Using Collaboration Patterns for Contextualizing Roles in Community Systems Design

Aldo de Moor

CommunitySense, Tilburg, the Netherlands

Abstract: Activation of collaborative communities is hampered by the communicative fragmentation that is at least partially caused by their distributed tool systems. We examine the role of domain, conversation, and functionality roles in modelling community activation. We show how collaboration patterns can be used to design appropriate socio-technical solutions. These patterns contextualize the various types of roles by linking them to the (1) relevant usage context (2) communicative workflow stages and (3) functionality components across the tool system.

Keywords: communities, activation, collaboration patterns, roles, socio-technical systems, design

Introduction

Collaborative communities are communities in which there are not only shared practices, but also common goals, such as the joint production of goods or services. These communities are increasingly important for getting work done in business, research, education, and other professional domains. For such communities to mature and perform, well-designed information and communication processes are needed. Typically, these processes require an evolving mix of ICTs over time (Stephens 2007). Supporting these communities, therefore, are increasingly distributed tool systems of (legacy) information systems, classical information and communication tools like word processors and mailing lists, and a rapidly expanding chest of social media tools like microblogging (e.g. Twitter), social networking sites (e.g. Facebook, Ning), wikis (e.g. MediaWiki) and social bookmarking tools (e.g. Delicious).

Many collaborative communities fail. Typically, after an initial boost of activity upon launch, they slowly wither, with the occassional outburst on a discussion board, if at all. Contrary to what many believe, this is often not caused by lack of motivation. The sustained success of many professional society websites/mailing lists and of course of the flagship social media project Wikipedia prove that motivation to contribute is often plentiful (Spinellis and Louridas 2008). Instead, the fragmentation of communicative acts across tool-system functionalities and the unclear collaborative context of this system results in a lack of activation. We define activation as supporting the initiation, execution, and evaluation of goal-oriented computer mediated communication processes in collaborative communities in order to increase the effectiveness and efficiency of their collaboration (de Moor 2008). This activation often fails due to insufficiently reflecting on how to support the initiation and evaluation stages of the communicative workflows (de Moor and Weigand 2007). To more precisely analyze such socio-technical problems and their possible solutions, we have a need for a socio-technical pattern language.

Technical design patterns, such as prominent in human-computer interaction and software engineering pattern languages, focus on interface, interaction, and implementation (Borchers 2000). Socio-technical patterns, on the other hand, have a broader, application domain levelscope. They are especially useful at beginning of social software projects to describe the complex nature of the interactions between the social and the technical systems that need to be built (Dixon 2009). In (de Moor 2009) , we describe how one class of these patterns, collaboration patterns, can be used as building blocks for the socio-technical design of community systems.

Roles are important constructs in communities, in which they have a structuring, coordinating and supporting function (Herrmann et al 2004). Communities have only few formal roles but often many dynamic, informal roles. Role development is key and takes place by perceiving the repetition of social interaction patterns based on patterns of expectations (Herrmann et al. 2004). In this paper, we expand on our previous work by exploring the “role of roles” in analyzing and addressing communicative fragmentation problems in collaborative communities. We illustrate how our collaboration pattern language can be used to more clearly perceive and analyze the socio-technical context of the required roles.

The next section outlines the role of roles in framing communicative fragmentation. We then introduce our collaboration pattern language and show how it can be used to model the context of roles. We end with discussion and conclusions.

Roles in Community Systems Design

The community system should provide a full range of communicative and collaborative functionalities (Preece 2000). Before examining these functionalities, we first reflect on the role of roles in the design of the socio-technical systems of collaborative communities.

Domain, conversation, and functionality roles

We distinguish three types of interrelated roles in collaboration patterns*:* domain, conversation, and functionality roles. This subdivision does not have to do with how “social” or “technical” the roles themselves are, but whether they represent a domain interest, do the communicative work, or use or maintain the enabling technologies.

- *Domain roles* are the roles somebody plays in the capacity of being a stakeholder member of a collaborative community. These roles can range from formally defined roles such as editor, reviewer, and author in a scholarly publishing community, to the numerous informal reader, contributor, collaborator, and leadership roles needed in any productive community (Preece and Shneiderman 2009).
- *Conversation roles* are the initiating, executing, and evaluating roles that community members play in their capacity of being participants in the network of communicative workflows making up the goal-oriented collaboration process (de Moor and Weigand 2007). This perspective on collaboration is grounded in the Language/Action Perspective which emphasizes what people do by communicating, how language is used to create a common basis for communication partners, and how their activities are coordinated through language (Winograd and Flores 1986).
- Both of the previous types of roles are used to model the requirements originating from the purposeful social system. *Functionality roles*, however, originate from the technical system and describe roles that people need to play in effectively using tool functionalities. In other words, they capture the *potential* interactions enabled by the tools. Instead of describing *The User* of a tool, they describe roles (as combinations of responsibilities, capabilities and permissions) that are needed to *best* apply tool functionalities in particular collaborative contexts. A good example is given by effective use-patterns of wikis, capturing lessons that were learnt by analysing many cases in which wikis are used in professional settings (Mader 2007). Some typical wiki-functionality roles include Champion, Contributor, Maintainer, WikiGardener, and WikiZenMaster.

Communicative workflow loops

Without effective communication, collaborative communities cannot work. At the heart of our approach, grounded in the Language/Action Perspective is the communicative workflow loop (de Moor and Weigand 2007). In theory, a successful communication loop consists of four communicative stages (plus a production stage), in which various pairs of conversation roles are involved:

- *Request*: the initiator (I) requests the executor (X) to do something
- *Promise*: the executor promises to do it to the initiator
- (*Production*: the executor does the job)
- *Report*: the executor reports to the evaluator (E) that the job has been done
- *Evaluate*: the evaluator checks the result and, if satisfied, approves of the job done.

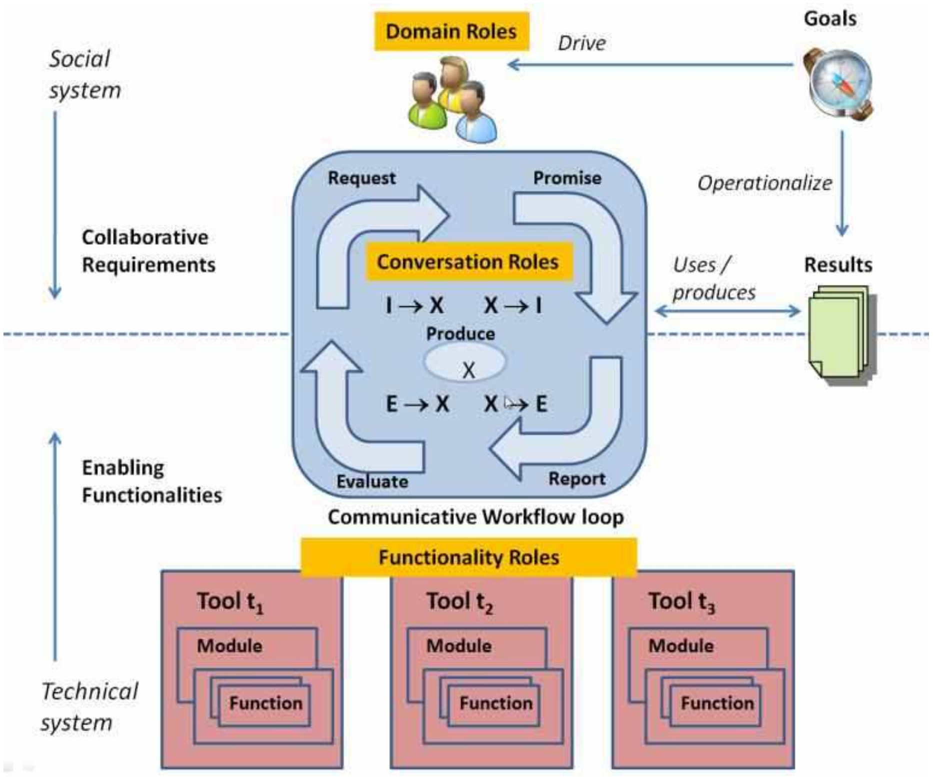

Figure 1 The Communicative Workflow Loop in context

This communicative workflow loop does not exist by and for itself, but is part of a larger direct context. Most importantly, it bridges the social system (generator of the collaborative requirements) and the technical system (providing the functionalities that enable these requirements). Starting point for analysis at the social system level are the *goals*, which drive stakeholders playing domain roles to work (together) on *results* that operationalise the goals. A *workflow* can have one or more of these results from previous workflows as its input, e.g. a draft document, and always produces at least one result, e.g. a copy-edited document.

From the technical system, the communicative workflow loop is grounded in a system of *tools*, which can be decomposed, insofar necessary, into *modules*, and individual *functions*. Together, this context describes an enabled communication loop, a "cell of productive interactions" which can be recombined to form much more complex and larger sets of "collaboration tissues" enveloping the collaborative community.

In practice, many complications occur: any of these stages can spawn new workflow loops, workflow loop stages fail, people often renege on their commitments, conflicts arise etc. However, as a lens for diagnosing messy reality and as a building block for community systems analysis and design, the workflow loop is a very important construct.

In (de Moor 2010), we examined how such contextualized workflow loops can be used as building blocks for complex social media systems design. In this paper, we are interested in how the various types of roles interrelate and in what socio-technical context they are situated. Ultimately, all roles are played by individuals or groups of human beings[i]. However, while from a social system/requirements point of view conversation roles are played by domain roles, from a technical system/enabling functionalities point of view the conversation roles are played by functionality roles. The challenge is how to meaningfully map domain role-instantiated conversation roles to enabling functionality roles. We take meaningful mappings to be contextualized role mappings that outline socio-technical solutions enabling goal-oriented communication, thus reducing the communicative fragmentation caused by distributed tool systems. Now, what does this socio-technical role context look like in systems design practice?

Contextualizing Roles with Collaboration Patterns

Collaboration patterns capture socio-technical lessons learnt in optimizing the effectiveness and efficiency of collaboration processes. They make enabled functionalities *actionable* by describing how community members playing particular collaborative roles use specific functionalities for particular purposes. They are grounded in community purpose patterns (defining the why, what, and who of the community, see e.g. the Liberating Voices pattern language[ii]) and functionality usage patterns (describing the effective use of individual tools, such as wikis, see e.g. the Wikipatterns language[iii]). Collaboration patterns, in turn, can be the source for more design and implementation-oriented pattern languages like used in software engineering and HCI (de Moor 2009).

We have grounded collaboration patterns in a simple ontology of the socio-technical system (de Moor and Weigand 2007; de Moor 2009). Its main elements are a *usage context* (the social system) and a tool system. Core entities in the usage contexts are *goals*, *actors*, (information) *objects* such as results, and *workflows*. The tool system consists of four levels of functionality components: the *system* level, the *tools* making up the system, *modules* constituting a tool, and finally, bundled in a module, the (information/communication) *functions*, which are the atomic units of functionality. *Functionality mappings* define relations within and between various elements of the usage context and the tool system. These mappings match required functionalities from the usage context with selected enabled functionalities from the tool system, in order to produce socio-technical design patterns. We can represent and reason about these mappings with conceptual graphs. One advantage of this semantic network formalism is one can use it to construct and analyze generalization hierarchies of graphs (Sowa 1984). Through so-called projections of more generic graphs (such as templates) into more specific graphs (such as the actual socio-technical system configuration in a particular community), one can match requirements with enabling functionalities. The resulting shortlist of possible solutions can then be presented to community members in a simple graphical or verbal notation, facilitating discussion about which socio-technical solution is optimal from the point of view of the community.

Collaboration patterns include goal, communication, information, task, and meta-patterns. In earlier work (see (de Moor 2009) for an overview), we gave a detailed description of our typology of collaboration patterns, the ontology to conceptualize the usage context and tool system, the socio-technical mappings between these subsystems, and the way in which to obtain, represent, and use these patterns to model the socio-technical system and its activation.

Communicative requirements and enabling functionalities meet in so-called *enabled collaboration patterns*. In Figure 2, we present a template of this pattern:

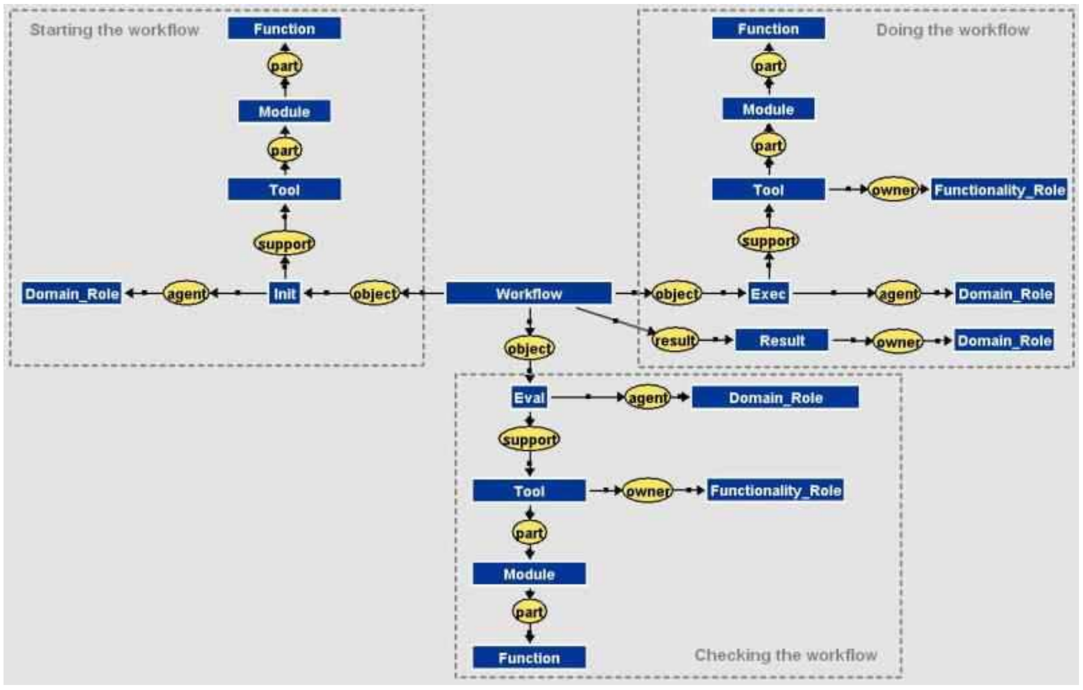

Figure 2 The enabled collaboration pattern template

The yellow ellipses indicate relations between concepts (the blue squares). The semantics of both concept and relationship types are formally defined in type hierarchies (which are not discussed here). Each workflow consists of three main parts: starting the workflow (consisting of both the request/promise communicative acts), doing the workflow (consisting of the productive act generating the result), and checking the workflow (consisting of the report/evaluate communicative acts). Each of these stages has at least one domain role attached to it, playing the core conversation role of that stage (initiator, executor, or evaluator, respectively)[iv]. Each stage is also supported by at least one tool, module or even function within a tool.

The complexity and rate of evolution of the interdependencies of goals, roles, workflows, and tool functionalities in real-world communities are often very high. Collaboration patterns help to capture these interdependencies at just the right level of specificity, adding specification details where needed, yet leaving degrees of freedom where possible. Typically, such patterns are selected and configured in a couple of iterations, from generic templates to concrete, community-specific patterns which can be used, for instance, for role, workflow and tool configuration and for documentation generation. We show such an iteration using the (hypothetical) scenario described in (de Moor 2009) of the socio-technical system of the ESSENCE (E-Science/Sensemaking/Climate Change) collaborative community[v]. We first present an enabled communication pattern describing a generic socio-technical approach to a report editing task. We then refine this pattern to tailor it to the specific needs of this particular community.

Scenario: Co-Editing a Report Consensus Section

The long-term goal of ESSENCE is to improve global climate change policy making through the wise use of sensemaking technologies[vi]. These are in this case web-based tools that help frame issues, the positions stakeholders take on these issues, and arguments pro and contra that they attach to these positions. Such tools have significant potential for mapping the very complex debate about causes and effects of, and policies needed to deal with climate change. However, just making these tools available is not going to make anything happen. They are used in an elaborate socio-technical context of collaborating stakeholders (consisting of organizations like governments, corporations, NGOs, research institutes, mass media and individuals like citizens, subject matter experts, and journalists) and tools (e.g. social networking sites like Facebook, wikis for collaborative document editing, (micro)blogs like Wordpress and Twitter to notify audiences of developments, mailing lists to communicate in work groups, and so on).

One goal of ESSENCE could be to produce high-quality reports on climate change. These reports are *neutral* in the sense that each report has a consensus section representing the *common* view of all stakeholders, yet also allowing for *alternative* position sections, where dissenters can describe their point of view. The wording of the consensus section is very important, as all stakeholders must agree to the claims made there. A relevant collaboration pattern from the pattern library is presented in Fig. 3. It captures details essential for the design of the part of the socio-technical system needed to accomplish this goal. It is grosso modo[vii] a specialization of the enabled communication pattern template presented in Fig. 2. Typically, such a pattern is used to describe reusable lessons learnt in a collaborative community, outlining the why, what, who, and how of the workflow.

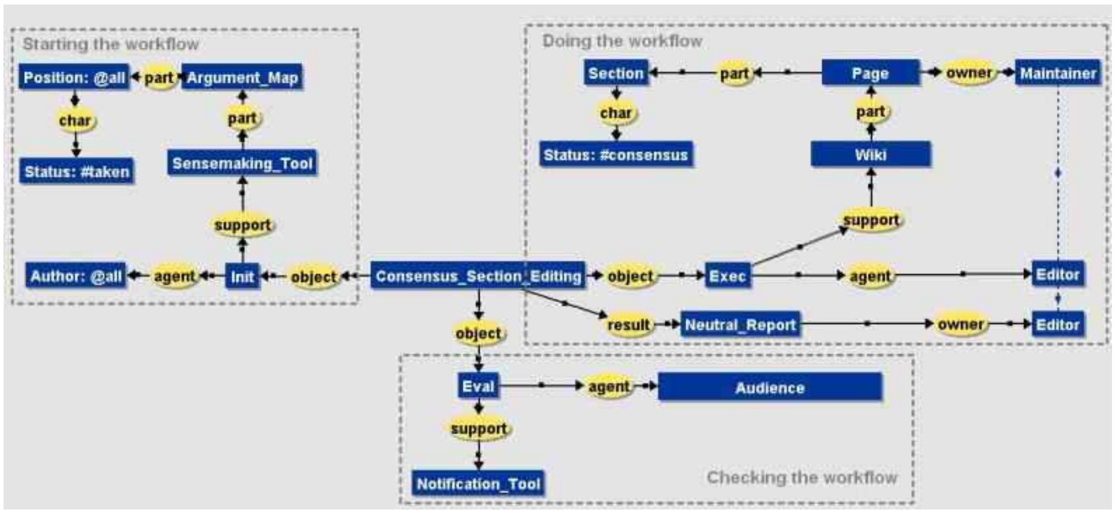

Figure 3 An enabled collaboration pattern for co-editing a report consensus section

In this case, it consists of a composite collaboration pattern comprising a communication pattern (modelling the Consensus Section Editing workflow), merged with a goal pattern (that a neutral report is to be produced), and two enabled functionality patterns (describing some characteristics of the enabling sensemaking tool and wiki, respectively). The owner of the report is the (domain role) Editor. This role can be played by multiple people simultaneously, but this is not modelled here.

- *Starting the workflow:* The consensus section editing process is initiated by the (domain role) Author. After all authors have taken all available positions in the argument map, the executor (i.e. Editor) responsible for doing the workflow is notified (this could be done, for instance, by the sensemaking tool automatically sending an e-mail with the link to the relevant summary map to the Editor upon the completion of the position taking).

- *Doing the workflow:* The ultimate goal of creating a neutral report is realized by the Editor(s) creating a consensus section on one of the wiki pages, which clearly delineates what positions are unanimous and which positions differ. A key element of role mappings is what in Conceptual Graph theory are called "lines of identity". These dashed lines link concepts, meaning that those concepts are played by the same individuals. In this case, the Editor who is the owner of the report is the same as the Editor who does the actual consensus section editing and is also the same as the (functionality role) Maintainer of the page on which the consensus section resides. The Maintainer role is a technical role with specific functionality privileges on the wiki, for example that this individual may lock the page preventing further editing after the report is finished.

- *Evaluating the workflow:* Finally, once the editing of the consensus section has been completed, the (domain role) Audience evaluates the outcome, supported by some notification and discussion tool (e.g. mailing list, RSS feed, Twitter). The Audience role is a composite role that could be made up of all (domain) stakeholder roles, which in turn can be played by many individual people.

The community decides this is a useful pattern to customize for its own specific context and further refines it for its system design and configuration purposes. Such a pattern we call an *implemented collaboration pattern* (fig.4).

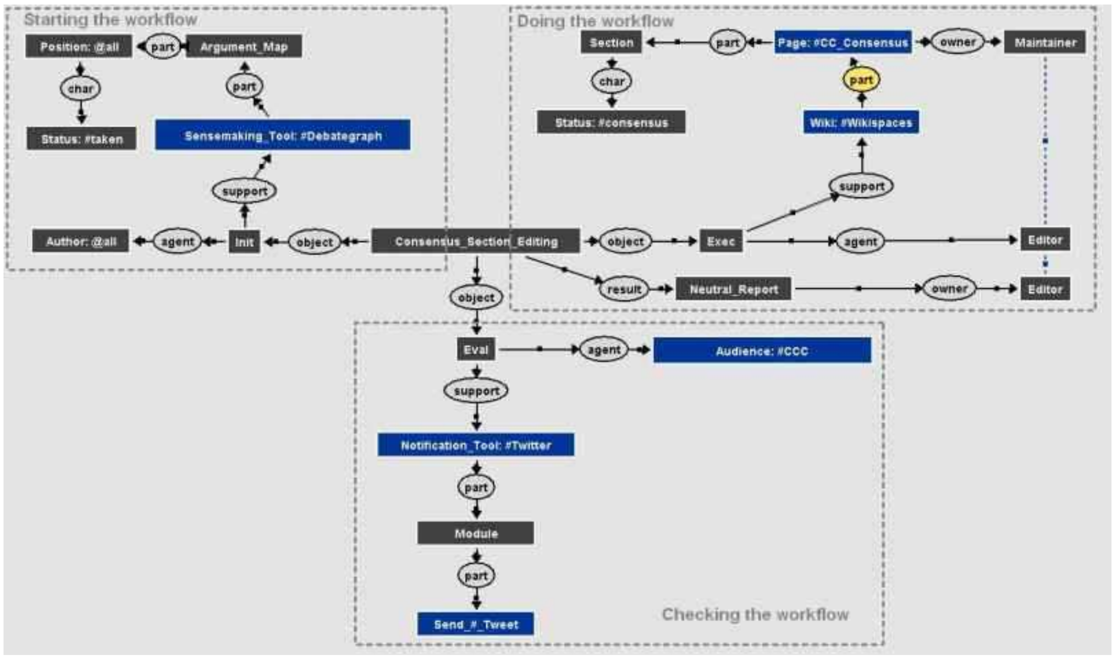

Figure 4 An implemented collaboration pattern for co-editing a report consensus section

The colored components are the element that changed when configuring the (generic) enabled pattern of Fig. 3 for the specific needs of this community. To produce the argument map, Debategraph is used, a state-of-the-art sensemaking tool[viii]. For a wiki, Wikispaces[ix] is chosen as it is hosted, reliable, and usable. A page called the "Climate Change Consensus Page" is created on this wiki. The audience in this case is the, largely uncharted Climate Change Community. To keep them in the loop, Twitter is selected as the supporting tool, as it allows for the easy reporting and receiving of feedback to and from strangers through its "hashtag (#) topics", which allow everybody searching tweets on that topic to be informed and reply[x].

Discussion and Conclusions

In this paper, we focused on the role of roles in our emerging collaboration pattern language. We examined how this language can be used to define the context of domain, conversation, and functionality roles in order to design better socio-technical systems for collaborative communities, and to reuse and configure the lessons learnt that the patterns capture.

Role development means the shaping of social interaction patterns through a set of role mechanisms such as role-taking (how a person acts with respect to the expectations of a specific role), role-making (how a person lives a role and how s/he transforms the expectations into concrete behaviour), and role-definition (defining the tasks and expectations associated with a role). Developing roles in a CMC setting needs a mediating tool system (Herrmann et al. 2004; Jahnke et al. 2005). Typical workflow modelling approaches, such as Business Process Modeling Notation (BPMN)[xi] often leave the exact communicative commitments and interactions of and between roles unclear, as well as how they are supported in this by the tool system. In contrast, our collaboration pattern approach provides socio-technically contextualized role definitions required for communicative workflows acceptable to and owned by the community. They can be used to - at just the desired level of detail - match the required social context with the enabling “functionality substrate” of the tool system in which these roles can play out. Of this, we showed only one example. Real-life collaborative communities, especially when operating mostly online and on a global scale need many more of such patterns. How to scale, connect, and make “meta-sense” of these patterns is still an open issue.

We only looked at one source of activation problems: the high thresholds for (inter)action caused by communicative fragmentation across tool systems. There are many other causes of activation problems, such as the attachment of members to the group as a whole as well as to individual other group members (Ren et al. 2007), their informational behaviours (Burnett and Buerkle 2004), and so on. Still, collaboration patterns can help to build and test design hypotheses for dealing with these other phenomena as well, as collaboration pattern entities like goals, roles, tasks/workflows, functionalities are core elements of these theories. Thus, these sociological theories may help come up with the empirical grounding informing which patterns would work in practice, while the collaboration pattern language can help to translate these findings into better systems designs.

It is important to realize that our approach goes beyond the typical CSCW (Computer Supported Cooperative Work) perspective that sees roles as mere mediating constructs offering a specific set of technical access permissions/restrictions to functionality or data. Collaboration patterns can help to reduce the socio-technical gap present in many of these systems, making it possible to specify the technical system implications of essential yet fuzzy social concepts like legitimacy, freedom, privacy, and democracy (Whitworth 2006). Furthermore, these patterns are not static technical specifications, but are meant as analytical instruments for *collaborative sensemaking*. The patterns need to be widely discussed in the community, for example along the lines of the methodology for socio-technical pattern elicitation and application proposed by (Dixon 2009), in order to clarify the often incompatible expectations about role definitions among different community members. Conceptual graph formalisms and tools exist to automatically match enabled with required patterns and to create verbal representations in controlled natural language. Such facilities would be very useful to present patterns to end-users in a form they can understand.

Collaboration patterns for role development can be applied in many different ways, for example in improving the selection, linking, configuration of tool systems through RBAC (Role Based Access Control)-mechanisms that should support the social needs of the community (Jahnke et al. 2005). The patterns can also be used to semi-automatically compose targeted, role oriented manuals and tutorials, instead of having to expose “the user” to the complete set of “context-free”, voluminous manuals. Using collaboration patterns to precisely tie domain, conversation, and functionality roles to one another and to their supporting tools, can reduce the communicative fragmentation caused by distributed tool systems, and thus help activate communities. Collaboration patterns then act as an interlingua: they can be easily made understandable to community members, making them owners of process change, while being exact enough for technical stakeholders like system administrators to appropriately design and implement the community systems.

We are working on a tool, CommunitySensor, which can be used to elicit, visualize and reason about socio-technical maps that provide the relevant context of communicative workflows, roles, and functionalities. Such tools could help empower communities to examine and resolve their collaborative breakdowns themselves.

People often wonder why we should put so much effort in the design of the socio-technical system of collaborative communities. Would not they best auto-magically organize themselves? This is akin to having many suppliers deliver loads of tiles, bricks, and planks to a construction site and hope that the construction workers will somehow make sense of those materials without having a drawing and plan for action designed by an architect. Socio-technical collaboration systems are similarly fragile and complex, with the smallest missing detail being able to have the whole system collapse. Thinking deeply about the design of such systems is hard work, but essential. In this paper, we hope to have contributed to thinking about new methodologies for building socio-technical houses that communities can and do want to live in.

References

Borchers, Jan O. 2000. “Interaction Design Patterns: Twelve Theses.” in Pattern Languages for Interaction Design: Building Momentum, CHI 2000, The Hague, the Netherlands, Apri 2-3, 2000.

Burnett, Gary, and Harry Buerkle. 2004. “Information Exchange in Virtual Communities: A Comparative Study.” *Journal of Computer-Mediated Communication* 9.

Dixon, D. 2009. “Pattern Languages for CMC Design.” Pp. 402-415 in *Handbook of Research on Socio-Technical Design and Social Networking Systems*, edited by B. Whitworth and A. de Moor. Hershey PA: Information Science Reference.

Herrmann, T., I. Jahnke, and K.-U. Loser. 2004. “The Role Concept as a Basis for Designing Community Systems.” in *Proc. of COOP 2004, Hyères les Palmiers, France, May 11-14, 2004*.

Jahnke, Isa, Carsten Ritterskamp, and TThomas Herrmann. 2005. “Sociotechnical Roles for Sociotechnical Systems: A Perspective from Social and Computer Sciences.” in *AAAI Fall Symposium, Roles: an Interdisciplinary Pespective*. London: AAAI Press.

Mader, S. 2007. *Wikipatterns*. Wiley.

de Moor, A. 2008. “Activating Online Collaborative Communities.” Pp. 97-108 in Proc. of the 5th International Conference on Action in Language, Organisations, and Information Systems (ALOIS 2008), Venice, Italy, May 5-6, 2008. University of Trento.

de Moor, A. 2009. “Collaboration Patterns as Building Blocks for Community Informatics.” in Proc. of the 6th Prato Community Informatics Research Network Conference, Prato, Italy, November 4-6, 2009.

de Moor, A. 2010. “Conversations in Context: A Twitter Case for Social Media Systems Design.” in *Proc. of I-SEMANTICS 2010, September 1-3, Graz, Austria*. ACM.

de Moor, A. 2009. “Optimizing Social Software System Design.” Pp. 273-293 in *Conceptual Structures in Practice*, edited by Pascal Hitzler and Henrik Scharfe. CRC Press.

de Moor, A., and H Weigand. 2007. “Formalizing the Evolution of Virtual Communities.” *Information Systems* 32:223-247.

Preece, J. 2000. Online Communities : Designing Usability, Supporting Sociability. Chichester ; New York: John Wiley.

Preece, J., and B. Shneiderman. 2009. “The Reader-to-Leader Framework: Motivating Technology-Mediated Social Participation.” *AIS Transactions on Human-Computer Interaction* 1:13-32.

Ren, Y., R. Kraut, and S. Kiesler. 2007. “Applying Common Identity and Bond Theory to the Design of Online Communities.” *Organization Studies* 28:377-408.

Sowa, John F. 1984. Conceptual Structures : Information Processing in Mind and Machine. Reading, Mass.: Addison-Wesley.

Spinellis, Diomidis, and Panagiotis Louridas. 2008. “The Collaborative Organization of Knowledge.” *Commun. ACM* 51:68-73.

Stephens, Keri K. 2007. “The Successive Use of Information and Communication Technologies at Work.” *Communication Theory* 17:486-507.

Whitworth, B. 2006. “Socio-Technical Systems.” Pp. 533-541 in *Encyclopedia of Human Computer Interaction*. Hershey: Idea Group.

Winograd, Terry, and Fernando Flores. 1986. *Understanding Computers and Cognition : a New Foundation for Design*. Norwood, N.J.: Ablex Pub. Corp.

Footnotes

[i] In reality, software agents can take over some simple, relatively context-free tasks. However, ultimately, human beings are responsible for what these agents do, which is crucial for efficiently assigning responsibilities and resolving collaborative breakdowns.

[ii] http://www.publicsphereproject.org/patterns/

[iii] http://wikipatterns.com/display/wikipatterns/Wikipatterns

[iv] Note that in the starting and evaluation stages, always two conversational roles are involved (initiator/executor and executor/evaluator, respectively), but only the main agent of that stage is represented here, as the other one can be deduced from the structure of the workflow loop.

[v] http://events.kmi.open.ac.uk/essence/

[vi] For instance: http://debategraph.org/, http://cohere.open.ac.uk/

[vii] Strictly speaking, it is not a complete specialization, as some module and function concepts plus their relations are not shown. However, theoretically they are still there, with “generic referents” (a conceptual graph notion, the details of which are not relevant here). To prevent clutter, we leave them out here as they do not add information.

[viii] http://debategraph.org

[ix] http://www.wikispaces.com

[x] See (A. de Moor 2010) for a more in depth treatment of this functionality.

[xi] http://www.bpmn.org/