The Role of Roles in Computer-mediated Interaction

Stephan Lukosch
Delft University of Technology
Faculty of Technology, Policy, and Management
PO box 5015, 2600 GA Delft, The Netherlands
s.g.lukosch@tudelft.nl

Till Schümmer
FernUniversität in Hagen
Department for Mathematics and Computer Science
Universitätsstr. 1, 58084 Hagen, Germany
till.schuemmer@fernuni-hagen.de

Abstract

Roles coin any social interaction. In this paper, we present basic practices for designing roles in collaboration settings. These patterns should help the designer of collaborative systems to reflect roles in the system design and thereby steer group interaction.

1 Introduction

The concept of a role is omnipresent in any interaction. This paper, e.g., has been written by two humans who took the role of the author. After one author created an initial draft of one section, the other author took the role of a devil's advocate. He questioned the theses of the first author and thereby helped him to clarify his point. When we submitted the paper, it was received by two people playing the role of a conference and a programme chair of EuroPLoP 2008. They checked the formal content of the paper and passed it on to a group of 10 people who were in the role of a programme committee. These people were expected to provide an assessment of the paper and judge whether or not it could be raised to a sufficient quality during a shepherding process. The programme committee

Proceedings of the 13th European Conference on Pattern Languages of Programs (EuroPLoP 2008), edited by Till Schümmer and Allan Kelly, ISSN 1613-0073 `<issn-1613-0073.html>`.

members passed their feedback back to the programme chair who released this paper for shepherding. At this point another group came into play: A pool of shepherds scanned this paper and decided whether or not they wanted to play the role of a shepherd. The shepherd's responsibility was to point out strong and weak points of the paper and help the authors to improve the weak points.

We could continue this story for one or two additional pages and thereby outline the interaction process that supported the evolution of this paper. From the example, we can already identify the core of the role concept: *A role combines prototypical behavior, rights, capabilities, and obligations.* Compared to this, tasks are expected activities. They are related to roles in workflows: From a rather abstract viewpoint which is sufficient for this paper, a workflow describes a sequence (or network) of tasks and relates them with roles. When enacting a workflow, the roles will be filled by concrete users who are then obliged to perform the task in a given time frame.

At the beginning of the process of our example, the author is expected to behave in a way that he writes the text of the paper, he has the right of expressing his thoughts. This however implies cognitive capabilities (e.g., the ability of formulating sentences or the capability of synthesizing new lines of thought) as well as technical capabilities like the access to a word processor or pen and paper. Finally, the author has the obligation of delivering the text in time with a sufficient quality.

The above example shows that a role owner has a different perception of the role as persons that are expecting specific activities from the role owner. From a philosophical perspective, Mead (1934) explored the two sides of the self, the 'me' as the social self and the 'I' as a response to the 'me'. In addition to Mead's theory, Cronk (2005) points out:

> There is a dialectical relationship between society and the individual; and this dialectic is enacted on the intra-psychic level in terms of the polarity of the 'me' and the 'I'. The 'me' is the internalization of roles which derive from such symbolic processes as linguistic interaction, playing, and gaming; whereas the 'I' is a 'creative response' to the symbolized structures of the 'me' (i.e., to the generalized other). (Cronk 2005)

This makes clear that there is always a dialogue between the role that contributes to the "me" and the "I" that constitutes the specific moment and the actions. Depending on the context of the self, there can be more or less need for creativity. If the human participates in a *strict workflow* or *production workflow* (Borghoff and Schlichter 2000) as we know it from *workflow management systems*, creativity is not desired in the participant's response. System design for such interaction should thus codify roles and restrict the capabilities of the user to actions that contribute to the expected behavior.

Looking at real collaboration scenarios, however, often shows a different style of interaction. Bardram (1997), e.g., investigated collaboration workflows in hospital settings and showed that plans are often created in-situ. Patients and doctors adapt their behavior so that it fits with the current situation. Environments for ill-structured tasks that are difficult to describe independently of current actions and that require agile decisions about future actions, often rely on social roles and at the same time provide all users with capabilities that extend the capabilities of their specific role. This allows participants to diverge from their pre-defined roles as needed. In the extreme case, technology support for roles is not required and coordination of expectations is supported by awareness mechanisms (e.g., a REMOTE SELECTION$_{\rightarrow \text{P4CMI}}$ that can be used to communicate on which artifacts a user is currently working).

In this paper, we present patterns for an intermediate understanding of the role concept: Designs in which roles are explicitly modeled while still providing space for divergence from the role (at least in some patterns).

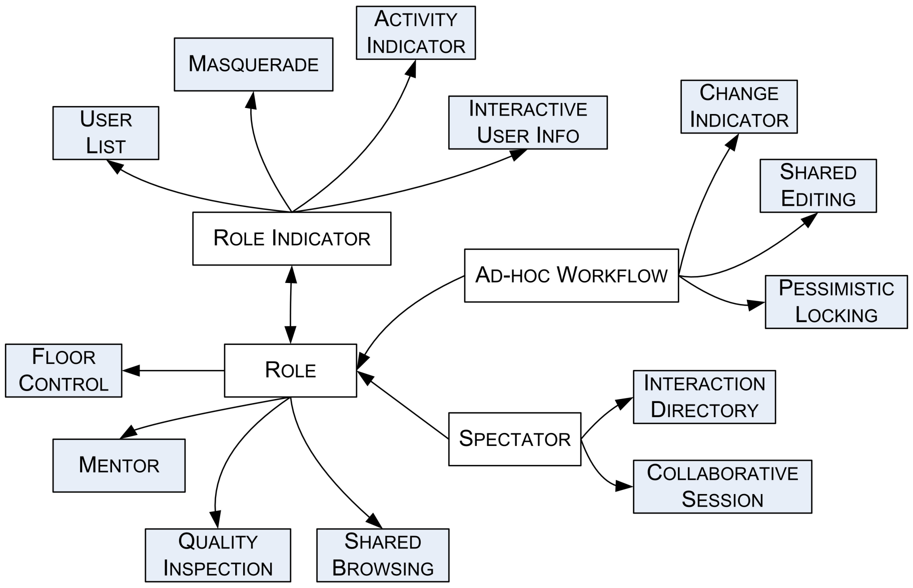


Figure 1: Pattern map

The patterns in this paper extend the 72 patterns for computer-mediated interaction that we present in (Schümmer and Lukosch 2007). Figure 1 shows the relations between the patterns in this paper and the patterns for computer-mediated interaction. The patterns for computer-mediated interaction follow the assumption that roles are part of the social agreement between the users of a collaborative system. In this paper, we frequently reference these patterns. Such a reference will be indicated by $_{\rightarrow \text{P4CMI}}$ next to a pattern name. Thumbnails of the referenced patterns can be found at the end of this paper. The four patterns in this paper are:

– ROLE: Explicitly model the responsibilities and capabilities of a role.

– ROLE INDICATOR: Decorate the VIRTUAL ME of a user with a symbol that represents the user's current role.

– AD-HOC WORKFLOW: Communicate dependencies between tasks and roles.

– SPECTATOR: Allow users to observe other users' actions.

While the first two patterns ease the understanding of the individual's capabilities and responsibilities, the third pattern supports the group's reflection on the interaction process. The last pattern finally helps to gain an outsider's view of the group process and is an example of a more concrete ROLE.

2 The Patterns

2.1 Role

Photo: Tyler, `http://www.flickr.com/photos/trp0/406897468/`

Intent

Model the expected interaction in the collaborative application.

Context

You designed a system for computer-mediated interaction that shall support a specific group process.

Problem

Users have problems to structure their interaction in the group. Especially, some users act in a way that is not anticipated by other users. This can hinder the interaction to reach the intended goal.

Scenario

John and Paul interact in a software project. They would like to do an XP session using an application sharing tool, but it is pure chaos. Both start typing text, both want to program as the inspiration enters their minds. They are in trance and totally ignore the presence of each other, as there is no group process which structures the interaction. As a result, no real collaboration takes place and numerous conflicts evolve.

Symptoms

You should consider to apply the pattern when . . .

– collaboration needs supervision and guidance.
– there are administrative tasks to do that require deeper understanding of the consequences.
– some users are unaware of the group process and do not behave according to the other users' expectations.
– users see features that they should not or cannot use.
– users repeatedly assign a comparable combination of access rights to different users.

Solution

Therefore: Define roles that describe what the owner of the role is supposed to do. Also specify in a role which tools may be used in order to reach the role's intended

goal. Link the roles to users when they engage in the group process.

Collaborations

Represent Roles in your collaborative system. Each Role has a description and is associated to activities that can be performed by the person having this role.

Users can perform all activities that are available in their roles and necessary to complete the assigned task. Before an activity triggered by the user is executed, the system checks if the user owns a role that is associated to the activity. If not, no action will be performed.

There are two special roles: the omnipotent role allows a user to perform any action (an example is the role of an administrator) and the empty role is not related to any actions (this is equal to no role).

Roles can contain other roles. By that, the set of possible roles is combined. Users can play more than one role at the same time which means that they can execute any action allowed in any of their roles.

Roles can be assigned to and withdrawn for a user. In order to influence the user's actions, the user has to see what role he plays. This can be done by means of a Role Indicator$_{\rightarrow 2.2}$ or by sending a message to the users whenever they should switch their role.

It is most common to assign a default user role to an account at the moment the account is created. A system administrator can assign roles with advanced rights to user accounts afterwards (Quick Registration$_{\rightarrow \text{P4CMI}}$). The system administrator role is assigned to a user account at the moment of system installation.

Rationale

The explicit notion of a role helps the users to understand their current situation in the group process. Since the role carries a description that explains what is expected from the person playing the role, it can help the users to fit their actions with the role.

The connection between role and action explicitly defines how a user can reach the role's goal. Assigning a user to the role ensures that all actions connected to the role can be executed by the user performing the role.

Assigning the allowed activities for a role instead of specific users reduces the amount of time spent on tool administration.

Check

When applying this pattern, you should answer these questions:

– What is the default or minimal role a user must have to act in the system?

– What role hierarchy should be implemented?

– Who is the authority to manage the association of roles to users? Can users pass roles on?

– What are the events that give rise to a modification of the role?

Danger Spots

There are many reasons why users may fail to fill their role. They may be absent because of illness or they may lack competencies required to act in this role. For such cases, the system has to provide means for reassigning the role to another user.

Users may also abuse the power that is given by the role. Again, the system needs to provide mechanisms to revoke the role from such users.

Especially in creative processes, roles cannot be pre-defined. The definition of roles may be impossible at all, though ROLES might emerge from interaction implicitly. The AD-HOC WORKFLOW$_{\to 2.3}$ pattern discusses the role of roles in such contexts.

Known Uses

Scripted Exercises in CURE (Haake 2007): In the context of computer-supported collaborative learning, scripts have been used to guide students through the exercises. The students were asked to first brainstorm concepts learned in the course, cluster the material, and finally write an essay on the topic. The essay writing process was supported by two roles: The author was asked to create a draft of an essay. Then, the other group members took the role of a reviewer and annotated the initial essay. After receiving the reviews, the author could modify the text or pass his role on to another group member and become a reviewer instead.

Blackboard: In Blackboard (Blackboard Inc. 2009), each user role has a specific set of permission levels:

– Course Builder - has access to all features except Assessments and Course Tools.

– Grader - has access to the grade book and is able to create and modify assessments.

– Instructor - has access to all course functions. This includes adding and modifying content, controlling user and group functions, creating assessments and entering grades, and controlling discussion boards and virtual classroom functions.

– Student - has access to all course content but cannot modify content. This is the only role able to take assessments and have grades recorded in the grade book.

– Teacher Assistant - shares the same level of access as the instructor. Although the rights are the same, the name of the role conjures different expectations regarding the teacher's behavior.

The admin assigns a new role to the selected user account.

Many other CSCL systems work the same way, e.g. Moodle (Moodle 2009), ILIAS (ILIAS 2009), and Synergeia (Stahl 2002).

World of Warcraft (`http://www.wow-europe.com/`) is a distributed multi-user game in which teams can play together against other teams. To initiate a team, one user invites other users to the team. The inviting user will have the role of a team leader and has special rights such as inviting new players to the team or deciding on how the goods are shared among the team members.

SourceForge.net is an open source software development web site in which project members can have different ROLES, e.g. administrator, developer, translator, etc.

Related Patterns QUALITY INSPECTION$_{\rightarrow \text{P4CMI}}$, MENTOR$_{\rightarrow \text{P4CMI}}$, SHARED BROWSING$_{\rightarrow \text{P4CMI}}$ are examples of patterns that rely on roles. These patterns focus on the group process and show how specific user roles can be supported.

FLOOR CONTROL$_{\rightarrow \text{P4CMI}}$ describes how roles can be passed on to other users.

ROLE INDICATOR$_{\rightarrow 2.2}$ shows which role a user currently has.

ROLE-BASED ACCESS CONTROL (Schumacher et al. 2005) discusses the issue of access rights in relation to roles.

2.2 Role Indicator

Photo: Ron Bird, FreeDigitalPhotos.net

Intent

Let each member of a group know which Roles the other members have.

Context

You are developing a computer-mediated environment, where users can have different Roles with different rights.

Problem

Users are not aware of capabilities as well as responsibilities of other users.

Scenario

Consider a globally distributed development project in which a large team co-constructs a game engine together with some test users. Molo, one of the African test users, has problems installing the new version of the game engine. Unfortunately, the water supply simulation game which he uses for testing does no longer work on top of the newest game engine. He would have liked to talk to the test officer in the project, but he does not know who this person currently is.

Symptoms

You should consider to apply the pattern when ...

- Users want to perform a specific activity but are not allowed to do this.
- Users frequently ask someone else to perform specific activities.
- Users do not know who has enough rights to perform specific activities.
- Users treat other users as if they have a different role.

Solution

Therefore: Visualize the Role$_{\rightarrow 2.1}$ of the interacting users whenever a user is shown in the user interface.

Collaborations

Integrate an element in the User List$_{\rightarrow \text{P4CMI}}$ or the Interactive User Info$_{\rightarrow \text{P4CMI}}$ so that the current role of a user is revealed. Make sure that the role representation is unique and can be understood by all interacting users. When users can change their role, update the role information whenever a user switches to another role.

Rationale

As each user's role is visualized, users can easily lookup their own role in the interaction process and also identify the role of their peer users. This allows users to interact with each other according their roles.

Check

When applying this pattern, you should answer these questions:

– What are the different ROLES?

– Where are you going to visualize the ROLES?

– How are you going to visualize the different ROLES? Are you going to use different icons for each ROLE or will you use textual labels?

Danger Spots

Users might not want that their role is revealed. In such cases, you should allow users to turn their role indicator off.

Users might have different roles at the same time. This makes it difficult to decide which role is shown to the other users.

Known Uses

World of Warcraft (`http://www.wow-europe.com/`) requires that each team has one leader. The leader has special rights, e.g. a leader can decide how rewards are distributed in the team or a leader may invite new team members. Leaders can be identified in the USER LIST$_{\rightarrow \text{P4CMI}}$ as their representation is associated with a small crown (cf. Figure 2).

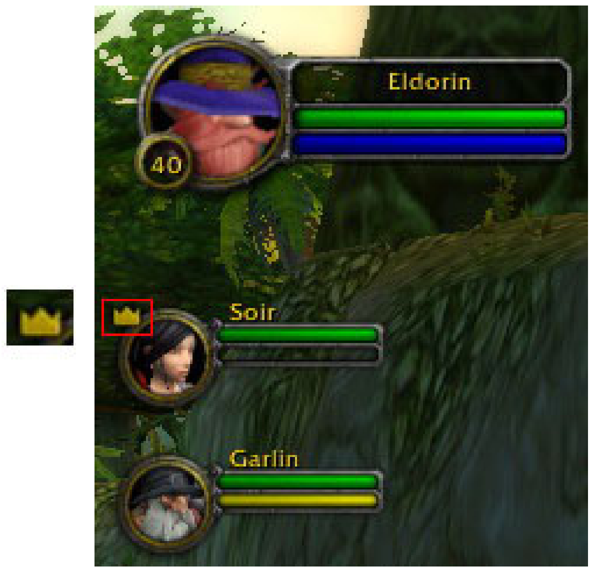
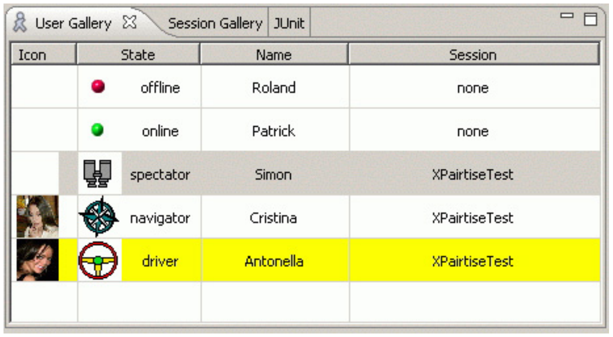


Figure 2: ROLE INDICATORS in World of Warcraft

Vitero `http://www.vitero.de` is a conferencing system which supports up to two moderators. The moderators can pass a microphone icon to user which want to talk. The microphone is shown to all other users as well so that they stay aware of who has currently the speaker role.

XPairtise (Lukosch and Schümmer 2007) is a tool for distributed pair programming. For supporting and teaching distributed pair programming, XPairtise distinguishes three different roles, i.e. navigator, driver, and Spectator$_{\rightarrow 2.4}$. Figure 3 shows how the different roles are indicated in the User List$_{\rightarrow \text{P4CMI}}$.



Icon	State	Name	Session
	● offline	Roland	none
	● online	Patrick	none
	spectator	Simon	XPairtiseTest
	navigator	Cristina	XPairtiseTest
	driver	Antonella	XPairtiseTest

Figure 3: Role Indicator in Xpairtise

Related Patterns Role$_{\rightarrow 2.1}$**:** Role Indicator describes where and when to visualize different Roles.

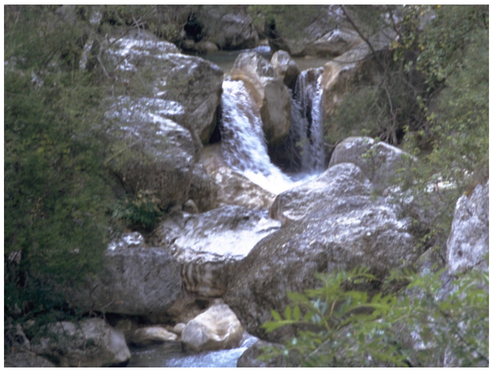
User List$_{\rightarrow \text{P4CMI}}$ allows to visualize different Roles by simply extending the user representation in the list.

Masquerade$_{\rightarrow \text{P4CMI}}$ describes how users can control what kind of personal information they reveal to other users. This can include information about a user's Role.

Activity Indicator$_{\rightarrow \text{P4CMI}}$ shows for awareness purposes the activities of the collaborating users. By analyzing the activities of a user, it is also possible to identify a user's role in the collaboration process.

Interactive User Info$_{\rightarrow \text{P4CMI}}$ equips a user representation with a context menu that allows to start an interaction with the represented user. It can easily be used to visualize the user's current Role.

2.3 Ad-hoc Workflow



AKA	Interaction Script
Intent	Communicate and make explicit dependencies between tasks and roles.
Context	You are interacting in a creative, ill-structured group process.
Problem	**Plans are a good thing. Having well-defined roles and tasks creates safety in performing tasks. However, ill-structured group processes are highly non-deterministic which means that they cannot be pressed in pre-defined task schemas. Plans will fail in most of these projects.**
Scenario	Consider a typical XP project. Linea, the customer created a set of task cards and arranged them in a lovely sequence. She created a picture of the project and believes that the team will be able to create a good solution by simply implementing the tasks. But already after the first task was done, the developers feel that the plan does not fit the context and after her first tests with the resulting system, Linea also feels the need to change the plan.

Symptoms *You should consider to apply the pattern when ...*

- strict workflows are too strict for dynamic/creative group processes.
- users are not aware of the intended group process and their future tasks.
- users expect others to do their work.
- users do not understand the dependencies between tasks.

Solution **Therefore: Collaboratively create an explicit representation of tasks as a shared document and thereby achieve a shared understanding of ad-hoc plans that dynamically adapts to the current group process.**

Collaborations In the simplest form, the group members can use a wiki to list the different tasks and responsibilities. For synchronous collaboration, the group members can also make use of a Shared Editor$_{\rightarrow \text{P4CMI}}$.

Users should be supported in creating shared tasks, creating relations between tasks, and assigning ROLES to tasks. Visualize the representation in a way that users can understand and perform the workflow. In case of a textual representation the workflow follows the linear structure of the text, e.g. each task is a list item. In case of a visual representation, the workflow is represented as a directed graph. Use the visual workflow to document current steps and guide the group process but allow the group to adapt the process as soon as it is necessary.

Rationale

Wil van der Aalst et al. (1999) described the taxonomy of collaborative work as shown in Figure 4. They classify collaborative work along two dimensions: structuredness and the center of attention.

Work can be highly structured as it is the case in optimized production workflows or it can be inherently unstructured as it is often the case for collaborative problem solving activities. The support for the group interaction can focus on making information available to all group members (and providing the best comprehensible awareness on the group members' activities) or it can focus on supporting the interaction process and thus guide the group members through the required steps.

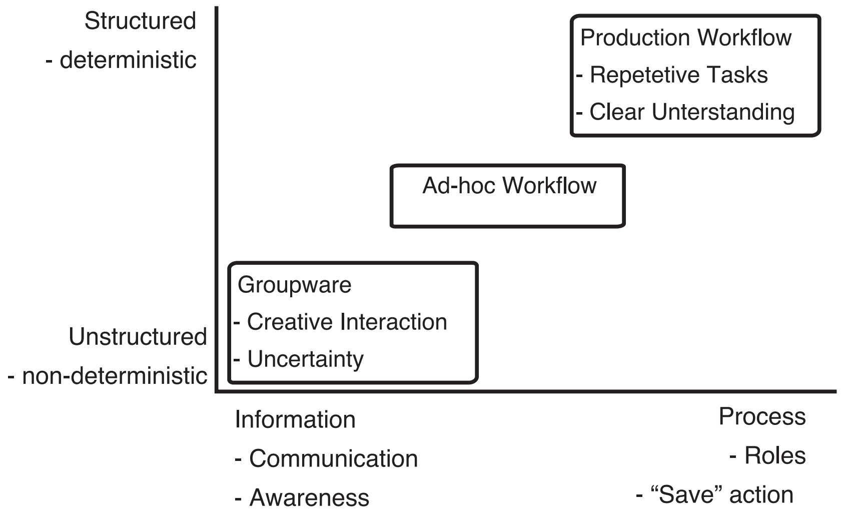


Figure 4: Situating collaborative work according to the structure and focus dimensions (inspired by van der Aalst et al. (1999)).

Groupware applications are typically situated in the lower left corner of the diagram: they support creative interaction and help a group to interact on shared information spaces. They focus on improving the communication between the group members and make them aware of each others' actions.

Workflow management systems on the other hand focus on guiding the users through the process. The process is pre-defined and the group members are only required to perform their steps in the process. Communication is pre-structured and in most cases reduced to the communication acts required for executing the workflow.

An AD-HOC WORKFLOW helps to structure the implicit processes of information-centered unstructured interaction. During the co-construction of the Ad-Hoc Workflow, the group members become aware of the required steps for the specific tasks and coordinate their efforts.

During the enactment phase of the workflow, the group members document their progress in the process and thereby increase the awareness of the group's activities. Since group members are allowed to deviate from the AD-HOC WORKFLOW, they keep the flexibility of information-centered collaboration.

Check *When applying this pattern, you should answer these questions:*

- When will you create the workflow in your group process? Can you distinguish coordination phases from collaboration phases?
- Will you create a graphical or a textual representation of the workflow?
- How do you visualize active tasks?
- Can you use the workflow representation to track your work?

Danger Spots Workflows are often not just a a linear sequence. Especially for iterative processes, workflows may include iterations or parallel processing streams. The user has to be careful that no circular dependencies occur that may lead to deadlocks. In a deadlock situation, user A waits for user B to complete task 1 before A can start task 2. At the same time, user B waits for user a to complete task 2 before he can start task 1. In general, the groupware system should highlight potential deadlocks in the workflow. This can be done by checking the following deadlock conditions that are common knowledge in operating systems research and visualizing those tasks for which the conditions apply:

Mutual exclusion: the tasks assume that the performer of the task has exclusive access to a shared resource (see PESSIMISTIC LOCKING$_{\rightarrow \mathrm{P4CMI}}$).

Hold and wait: tasks require more than one shared resource. Once the performers have obtained a PESSIMISTIC LOCK$_{\rightarrow \mathrm{P4CMI}}$, they keep the lock and request another lock to complete the task.

No preemption: there is no way to force a performer of a task to give back *his* resources.

Circular wait: task have a circular dependency as outlined in the previous paragraph.

Changes to the workflow can place another burden on the users. Once the ad-hoc workflow was changed, the users have to understand the new group process and adapt their behavior to act according to their role. Changes to the workflow should thus be highlighted, e.g., by placing a CHANGE INDICATOR$_{\rightarrow\text{P4CMI}}$ on the changed sections.

Known Uses

Chips / XChips (Rubart et al. 2001) visualized ad-hoc workflows as graph structures. The users could define task graphs and relate tasks with roles. When enacting the ad-hoc workflow, the users can assign group members to roles and thereby express responsibilities of individuals for specific tasks.

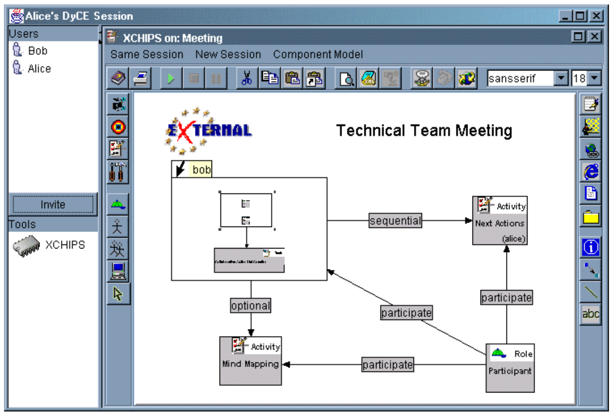


Figure 5: AD-HOC WORKFLOW in xChips

Figure 5 (Rubart and Haake 2003) shows how XChips was used to create an AD-HOC WORKFLOW for a company specific meeting (picture reprinted with authors' permissions).

DigiMod (`http://www.teambits.de/`) is a meeting facilitation support system that allows facilitators to structure the different phases of the meeting. Facilitators create tasks that describe what the participants should discuss and how the discussion should take place (e.g., as a structured brainstorming). During the meeting, DigiMod visualized the sequence of meeting steps and highlights the current task.

Related Patterns

ROLE$_{\rightarrow 2.1}$: Tasks are performed by ROLES. In the definition phase of the AD-HOC WORKFLOW, the group members name (or create) roles and associate them with tasks.

NO AGENDA, NO MEETING argues that all meetings should have an agenda. The agenda is comparable to a linear AD-HOC WORKFLOW.

SHARED EDITING$_{\rightarrow\text{P4CMI}}$ The workflow representation should be created using a shared editor. This allows all group members to participate in the creation of the workflow and contribute their views of an optimal problem solving path.

2.4 Spectator

Photo: Federico Stevanin, FreeDigitalPhotos.net

Intent
Allow users to observe the activities of other interacting users.

Context
Users are interacting in a computer-mediated environment to achieve a shared goal.

Problem
Users are interacting in a computer-mediated environment but are not familiar with the environment. These users perform activities which disturb the interaction and collaboration of other users.

Scenario
John and Paul have now managed to work in pair programming sessions. Now, their company is growing as it acquires more and more projects. The new employees are requested to work in pair programming sessions as well to ensure the high software quality, but they do not know how to do this. The manager requests John and Paul to teach the new employees, but their tool has only been designed to support one driver and one navigator in a pair programming session. Thus, John and Paul do not know how they can show the new employees how to interact in pair programming sessions.

Symptoms
You should consider to apply the pattern when …

- Unexperienced users are not accepted by experienced users.
- Unexperienced users disturb the collaboration of other users.
- Users want to communicate their experience with a computer-mediated environment but do not know how.

Solution
Therefore: Allow users to view and follow the interaction in an ongoing Collaborative Session$_{\rightarrow\text{P4CMI}}$ as Spectator. Ensure that these Spectators cannot influence the interaction.

Collaborations
Allow users to select an active Collaborative Session$_{\rightarrow\text{P4CMI}}$ from an Interaction Directory$_{\rightarrow\text{P4CMI}}$ and to choose whether

they want to join the session as a regular participant or as a SPECTATOR. When joining as SPECTATOR users can freely navigate in the shared artifacts which are used in the session. They can also view the activities of the other regular participants but they cannot influence the activities by modifying the shared artifacts as well.

Rationale The SPECTATOR ROLE does not allow that users influence the activities of other users. However, SPECTATORS can view and understand the interaction of other users and thereby learn how to interact in the computer-mediated environment.

Check *When applying this pattern, you should answer these questions:*

– Are you going to inform regular participants in a COLLABORATIVE SESSION about SPECTATORS viewing their interaction?

– Is informing the regular participants enough or should they in some cases be asked for explicit permission?

– Are SPECTATORS allowed to contact regular participants or other SPECTATORS?

– Are you going to provide mechanisms for SPECTATORS which make them aware of the other users' activities?

Danger Spots Ensure that SPECTATORS cannot access private artifacts of other users. You may even consider ROLE-BASED ACCESS CONTROL (Schumacher et al. 2005) to decide which artifacts SPECTATORS or other roles may access.

Known Uses **Counter Strike Source** (`http://store.steampowered.com/app/240/`) is a multi-user game in which teams compete with each other in COLLABORATIVE SESSIONS$_{\rightarrow \text{P4CMI}}$. Before participating in such a COLLABORATIVE SESSION, players have to decide which team they want to join. As additional opportunity, players can decide to join as a SPECTATOR. SPECTATORS can move around like regular participants but cannot influence the current game. For that purpose, players can switch their ROLE$_{\rightarrow 2.1}$ and become a regular participant.

Guild Wars is a MMORPG (Massively multiplayer online role-playing game) which apart role playing supports team competitions. The team competitions can be viewed by SPECTATORS.

XPairtise (Lukosch and Schümmer 2007) is a tool for distributed pair programming. Apart from the ROLES of a driver and navigator, XPairtise also supports SPECTATORS which can follow an ongoing distributed pair programming session. When joining a pair programming session, users can choose between the different supported roles (cf. Figure 6).

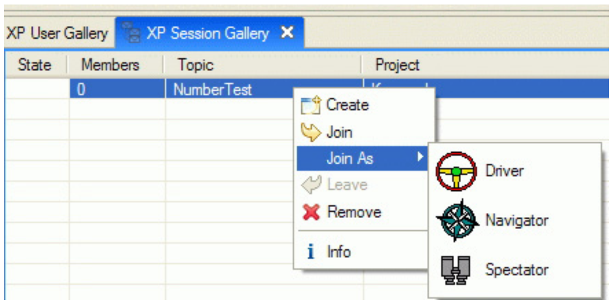


Figure 6: Role-dependent join in XPairtise

Related Patterns COLLABORATIVE SESSION$_{\rightarrow \text{P4CMI}}$: SPECTATORS can view the interaction in a COLLABORATIVE SESSION.

ROLE$_{\rightarrow 2.1}$: A SPECTATOR has a concrete ROLE.

ROLE-BASED ACCESS CONTROL (Schumacher et al. 2005) discusses the issue of access rights in relation to roles and can be used to define which artifacts a SPECTATOR may access.

3 Conclusions

The patterns of this papers discussed the role of roles in computer-mediated interaction settings. We presented a small selection of proven practices for modeling roles in such settings. However, we are aware of the fact that these patterns can only be a starting point towards a larger collection of practices that help groups to structure their group processes. It still needs to be investigated to what extent these practices can and should be written as patterns.

In some contexts, more domain-specific patterns have shown to be very helpful for practitioners of that specific domain. One example is the meeting patterns collection (Schuemmer and Tandler 2008) that names concrete roles in a group meeting, such as the role of the facilitator or the presenter. We foresee that more domain-specific pattern collections will emerge – and if there is no concrete collection for your specific domain, you may consider taking the role of an author and share your collaboration experience. May the patterns of this collection help you to structure your concrete patterns as well as concrete application that support your domain.

Acknowledgments

We would like to thank our shepherd Andreas Fießer for his excellent and challenging questions that helped to improve the patterns in this paper.

References

Bardram, J. E. (1997). Plans as situated action: an activity theory approach to workflow systems. In *ECSCW'97: Proceedings of the fifth conference on European Conference on Computer-Supported Cooperative Work*, Norwell, MA, USA, pp. 17–32. Kluwer Academic Publishers.

Blackboard Inc. (2009, February). Blackboard home. http://www.blackboard.com/.

Borghoff, U. M. and J. H. Schlichter (2000). *Computer-Supported Cooperative Work*. Springer-Verlag Berlin Heidelberg New York.

Cronk, G. (2005). George Herbert Mead – The Internet Encyclopedia of Philosophy. http://www.utm.edu/research/iep/m/mead.htm.

Haake, J. M. (2007). Computer-Supported Collaborative Scripts: Einsatz computergestützter Kooperationsskripte in der Fernlehre. In *DeLFI 2007, 5. e-Learning Fachtagung Informatik*, pp. 9–20.

ILIAS (2009, February). ILIAS open source LMS. http://www.ilias.de/.

Lukosch, S. and T. Schümmer (2007, September). Enabling distributed pair programming in Eclipse. In *10th European Conference on Computer-Supported Cooperative Work (ECSCW'07), Workshop 'The Challenges of Collaborative Work in Global Software Development'*.

Mead, G. H. (1934). *Mind, Self, and Society.* The Chicago University Press, Ltd., London.

Moodle (2009, February). Moodle.org: open-source community-based tools for learning. http://moodle.org/.

Rubart, J., J. M. Haake, D. A. Tietze, and W. Wang (2001). Organizing shared enterprise workspaces using component-based cooperative hypermedia. In *HYPERTEXT '01: Proceedings of the 12th ACM conference on Hypertext and Hypermedia*, New York, NY, USA, pp. 73–82. ACM.

Rubart, J., W. W. and J. M. Haake (2003). Supporting cooperative activities with shared hypermedia workspaces on the www. In *Alternate Track Proceedings of WWW 2003.* MTA SZTAKI.

Schuemmer, T. and P. Tandler (2008). Patterns for technology enhanced meetings. In *Proceedings of EuroPLOP'07*, Konstranz, Germany. UVK, Konstanz.

Schumacher, M., E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad (2005). *Security Patterns.* Chichester, UK: Wiley.

Schümmer, T. and S. Lukosch (2007). *Patterns for Computer-Mediated Interaction.* John Wiley & Sons, Ltd.

Stahl, G. (2002, September). Groupware goes to school. In J. M. Haake and J. A. Pino (Eds.), *Groupware: Design, Implementation, and Use, 8th International Workshop, CRIWG 2002*, LNCS 2440, La Serena, Chile, pp. 7–24. Springer-Verlag Berlin Heidelberg.

van der Aalst, W. M. P., T. Basten, H. M. W. Verbeek, P. A. C. Verkoulen, and M. Voorhoeve (1999). Adaptive workflow-on the interplay between flexibility and support. In *International Conference on Enterprise Information Systems*, pp. 353–360.

Appendix: Pattern Thumbnails

Activity Indicator

Problem: Users need time to perform a task but only the results are shared among them. In a collocated setting users are accustomed to perceive non-verbal signals such as movement or sounds when another user is active. If the users are distributed, these signals are missing. Users are therefore not aware of other users' activities, which can result in conflicting work or unnecessary delays.

Solution: Indicate other user's current activities in the user interface. To reduce interruptions, use a peripheral place or a visually unobtrusive indicator.

Change Indicator

Problem: While users works on independent local copies of artifacts, their check-out frequency for the artifacts may be low. As a result, they may work on old copies, which leads to potentially conflicting parallel changes. The conflict is worse if two parallel modifications have contradictory intentions.

Solution: Indicate whenever an artifact has been changed by an actor other than the local user. Show this information whenever the artifact or a reference to the artifact is shown on the screen. The information should contain details about the type of change and provide access to the new version of the artifact.

Collaborative Session

Problem: Users need a shared context for synchronous collaboration. Computer-mediated environments are neither concrete nor visible, however. This makes it difficult to define a shared context and thereby plan synchronous collaboration.

Solution: Model the context for synchronous collaboration as a shared session object. Visualize the session state and support users in starting, joining, leaving, and terminating the session. When users join a session, automatically start the necessary collaboration tools.

Floor Control

Problem: Synchronous interaction can lead to parallel and conflicting actions that confuse the interacting users and makes interaction difficult.

Solution: Model the right to interact in the shared collaboration space by means of a token and only let the user holding the token modify or access the shared resources. Establish a fair group process for passing the token among interacting users.

Interaction Directory

Problem: Finding existing contexts to start interaction and memorizing older contexts to continue an interaction is difficult.

Solution: Provide a shared space that is available to all users in which users can store and retrieve interaction contexts.

Interactive User Info

Problem: Users are aware of other users in the collaboration space and can identify them, but they don't know how to start tighter interaction with a specific user.

Solution: Equip the user representation with a context menu that provides commands for finding out more information on a user and for starting tighter collaboration with the user.

Masquerade

Problem: Your application monitors the local user. The information gathered is used to provide awareness information to remote users. While this is suitable in some situations, users often do not act as confidently if they know they are monitored. Users may feel a need to avoid providing any information to others.

Solution: Let users control what information is revealed from their personal details in a specific interaction context. This means that users must be able to filter the information that is revealed from their personal information. Remember to consider reciprocity.

Mentor

Problem: Newcomers do not know how community members normally act in specific situations. They are not used to practices that are frequently applied in the community.

Solution: Pair newcomers with experienced group members who act as mentors. Initially let newcomers observe their mentors, and gradually shift control to the newcomer.

Pessimistic Locking

Problem: You want to ensure that changes performed by the user are definitely applied, even if more than one user wants to modify the same shared object at the same time.

Solution: Let a site request and receive a distributed lock before it can change the shared state. The lock can have different grain sizes. The grain size of a lock determines how much of a shared data object, or of all shared data objects, can be modified after getting one lock. After performing the change, let the site release the lock, so that other sites can request and receive it for changing the shared state.

Quality Inspection

Problem: Members participate in a community to enjoy high-quality contributions from fellow members. However, not every contribution has the same quality. Low-quality contributions can annoy community members and distract their attention from high-quality gems.

Solution: Select users as moderators and let them release only relevant contributions into the community's interaction space. Give moderators the right to remove any contribution and to expel users from the community.

Remote Selection

Problem: Users select artifacts to start an action on the artifact. Selecting an artifact is considered as taking the artifact under personal control. Whenever two users select the same artifacts, this leads to coordination problems.

Solution: Show remote users' selections to a local user. Make sure that other users who are interested in a specific artifact are aware of all distributed co-workers who have selected the object.

Shared Browsing

Problem: Users have problems finding relevant information in a collaboration space. They often get lost.

Solution: Browse through the information space together. Provide a means for communication, and collaborative browsers that show the same information at each client's site.

Shared Editing

Problem: Users are sharing data for collaboration. The need to edit the shared data simultaneously emerges, but the shared single-user application does not allow concurrent editing.

Solution: Provide a shared editor in which users can manipulate the shared artifacts together. Ensure that state changes are instantly reflected in all other users' editors, and provide mechanisms that make users aware of each other.

User List

Problem: Users do not know with whom they do or could interact. Consequently, they do not have the feeling of interacting in a group.

Solution: Provide awareness in context. Visualize who currently is accessing an artifact or participating in a Collaborative Session$_{\rightarrow\text{P4CMI}}$. Ensure that the information is always valid.

Virtual Me

Problem: In a large user community, account names look similar. But users need to communicate their identity in order to interact with other users.

Solution: Allow the users to play theater! Provide them with means to create a virtual identity that represents them while they act in the system. Show the virtual identity when the user is active.