facilitator, and approaches that aim to directly assist participants. The second dimension distinguishes between approaches that require manual intervention in order to be operable and ones that can provide automatic support to participants. Due to the rising complexity of the collaborative working environments (e.g. Virtual Breading Environments, Virtual Organizations etc.), we argue on shifting the attention towards directly assisting participants in automatic ways and developing new tools that can proactively recommend corrective actions in ongoing collaborations.

The survey has identified potentially fruitful research directions. The use of ontologies and taxonomies in collaboration patterns, as identified in our survey, provides the basic technological background that enables to capture knowledge about collaboration and reason about formalized patterns of collaborative work. Further research is required to develop and validate ontological structures of collaboration patterns. Such structures will allow not only the specification of sound and complete conceptual models but also enable the processing of rich social interactions and facilitate additional functionalities, such as mining and recommendation of appropriate collaboration tasks to allow the full exploitation potential of the social semantic web.

More and more, collaborative environments are developed in such a manner that their functionalities are provided as services to various service consumers. These services exploit new distributed computing paradigms, such as grid computing and cloud computing, to enable innovative pervasive and mobile collaboration scenarios and allow people to work across organizational boundaries and collaborate among organizations and communities. Collaboration patterns represent a fruitful research domain in SOA-based collaboration services and protocols and may be explored in efforts that address integration issues among different organizations in collaboration scenarios of virtual teams and/or communities.

6. ACKNOWLEDGMENTS

This work has been partially funded by the European Commission under project SYNERGY ICT No 63631.

7. REFERENCES


[33] eAce Project available online at: http://www.eace-project.org/


