Groupware for Collaborative Tailoring

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
des Fachbereichs Informatik der
FernUniversität in Hagen

vorgelegt von
Lic. Inform. Alejandro Fernandez

Hagen, April 2005

To Analia and Magdalena.

Acknowledgments

Now, while I make the last corrections to this document, I look back and try to summarize the past years. It was February 2001 when we (my wife and I) landed in Frankfurt airport. We had clear goal, getting *our* Ph.d. I just did the research and the writing; she did everything else. Of course, we received a little bit of help (well, maybe more than just a bit).

I thank my advisors, Jörg Haake and Adele Goldberg for their continuous support and guidance.

Thanks Jörg. You are a great boss and advisor. I wish I ever acquire your ability to explain the most complex concepts with a simple drawing. I wish I ever have your patience with students and colleagues and your ability to see the best in them and help them develop it. Your a great motivator.

Adele, what can I say... I never imagined I could meet you, and suddenly you became my fairy godmother. I guess it is partly *your fault* that I got this far. You planted the seed of my work (present and future) when you pointed me to Donald Schön's work on Reflection in Action, Hutchins work on Distributed Cognition, Richard Gabriel's work on Software Habitability, and Peter and Trudy Johnson-Lenz definition of groupware. I am the luckiest student on Earth to have you as my teacher, mentor and friend. "Thanks" will never be enough.

Anyone who says that Germans are cold and distant, has not met *my Germans*. Of course they don't go around dancing, kissing and hugging. They simply take you with the families, without much noise, and make you feel at home. Thanks to the Schuckmanns, the Schümmers (both branches), and the Tandlers. Thanks to the Kässefondue troupe, to the IViewers and the Go4groupers. I regret of only one thing: I should have taught them more Spanish.

Working in the Concert division of Fraunhofer IPSI as been an experience I will never forget. Concert taught me the real meaning of the term collaborative work, with and without computer support. Thanks to all of your for the many hours of reflection about team work that nurtured my work. I miss the talks we had in the balcony or by the espresso machine.

Although I was many kilometers away, I never felt far from LIFIA. Gustavo, thanks for helping me with my lack of self-confidence. Alice, thanks for you visits in Darmstadt. They helped me feel part of the team. Fede, Richard, Diego, Diego and Lea, thanks for keeping my chair in the groupware group warm.

Thanks to my sister for taking good care of my parents while I was abroad. Thanks to my family in law for letting me take their Precious away.

Abstract

In everyday work, teamwork in the presence of the tools, the resources, and the processes that enable work is mostly transparent to the workers. They center their attention on performing work. However, a noticeable change in the work conditions, in the required quality of the product, or in the perceived results of work, may be experienced as a breakdown that brings teamwork to the center of attention. To deal with breakdowns it is currently common practice to include tailoring facilities in groupware systems. The extent to which these facilities are provided, and the way in which they are implemented, determine the power users have to change the groupware system. Determining these facilities has been the focus of most research on tailorability in CSCW. How *collaborative tailoring* (defined as, collaboration for and in tailoring) can be facilitated remains as yet undetermined.

This thesis tackles the problem of the lack of computer support for distributed team members that need to perform tailoring in the context of teamwork. The challenge of tailoring in the context of teamwork is to understand and support the needs of the group members, from the moment they encounter a breakdown during work until they have enacted the changes they deem necessary.

This thesis is based on the premise of participation as a means to achieve acceptance of change. The approach to support collaborative tailoring of teamwork presented in this thesis consists of a method for collaborative breakdown handling, a selection of specific groupware tools to be used for the deliberation activities defined by the method, and guidance in the form of scaffoldings for the application of the method. Breakdowns can also occur during tailoring. To deal with breakdowns that occur during tailoring, the method, the tools, and the scaffolding can be tailored. The proposed support for collaborative tailoring of teamwork is delivered as a stand-alone groupware system for collaborative tailoring. The system can be deployed along existing groupware systems, thus extending them with support for collaborative tailoring.

This thesis exceeds related work by approaching tailoring of teamwork as a social system with a model that explains tailoring as the result of collaborative breakdown handling. The requirements of communication, collaboration, cooperation and coordination, and negotiation observed in the social system are supported by the corresponding technical system. The approach in this thesis is not limited to its application in a particular scenario or groupware system. The only requirement is that the target system/scenario can be tailored. The approach has been conceived to enable and support its own evolution as the result of its tailoring.

Contents

Chapter 1

Introduction

The goal of CSCW–Computer Supported Collaborative Work–is both to study how people work together using computers and how computers can be used to improve how people work together. Ultimately, CSCW researchers attempt to change work habits through the introduction of computer-based systems. These systems are typically called "groupware"[38]. Their invention is a challenge because the very nature of working together continually changes as a consequence of changing work needs, but also as a consequence of how the systems themselves tend to change work relationships and processes. Systems must themselves adapt to reflect the inevitable and often unpredictable differences between the requirements of support for collaborative work documented during analysis and the actual requirements. *Tailoring* refers to continued development of an application by making persistent modifications to it [30]. Tailoring is initiated in response to an application being inefficient or difficult to use. It can occur during installation, before use, or during use. It involves users and developers [42]. This thesis asks the question of how to design our systems to adapt to change, not automatically, but rather through the team's negotiated tailoring of its work processes, communication, resource management, and production of work results? Can we provide people working together with the ability to tailor their own teamwork and, in doing so, create effective tailored groupware?

In everyday work, teamwork in the presence of the tools, the resources, and the processes that enable work is mostly transparent to the workers. They center their attention on performing work. However, a noticeable change in the work conditions, in the required quality of the product, or in the perceived results of work, may be experienced as a breakdown that brings teamwork to the center of attention. That is to say, a change may drive workers to reflect about teamwork. Heidegger [1] demonstrated that, while performing work, the presence of objects and properties of the world is transparent to the worker. Only in the event of a breakdown do some of these objects and properties become visible and demand attention.

Workers' inability to understand what to do can also motivate reflection. Hettinga [32], in her research on evolutionary use of groupware, explores these triggers of reflection and labels them "breakdowns". A breakdown signals a mismatch between teamwork and what workers expect it to be. A breakdown

[1] See Winograd and Flores[76] for a discussion of Heidegger's work in the context of computers and cognition.

represents an opportunity for tailoring, a time when team members have the opportunity to make teamwork conformant with a new situation.

A breakdown sends a signal that is potentially perceived by many group members in different ways. *Whether someone* realizes the occurrence of a breakdown and *how* depends on each individual. How a person participates in work; how tools, resources, and processes support the work of each participant; and what each individual participant brings to work from professional experience, are all factors that affect the perception of a breakdown. Similarly, the changes that need to be made to solve the perceived problem will affect each worker's ability to do work differently. Negotiating on the basis of these differences contributes to an increased likelihood of acceptance of the changes. If people collaborate to agree on a change (both its *what* and its *how*) they are more likely to implement the change.

It is currently common practice to include tailoring facilities in groupware systems. Anders Mørch [42], et al., identifies three forms of groupware tailoring[2]: customization, integration, and extension. Customization is defined as the capability to modify the appearance of presentation objects or to change their attributes by selecting from a set of predefined options. Integration is defined as the capability to add new functionality without accessing the underlying implementation code. Extension is defined as the capability to improve the functionality of an application by adding new code. The extent to which these capabilities are provided, and the way in which they are implemented, determine the power users have to change the groupware system. Determining these capabilities has been the focus of most research on tailorability in CSCW. As it has been previously indicated, tailoring of teamwork requires collaboration. How *collaborative tailoring* (defined as, collaboration for and in tailoring) can be facilitated remains as yet undetermined. An approach that supports collaborative tailoring must, at least, provide the following capabilities:

- To be able to discuss and reach a shared understanding about the way work is currently done and the perceived problems;
- To be able to share and discuss ideas about how work may be done in the future to solve the perceived problems; and
- To be able to identify and agree on a course of action that takes the way work is done from the current to the desired situation.

These capabilities allow users to collaboratively reflect on teamwork; to collaboratively agree on changes in tools, resources, processes, or communication; and to collaboratively plan and execute changes. Figure 1.1 depicts the discussion so far.

Being able to tailor and having support for collaboration during tailoring does not complete the picture of collaborative tailoring. The expectations that motivated tailoring in the first place—to provide support that matches user's needs, and to provide support that evolves as these needs evolve—apply to the tailoring facilities themselves. Although it would be possible, for example, for the designers of a groupware system to provide an initial set of generic techniques for collaborative tailoring, they would hardly match the user's needs in the long

[2]The paper provides a comprehensive discussion regarding the different, and often contradicting uses of the term tailoring and of other related terms.

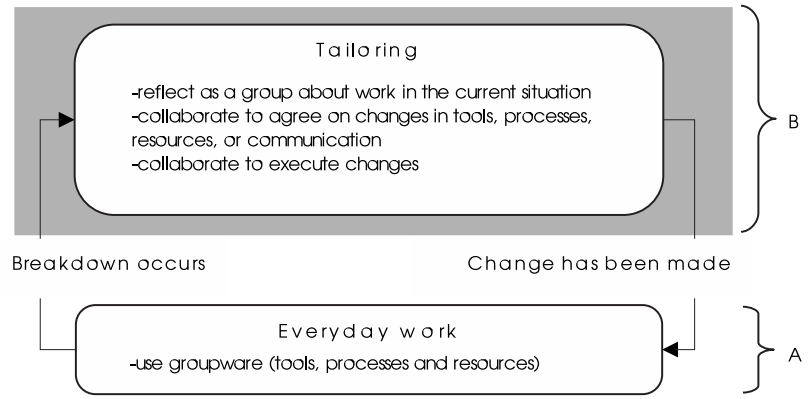

Figure 1.1: Tailoring teamwork

term. Firstly, successfully enabling users to discuss their work depends on the particular users, the way they participate in work, their professional experiences, and their preferences. Any one particular model of user structure, relationship and interaction can represent a source of problems (if not initially, certainly as the group structure changes with the incorporation of new members). Secondly, because as users learn to tailor, their expectations increase, and they may want to change their ways of tailoring. This change in needs renders the support provided by the system insufficient or at least inadequate.

Douglas Engelbart [23] studied the problem of organizational improvement and proposed a model that is applicable in this context. He identified three levels of capabilities. Level A comprises all capabilities that support the core business activities in an organization. Capabilities in level B support the activities that aim at improving capabilities in level A, notably, level B includes the capabilities for improving how work at level A is performed. Capabilities in level C provide a boost to this model by focusing on the improvement of the improvement capabilities. In this way, he points out that it is not enough to be able to improve work, but it is also necessary to be able to observe how improvement takes place, and then improve how to improve.

Figure 1.2 combines the previously presented view of collaborative tailoring with Engelbart's model. The activities in everyday work are supported by level A capabilities (these are the activities supported by traditional groupware). A breakdown that occurs in relation to an everyday activity (in level A) moves the context of discussion to level B. In level B, group members discuss ways to improve what happens in everyday work. Tailoring how work is done (this includes tailoring the groupware) is therefore a level B capability. Software to support the activities on level B takes the form of groupware to support collaborative tailoring.

At some point while working together to tailor, group members may realize that these collaboration facilities for tailoring are not enough or are inadequate (e.g., the process of tailoring they follow does not match the structure or needs of their organization). This realization brings the support for collaborative tailoring to the center of attention. If they are to continue tailoring work (changing

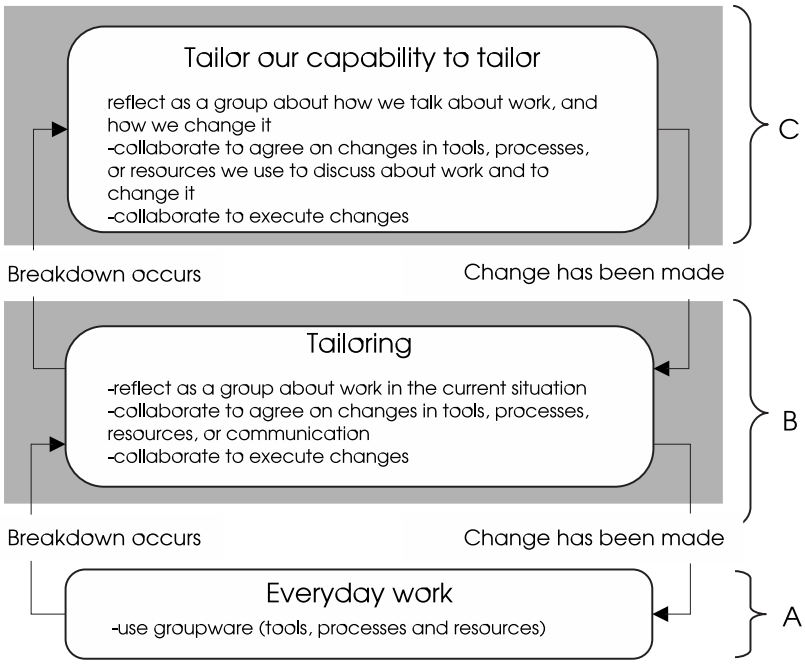

Figure 1.2: Collaborative tailoring in Engelbart's levels B and C

how activities occur in level A), they first need to get rid of the problems they encountered in level B. This breakdown takes them to level C, which in turn means they need to tailor how they tailor everyday work. The capabilities that were required to improve work become a requirement to improve the tailoring support:

- The capability to discuss and reach a shared understanding about how they currently tailor work, and about the perceived problems.
- The capability to share and discuss ideas about how tailoring may be done in the future to solve the perceived problems.
- The capability to identify and agree on a course of action that takes collaborative tailoring from the current to the desired situation.

Tailoring at level C may also be an activity supported by groupware.

Problem Statement

This thesis tackles the problem of the lack of computer support for distributed team members that need to perform tailoring in the context of teamwork. The challenge of tailoring in the context of teamwork is to understand and support the needs of the group members, from the moment they encounter a breakdown during work until they have enacted the changes they deem necessary. Any contribution in this area will act as a multiplier for the value of existing tailoring capabilities.

This thesis aims at the following contributions:

1. It proposes a conceptual view of collaborative tailoring of teamwork, centered on the existence of multiple perspectives and the occurrence of breakdowns.

2. It presents an approach to support collaborative tailoring of teamwork.

3. It provides a proof of concept of the recommended approach in the form of a groupware environment that supports collaborative tailoring of everyday practice according to the definition proposed here. In addition, the system supports collaborative tailoring of the team's tailoring practices and is itself tailorable. The approach hypothesizes two levels of reflection on how a team works together, and the exemplar groupware environment provides the context in which to apply work improvement practices and to then improve those very practices (thereby supporting effort at Engelbart's level C).

4. It validates the proposed approach through case studies of tailoring in real organizations.

Approach of This Thesis

This thesis is based on the premise of participation as a means to achieve acceptance of change. This premise and the context in which it is used are the focus of Chapter 2. Section 2.1 presents a scenario to illustrate where collaborative tailoring is found. The elements in the scenario, namely work processes, resources, tools, and communication are discussed in detail in Section 2.2. Section 2.3 is then a discussion of tailoring in the context of Engelbart's levels of improvement. The discussion leads to a set of capabilities that enable a group to tailor teamwork (documented as teamwork requirements).

Chapter 3 presents the state of the art. A discussion of what has being done regarding the development of tailorable support for teamwork situates this work in the context of related research on groupware tailorability. The picture of related work is completed with a review of existing research regarding the role of collaboration in tailoring.

As presented here, collaborative tailoring is another activity for the distributed work group. As such, it becomes the target for computer support. Chapter 4 presents a conceptualization of teamwork (targeting processes, tools, artifacts and communication) to support collaborative tailoring and a prototypical implementation of a groupware system that embeds this approach.

The lessons learned while defining, developing, and using teamwork for collaborative tailoring tell something about the design of groupware in general. These lessons are summarized before presenting conclusions and suggesting future lines of research.

Chapter 2

Problem Analysis

2.1 Scenario

The scenario documented in this Section describes a fictitious, distributed, software engineering organization. The core of the scenario consists of a description of breakdowns that the organization's members experienced. Section 2.2 characterizes teamwork in the context of similar organizations. Following, Section 2.3 recalls the scenario to present the conceptual view of collaborative tailoring that serves as the basis for the thesis, listing the requirements that the proposed groupware system needs to fulfill.

The scenario is motivated by the author's experience as a software engineer, and is based on experiences collected and documented by many others. Despite being imaginary, the breakdowns are inspired by the problems approached by published software engineering practices, such as Extreme Programming [12], and well-known organizational patterns so as to validate the scenario as realistic.

2.1.1 The Virtual Organization

Communication technology such as the Internet has been a booster for the emergence of virtual organizations [57]. Clustered Solutions is a fictitious virtual organization for the production of software for distributed work teams. Clustered Solutions acts as an umbrella for a cluster of existing organizations in the area of research, development, and consulting. Each of the participating organizations, Celsius, Globe, and Aconcagua, has a different strength that defines its role in the cluster. By being part of the cluster, the three companies increase their chances of success in the market. Figure 2.1 shows how Clustered Solutions is organized in a structure of teams from the three companies. Each team provides a core service or competence.

Celsius is a Swedish consulting company in the pharmaceutical industry. Celsius has produced software for many middle-sized companies in Sweden and Norway. During the last years, Celsius has observed a growing interest among its clients in forming strategic alliances with other pharmaceutical companies in Europe. This interest translates to an increasing number of requirements for software to support distributed teams. The role of Celsius as a member of the virtual organization is to act as a link to the market and as a domain expert. Celsius additionally contributes with a small (three-person) software

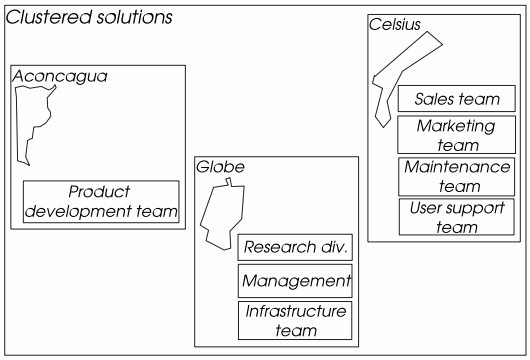

Figure 2.1: Team structure of Clustered Solutions.

development team devoted to software maintenance and support desk. Celsius takes care of marketing and commercialization.

Globe is a German research institute with focus on the conception, development, and evaluation of methods and software to support distributed work teams. Globe plays a double role in the virtual organization. First, Globe contributes with the method and the tools that support the function of the virtual organization. Second, Globe contributes with expert knowledge about the needs of distributed work teams, and with experience in supporting these needs with computers. Additionally, Globe takes care of managing the work of the virtual organization.

Aconcagua is a young software development company in Argentina. It originated as a spin-off of a research institute with a strong knowledge in object-oriented software engineering. In the past years, Aconcagua has provided offshore software development for many North American and European customers. Additionally, the economical situation of the country makes Argentina an attractive alternative for high quality, low cost software development. Aconcagua provides the core development force of Clustered Solutions.

2.1.2 The Product of Work

Clustered Solutions' strategy for product development is to exploit the commonalities of distributed work supported by computers across different domains. These commonalities inspire the development of generic services that are later customized to match the needs of particular customers. Depending on the amount of customization needed, the customization work is delegated to the core development team or to the maintenance team.

"Wisdom bricks" is a good sample of a system developed by Clustered Solution. It is a system to support training and continuing education in the workplace through mentoring and collaboration among peers. The strategy behind the development of the system is the creation of a set of reusable, self contained bricks that can be used to assemble a virtual company's training infrastructure. Figure 2.2 provides a coarse level overview of the systems' architecture.

The users have access to the system through a web-browser. The learning material and activities is accessible through a unifying portal. The main com-

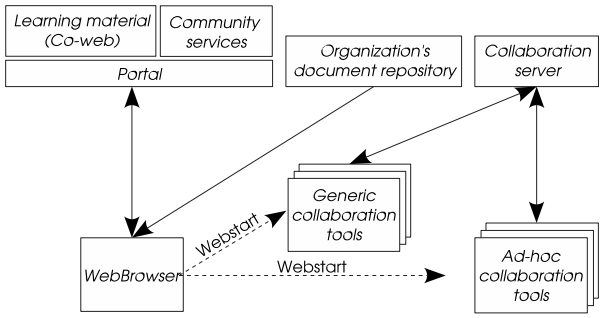

Figure 2.2: Coarse level overview of the systems' architecture.

ponents of the system are the portal, the repository of learning material, the community services, and the collaboration server.

The repository of learning material is provided as a collaborative hypermedia implemented on a variation of the WikiWikiWeb system [45]. The learning material is created and continuously enriched by expert trainers, mentors, and students.

Learners are teamed up with other learners. A mentor is assigned to each small team to serve as a guide through the material and exercises. Building up teams is one of the purposes of the community services subsystem. It additionally provides access to discussion forums, instant messaging, shared calendars, and shared file repositories. Mentors have the important role of showing how the learning material reflects the organization's history. To serve this purpose, learning material can include hyperlinks to documents from past projects, which are available from the organization's intranet.

A set of collaboration tools are made available to enable employees to learn at the workplace, working in tight collaboration with peers and with a mentor. The collaboration server serves the tools, which are started via links in the learning material via Sun's Java Webstart. Two types of tools are available, namely, generic collaboration tools such as a shared whiteboard and coupled web-browsing, and dedicated collaboration tools which are specifically developed to support collaborative learning of a particular concept.

The collaboration server is under constant development. New generic and dedicated tools are developed based on the demands of the customers and on the visions of the research division. A framework hides implementation details and provides developers with high-level abstractions that ease the creation of tools. The experience from the development of tools feeds the improvement of the framework that is periodically released.

2.1.3 A Work Process

The master plan of Clustered Solution dictates that the activities of the organization are to be planned on a half-yearly basis. Representatives from the three companies and from each of the work teams meet at the beginning of the period in a coordination meeting to agree on that period's plan. Work is organized in concurrently running projects in the areas of research and innovation, product

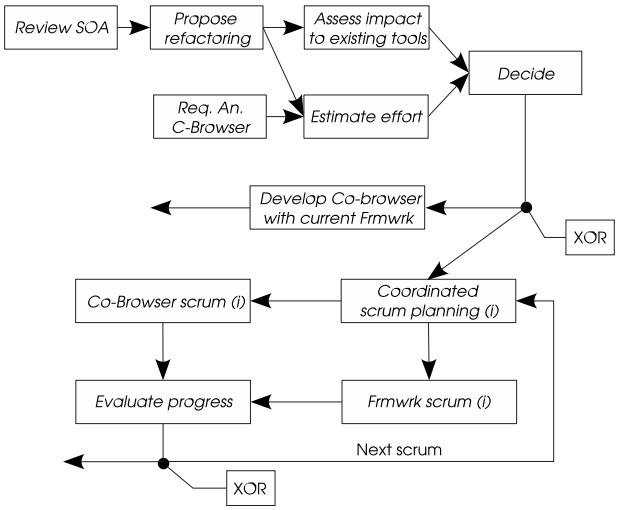

Figure 2.3: Action plan for the C-Browser project.

development, marketing, and improvement. During the coordination meeting, which can start as a virtual meeting weeks before the face-to-face meeting takes place, project proposals are developed, evaluated, and selected.

An accepted proposal created by the area of product development aims at the development of a generic coupled-browser, named the C-Browser, which was previously postponed. In addition, it proposes to use this opportunity to refactor the framework. The objective of the refactoring is to improve integration of collaboration tools in customer companies' existing groupware infrastructure. Figure 2.3 presents a coarse action plan. A review of the state of the art in groupware infrastructures commonly found in companies feeds a proposal for refactoring. The effort for the development of the C-Browser is estimated based both on the framework without refactoring, and on the expected characteristics of the refactored framework. The results of the effort estimation and an assessment of the potential impact that refactoring would have on existing tools are used to decide how to proceed with the project. If the effort/benefit ratio of refactoring the framework is not convincingly positive, the refactoring will be canceled and the C-Browser will be developed with the framework as it is. Otherwise, the development of the C-Browser and the refactoring of the framework will be conducted in tightly coordinated weekly cycles (i.e., Scrums).

Two developers from Aconcagua are assigned to the C-Browser. The creator of the framework is the only one authorized to change the framework.

2.1.4 Tools and Resources

Distributed teamwork in Clustered Solutions is supported by a set of traditional groupware tools. CVS, the version control system, is used to store all work documents as well as source code for systems developed by the organization. The infrastructure additionally provides a shared address book, discussion areas, instant messaging, calendar, and application sharing.

Interaction with the customer is usually performed via an issue report system. The issues documented with the system are a valuable resource for the customer support team, for the maintenance team, and for the development team.

The structure of the document repository was specified at a coordination meeting. The organization of the source files in packages and folders is fixed at the start of each project. The framework is a key resource for system development. It is kept in a separate module in the CVS repository. The structure of the framework module reflects the different subsystems and layers of the framework.

2.1.5 Breakdowns: a Driver of Change

The following breakdowns occurred during the execution of the project for the development of the C-Browser and the refactoring of the framework that was described in the previous sections.

Collective Code Ownership

It was decided that the refactoring of the framework would take place. For the last three weeks, the refactoring of the framework and the development of the C-Browser proceeded in concert. At the beginning of each scrum, the tool developers choose the requirements to be implemented in the week that starts. Additionally, they feed the framework developer with ideas for refactoring they had during the previous scrum. The framework developer releases a new version of the framework, explains the changes that were made, and selects the next target changes.

At the beginning of the fourth week, the developers of the C-Browser experience problems adapting the code to work with the new release of the framework. Integration takes two days, forcing the team to postpone some of the requirements of the C-Browser to the next scrum. In an informal meeting, the developers of the C-Browser discuss the possibility that this problem will reappear and informally assess the cost of a recurrence. For the developers, this is an issue that requires attention, therefore they decide to bring up the topic in the next scrum coordination meeting.

After the C-Browser developers talk about the cost of integrating the last release of the framework, it is clear that the effort was beyond the expectations of the original plans. The framework developer perceives the discussion as an attack. For the framework developer, the framework is far more important than any tool; when the framework is finished, all tools will see the benefits. A local manager moderates the discussion and helps the others realize that each of them is partially correct. The conversation continues, driving to the conclusion that the cost of ignoring the problem is too high.

Initially, the problem is defined by the symptom that was observed: "excessive integration effort required". Further analysis reveals the probable causes of the problem that should be examined to understand any possible solutions. The following list enumerates the most relevant of these causes.

1. The amount of changes that can be made to the framework in a week are too much to integrate.

2. A simple requirement for the framework that tool developers find on the first day of the scrum needs at least a week to be implemented by the framework developer.

3. One week is an adequate time frame for a scrum; smaller scrums add too much coordination overhead.

4. The refactoring of the framework needs continuous feedback from tool developers; they know the framework and represent the main source of requirements. The importance of the relation between the framework development and immediate use of the framework contradicts any proposal to do the framework refactoring more independently.

5. If every team member can contribute to the development of every piece of software, expertise bottlenecks are minimized, and the idle time waiting for new features can be reduced. However, it could bring along a loss of control that negatively affects the results. In contrast, having well defined, separated responsibilities regarding code editing increases control but also increases idle time and introduces expertise bottlenecks.

The three developers and the local manager exchange e-mails for two days trying to figure out how to proceed. An e-mail with a proposal motivates a counter-proposal that brings up new issues. Follow-up proposals try to consider as many issues as possible, focusing on what seems to be more important. By the end of the second day, the discussion seems to converge. The proposal suggests keeping the one week scrums, and to have the three developers form one single team. At the beginning of each scrum, the three developers will choose the requirements for the framework and for the tool. Any developer can modify the framework under the conditions that unit test coverage for the framework must always be over ninety percent, and only running code will be committed to the CVS repository. Integration will be done on a daily basis, at the end of the day.

The three developers and the local manager consider that the proposed solution has the potential to simplify the work of integration, while at the same time speed up the process of implementing the requirements that tool developers have for the framework. The framework developer is not very happy with losing absolute control of the framework, but nonetheless commits to contribute to the plan.

To implement the solution, all developers need to be given commit access to the framework module in the CVS repository. However, for the time being, the change is only relevant for developers using the new releases of the framework. A daemon process is included to report the results of the unit tests, specially highlighting the levels of code coverage. Before the solution is implemented, the local manager e-mails the management team at Globe to provide a detailed report of the issue and the proposed solution. Once the management team gives the OK, the solution is implemented.

All developers, including those from the maintenance team, are informed of the new work policy. The maintenance team expresses happiness about the change, because it means that in the future they will be able to fix certain bugs in the framework without the need to delegate all the work to the development team with the delays caused by such delegation. Moreover, they commented that such a way of working is commonly called Collective Code Ownership. Had

they been invited to the discussion, they would have suggested this approach and would have reported on successful experiences in the application of Collective Code Ownership at Celsius.

The handling of the breakdown changes the flow of work. The previously independent tasks of tool development and framework development now form a single task called "develop scrum", to which all developers are assigned. The process now includes a daily integration task. Setting test coverage to a minimum of ninety percent raises the standards of the programming tasks. The adoption of Collective Code Ownership changes the distribution of roles and responsibilities. The daemon process, a new tool, is included.

Coding Conventions

On a given day, one of the developers at Aconcagua triggers discussion about the lack of homogeneity that the code has acquired in the last weeks. It is difficult to infer the goal of methods and the semantics of the object's properties. Code became harder to read and modify. This slowed down development and jeopardized quality. This is specially noticeable during pair programming and during refactoring. When looking at the differences between source files during integration, many changes are simply layout and have no functional impact. These simple changes consume effort for integration because they still need to be carefully compared.

The organization employs Collective Code Ownership as a result of a decision made after a prior breakdown. Any developer can change any line of code to add functionality, fix bugs, or refactor. The representation shown in Figure 2.4 is used by the organization to display the key elements of the currently used development practices in meetings or discussions. The developer that triggered the discussion about code homogeneity used the diagram to mark the parts of the work practices where the problem seems to appear.

Developers at Aconcagua know from past experience that it is wise to invite the team at Celsius to any discussion about work breakdowns. They start to discuss the problem via e-mail. Some of those in the discussion decide to include in their reply-to-all e-mails other organization members they think would be interested in participating or who could make a useful contribution. In a short time, the discussion grows to involve a group of developers and researchers from the three institutions. The size of the group that engaged in the discussion seems to confirm the relevance of the problem. The infrastructure team prepares a discussion forum on the issue, and invites everyone who participated in the initial interactions.

An Argentinian developer on a stay at Celsius points out that the problem may be caused by the different backgrounds of all developers. Most developers at Aconcagua started programming Pascal, then moved to Smalltalk, and now program in Java. Their Java code has a mixed Pascal-Smalltalk flavor. In contrast, developers at Celsius come from a C++ background. Their Java code looks just like C++.

One of the youngest team members points out that this is a common problem whose sources have been discussed and documented in several places. She refers the others to Jeff Langr's book *Essential Java Style* [43]. An experienced Java developer contributes a WWW link to Sun Microsystems' *Code Conventions for the Java Programming Language* [1]. Moreover, he comments that many

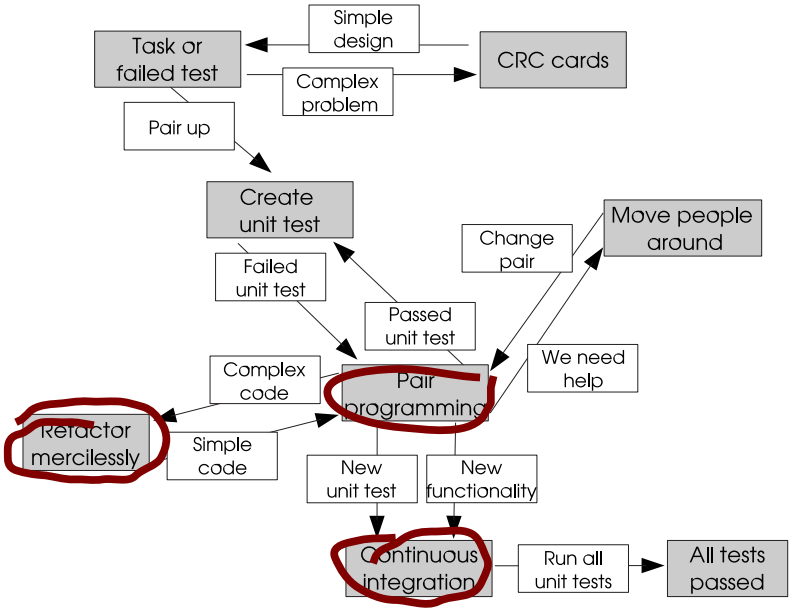

Figure 2.4: A representation of collective code ownership adapted from [72]. Key elements for a breakdown have been marked.

development tools provide mechanisms to format and check compliance with an agreed upon coding style. After a short period of discussion, most team members share the opinion that they are confronting a communication breakdown (where communication takes place through source code) and that a coding standard would help.

Some hours after the discussion started, a proposal argues for the adoption of Sun's conventions. After a short conversation about the positioning of opening curly brackets, those in favor of a Pascal style (Sun's conventions favor C style) accept that this detail is not worth the effort of documenting alternative guidelines. Sun's guidelines are adopted. All team members agree to code according to Sun's guidelines.

In order to ensure that the coding conventions are followed, a code check process is run before every commit to the code repository. On every attempt to commit, the process checks the code against a specification of the convention and generates a report. Code that does not follow the convention will not be committed.

Handling the breakdown changes how work is carried out. The definition of the programming tasks is extended to included the need to follow coding conventions. A link to Sun's coding conventions now appears as a resource for all programming tasks. The set of supporting tools now includes the check-style daemon.

Initially, the breakdown seems to be relevant only for the developers. However, as the coding conventions need to be enforced, the participation of the project managers became important. These managers contributed with an estimate of how measures of coding conventions can adversely affect the organi-

zation, notably how these rules could be received by less agile team members. Consequently, the managers offer a compromise solution for the incremental introduction of the standards in order to reduce initial effort and personnel conflicts. They propose to retain existing nonconforming code in those cases where the implementation stays substantially unchanged (where the definition of "substantially" is left up to the common sense of the developers).

Six weeks after the decision, the approach seems to be proceeding positively. The parts of the software system that are actively developed evolve towards full compliance to the guideline. Those parts that were already mostly finished when the breakdown was perceived stayed mostly unchanged. However, the breakdown did not recur because these finished parts of the system were seldom explored.

The breakdown started as an issue from one of the developers. Soon, other developers acknowledged the problem and started contributing to a discussion. As the discussion evolved towards a need to set up organization-wide rules, project managers felt the need to contribute. Discussion went through the phases of defining the impact of the problem, making a diagnosis of what the causes of the problem could be (based on knowledge of similar problems being already documented by others), evaluation of alternatives to approach the problem (also based on already documented solutions), and implementation of the chosen alternative.

The following list enumerates the main forces that those who participated in the original event acknowledged in a later meeting. The term "force" is used in the way Christopher Alexander [5, 4] introduced it to document the "system of forces" that form the problem or goal approached by a pattern, and as it was later adopted to document the problems approached by software design patterns [26]. Forces are the concerns that make the problem a problem, and must be taken into consideration in any solution. The forces of a pattern document the trade-offs, goals and constraints, and motivating factors/concerns that make-up the problem and its solution. Forces document why the problem is difficult. [51].

Force 1: Source code is a means of communication. It communicates intent and approach. The reader can be the original developer or someone else.

Force 2: In an environment that includes several organizational cultures, idioms that are particular to only one culture can make communication difficult for others.

Force 3: In some cases, idioms empower communication within the culture to which they belong (e.g., usage of i++ instead of i = i + 1 when both are applicable).

Force 4: Collective code ownership requires code to be readable and modifiable by other than the original creator.

Force 5: CVS (the version control system) does not differentiate layout changes from more significant changes.

Force 6: IDEs (Integrated Development Environments) provide functionality to automatically format entire source code files or parts of it following a well-defined style.

Force 7: Enforcing one unique style can help communications. However, having a unique style in an environment that includes several organizational cultures requires that some subgroups agree to abandon the use of idioms and/or to adopt the idioms of other subgroups.

Force 8: Having the opening bracket in the same line generates more compact code.

Force 9: Having the opening bracket in a new line makes it easier to identify the start and end of the method or class.

Force 10: Modifying a file only to accommodate to a coding style introduces unnecessary effort and risk.

Force 11: Refactoring is a good opportunity to reformat code to make it compliant to a particular style.

2.2 Teamwork

Work processes, resources, tools, and communication enable the distributed team to collaboratively develop the product. In Chapter 1, the term "teamwork" was introduced to label this phenomenon. Section 2.1 presented a motivating scenario of teamwork in the context of a distributed, software engineering organization. This section discusses teamwork at a more abstract level in more detail. Where necessary, the domain of discourse is narrowed to the part of interest for this thesis: gaining a representation of teamwork that can be used in handling breakdowns during work in the context of a groupware system.

2.2.1 The Distributed Team

This thesis looks at a distributed software engineering team that works in a coordinated manner to produce a software system. The characteristics of the product being produced introduce the need for participation of people from different disciplines, and with different professional and educational backgrounds.

The team is part of a larger organization where similar teams work. Employees of this organization can be members of more than one team.

Team members are physically distributed in small co-located sub-teams (potentially, only one individual in a single location). The physical separation, often spanning country borders, can imply a separation in time. This separation in space and time renders some common team practices impracticable—or at least prohibitively expensive. An example is regular, face-to-face, coordination meetings.

The form in which a team member views and approaches a problem is to a large extent influenced by the discipline the team member represents, and by the educational and professional background of the team member. The term "perspective" is used in this thesis to refer to a team member's stance toward a problem.

In a team, members from the same discipline or with similar background are likely to share a perspective. The term "shared perspective" refers to a stance towards teamwork problems that is common to a group of team members.

The shared perspectives of a team can be affected by changes in the organization and its context. For example, a change in the balance of competence of team members can cause existing shared perspectives to vanish, and new shared perspectives to appear.

2.2.2 Focusing Communication

In the event of a breakdown in teamwork, collaboration aiming at the resolution of the breakdown will focus on the product of work and on different aspects of teamwork. For example, discussions will make references to tools that cause problems, and to parts of the product whose quality is affected by the problem. A commonly understood description of the product and of teamwork is central in order to focus communication to solve the breakdown. These descriptions need to be consistent for all team members, need to be always available, and need to be complemented with the tools and functions that enable the distributed team to focus communication on any of the elements of the product and of teamwork.

2.2.3 The Product of Teamwork

Team activities concentrate around the product to be developed. The goal of the team is to produce a product within an agreed frame of time, budget, and quality constraints.

A product can be described with different degrees of details and formalism depending on the goal of the description. For example, the description of a product can aim at specifying the conditions of a contract with the customer. This would require a formalism that can be interpreted by the customer, and that simultaneously captures all the details that are needed for successful contract negotiation. A description of a product can be fed into a software system that computes some of the product properties and exchanges product information with other systems. In that case the formalism used to describe the product does not need to be human readable. However, it may need to conform to certain computational properties. STEP (Standard for the Exchange of Product Data) [36] is one such formalism.

Schematic representations of the product (e.g., diagrams, tables, sketches) are commonly used in meetings to focus discussion. A representation of the product on a whiteboard or on a piece of paper serves meeting participants as a common object to which contributions can refer. This usage of diagramming conventions in communication has been the research focus of Steve Whittaker et al. [74] and of Stephen Casner [18] among others. Although in many cases these representations may resemble some existing formalism (that reflects the background of team members), formalization is not a requirement. For Casner, these diagramming conventions are ways of expressing meaningful abstractions that are particular to a task or problem domain. These languages do not need to be formally defined in advance. On the contrary, team members develop the skill to use these conventions and share a background of knowledge that enables understanding. For Casner, conventions are defined through mutual agreement for the meaning of symbols, signs, and the techniques for manipulating them.

The sort of representations of products based on diagramming conventions discussed by Casner and Whittaker fits the needs of team members trying, together, to understand and to solve a breakdown in teamwork. Aside from the difference in the level of formalism, these diagramming languages capture a set of characteristics of the product similar to those captured by more formal languages.

2.2.4 Work Processes, Resources, Tools, and Communication

In the scenario, the work of the distributed team members to produce the product is organized in projects. The product, the quality criteria to be fulfilled, and the distribution of team members set the stage for the definition of the work processes, for the allocation of resources to tasks in these processes, for the choice of tools, and for the choice of forms of communication.

Work processes describe tasks (or activities) and their execution ordering. Tasks are associated through different constructors such as sequence, choice, parallelism and joint synchronization. Tasks can represent atomic units of work, or can be composed of subtasks organized in some execution order.

For the execution of a task, the participation of team members may be

required (unless the tasks can be fully automated). Workers are assigned to tasks usually by means of roles, themselves derived from skill sets. For a single task, the participation of several individuals may be required. Roles (or positions) are usually arranged and related to shape the organizational structure of the organization.

Certain tasks may require the use of specific tools. Moreover, the scope of tools can span beyond the boundaries of single tasks. This is the case where tools are used to support activities that are orthogonal to the production process. Examples of these activities are knowledge management and coordination.

During the production process, intermediate artifacts are generated, modified, and consumed. They flow through the net of tasks . Artifacts can also be seen as a part of the tools or as a mechanism for communication.

Work processes and the mechanisms that aid the conduct of work processes aim at supporting cooperation and coordination during work performance. If a task is to be performed by several team members in close collaboration, special assistance from tools is usually required. The needs of communication vary according to the type of interaction needed. Tight collaboration usually requires rich, synchronous communication.

Software systems that aim at the provision of support for communication, coordination, cooperation, and/or collaboration are named "groupware".

For collaboration towards a solution of a work breakdown, team members need a representation of the work process that is rich enough to cover the key aspects of the observed problem and of an eventual solution. Workflow diagrams, for example, can be used to simultaneously express several views of teamwork [67]. They capture work processes (the control flow view of the workflow), assignment of resources to tasks (the resource view of the workflow), usage of tools (the operational view of the workflow), and artifacts (the data view of the workflow). Other formalisms can be used along with workflow diagrams to express the remaining aspects of teamwork, such as organizational charts that can be used to express the governing structure of the organization.

Any system built to provide collaboration in handling teamwork breakdowns needs to have a representation that is (a) understood by the team members, (b) able to denote all four aspects of work (processes, resources, tools, and artifacts), and (c) consistent with the collaboration or coordination systems used by the team. Therefore, in building the exemplar system for this thesis, the context of intended use dictated much of the representational choices, and should be understood as so derived.

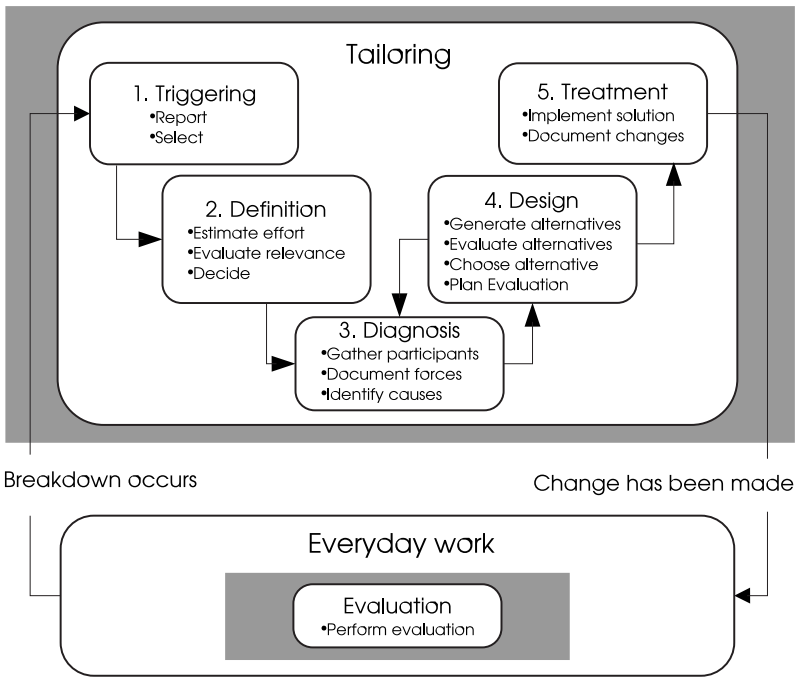

Figure 2.5: Overview of breakdown handling

2.3 Collaborative Tailoring of Teamwork

The scenario presented in Section 2.1 demonstrated how a single team member's perception of certain work problems transforms into a breakdown in the team's ability to work effectively. By raising awareness about a possible source of the breakdown, the team can focus its attention on improving its teamwork (even though work must be temporarily interrupted and refocused). Recognition that a breakdown exists and certifying its possible source engages the whole team in a phase of reflection aimed at taking the work back to its normal flow or, better, to an improved flow. Each breakdown handled represents *collaborative tailoring* of teamwork.

At the outset of the period of reflection, team members are not yet sure that the issue raised represents a relevant problem, or that in fact there is a breakdown. The outset of issue resolution is to define the problem. Both the resulting definition and the process leading to the definition create a common understanding about the nature and impact of the problem, and the importance of its resolution. In a commitment to understand the situation, team members agree to collaboratively prepare a detailed diagnosis of causes of the breakdown. The diagnosis then feeds the design in which team members collaborate to identify and plan possible courses of action.

The patterns of behavior found in the scenario have already been described in some form in the extensive literature on organizational and social psychology (including on organizational behavior and project management), usually under the topic of problem solving in groups and group decision making. There are

several models to explain how problems are identified and resolved in groups that match the events in the scenario. Most models of problem solving and decision making (e.g., those found in [58, 16]) include at least four phases: 1) a phase in which a problem is perceived and an attempt is made to understand the situation or problem (this includes raising awareness about the occurrence of the problem, evaluating its relevance, and making a detailed diagnosis); 2) a phase in which alternatives are generated and evaluated, and a solution is selected; 3) a phase which includes planning for and implementing the solution; and 4) a phase in which the solution is evaluated and modifications are made, if necessary. Figure 2.5 provides an overview of the phases of breakdown handling. Additionally, the figure lists the main objectives of each phase. In this thesis, perceiving and understanding the problem is seen as consisting of three phases namely, triggering, definition and diagnosis. The triggering phase, aims at reporting breakdowns and selecting breakdowns for handling. The definition phase aims at estimating the effort of handling breakdowns, evaluating their relevance, and deciding how to proceed handling them. The diagnosis phase aims at documenting the forces that make-up breakdowns, making sure that all stakeholders participate. The design phase aims at creating solution alternatives, evaluating these alternatives according to agreed criteria, selecting one alternative for implementation, and creating a plan for the evaluation of results once the selected solution is implemented. During the design phase, team members may gain new insight regarding the problem, thus making a review of the diagnosis necessary (indicated in the figure by the arrow going back from design to diagnosis). The treatment phase aims at implementing the chosen solution and documenting the implemented changes. Finally, the evaluation phase aims at conducting the evaluation activities which were planned in the design phase.

Ongoing research investigates the relation between the cognitive style of a group (i.e., the ways in which group members organize stimuli and construct meanings for themselves out of their experiences) and the patterns of activities the group follows for problem solving and decision making processes. Reports [44] suggest that differences in the cognitive style of a team lead to different processes for problem identification and decision making. As an aside, it can be noted that different processes may require different tools, meaning that no one system can provide appropriate tools for all teams, and that the potential for tailoring such a system by its users would provide a larger potential user base.

Breakdown handling is an example of a wicked problem where each step towards a solution changes the understanding of the problem itself [61]. There isn't only one definition of the problem, and there aren't right or wrong solutions, but simply "better", "worse", "good enough", or "not good enough".

Approaching a breakdown in teamwork is in many cases a situation characterized a the symmetry of ignorance[25]; none of the group members holds all the knowledge required to understand and solve the problem . On the contrary, many of them (potentially all of them) have a contribution to make. For example, in the breakdown of Coding Conventions of the scenario, the project managers realize their intervention is needed as the various proposed solutions require enforcing new policies. Moreover, they contribute with their ability to negotiate compromise solutions.

Team efforts to solve the breakdown and to restore the common team work flow may result in changes to the product of work, to work processes, to resources, to tools, and/or to the forms of communication. The impact of these

changes eventually reaches other group members.

Symmetry of ignorance, the complexity of resolving breakdowns in teamwork, and the global impact of changes in teamwork result in a need for collaboration. All group members are valuable participants, especially as each member is expected to cooperate in applying any solution. Once triggering takes place, there is an agreement among the team members that a breakdown exists. The next four phases of breakdown handling — defining, making a diagnosis, designing, treating, and conducting follow-up evaluation of a breakdown in teamwork— translate to tasks that require informed participation from all group members. This thesis introduces the term *collaborative breakdown handling* to label this situation.

The remaining sections of this section explore collaborative breakdown handling in more detail. Each sub-section is motivated by a system requirement that captures a challenge suitable to be the target of computer support. Particular attention is given to the need to cope with the dynamic nature of group decision processes, and therefore with the existence of alternative breakdown handling patterns.

2.3.1 Teamwork Support for Breakdown Handling

Requirement 1: Provide teamwork support for collaborative breakdown handling, both at the level of everyday work and at the level of tailoring.

The idea behind the breakdown handling phase is to acknowledge that a problem exists. Such acknowledge requires affected team members to agree on how to define the problem, to share that definition, but notably the construction of that definition, and to maintain awareness of progress towards resolving the identified problem.

Breakdown handling can be described as a high-level team activity consisting of smaller activities and organized in phases as shown in Figure 2.5. From the triggering of a breakdown, until the evaluation of the required modifications, state changes occur in the constituent activities (i.e., they are started, completed, or canceled) and the content of the related artifacts is modified. At a given time, the state of the constituent activities and the content of the related artifacts define the status of the breakdown handling (namely, how far they have gone and how much more must still be done). Members of the distributed team need to be aware of the status of breakdown handling in order to know what to expect and what is expected from them. Their participation in activities needs to be coordinated and they need to communicate effectively. Teamwork support for breakdown handling at this level consists of the specification of the breakdown handling processes, the coordination tools needed to support the execution of the breakdown handling processes, the data structures to hold the content of the artifacts and the state of the activities that reflect the status of breakdown handling, and the communication channels that are required to foster participation.

The solution that team members develop and choose during the design phase, is implemented in the treatment phase. The result of both phases depends on the forms of changes to teamwork (including to teamwork tools) that are available. Being able to identify the required changes is only valuable if the supporting

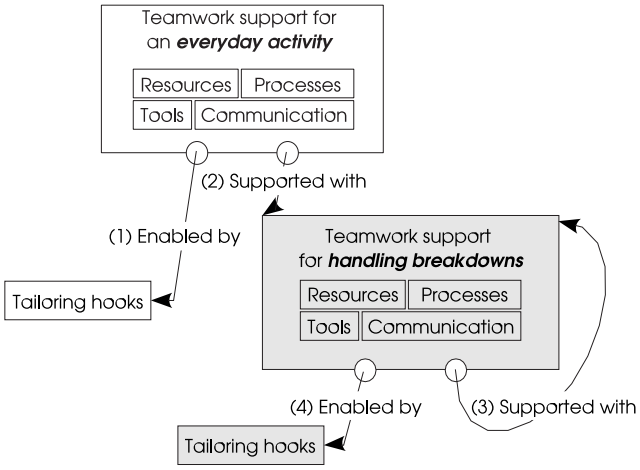

Figure 2.6: Schematic view of the relation between teamwork, tailoring hooks, and support for tailoring

systems are prepared to incorporate these changes. The term *tailoring hooks* is used in this thesis to refer to those aspects of teamwork that can be changed to enable tailoring (For example, a tailoring hook could be a software system API whereby specialized tools can be added to the system, accessing shared data repositories). To support collaborative tailoring of teamwork as the result of breakdown handling, it is required that for any team activity tailoring hooks are available, and teamwork support for breakdown handling is provided.

Collaborative breakdown handling, as a team activity, is itself subject to the occurrence of breakdowns. Attention is no longer directed to the ways of everyday work. The focus of attention is now on the ways of handling breakdowns (i.e., work processes, resources, tools, and communication applied during breakdown handling). In Engelbart's terms, this represents improvement at Level C. In consequence, (1) support for collaborative breakdown handling should be designed to also support handling breakdowns that occur during breakdown handling, and (2) teamwork support for collaborative breakdown handling (i.e., resources, processes, tools, and communication) should provide adequate tailoring hooks.

Figure 2.6 provides a schematic view of the relation between teamwork, tailoring hooks, and support for tailoring. Tailoring of teamwork in relation to any everyday activity is enabled by the provision of tailoring hooks in resources, processes, tools, and communication (label 1 in the figure). Breakdown handling is supported with specific teamwork support (label 2). The same teamwork support can be used to handle breakdowns encountered during tailoring (label 3). Changes to teamwork support for tailoring that result from breakdown handling are enabled by the provision of tailoring hooks in the resources, processes, tools, and communication for breakdown handling (label 4).

During the diagnosis phase, teamwork is analyzed for the causes of the breakdown. During the design phase, solution alternatives are designed and one of them is chosen for implementation. During the treatment phase, the chosen

solution is implemented, thus, introducing changes to teamwork. This view of collaborative breakdown handling assumes that teamwork stays unchanged between the phases of diagnosis and treatment. If that were not the case, the conclusions reached during diagnosis could no longer hold, and/or the solution chosen during design could no longer be applicable. Unforeseen and undetected changes to the original teamwork that occur during breakdown handling can render worthless large parts of the breakdown handling effort. Therefore, all attempts to change teamwork should be channeled through the teamwork support for collaborative tailoring, if only to avoid wasted effort.

2.3.2 Triggering Breakdown Handling

Requirement 2: Provide teamwork support for triggering breakdown handling.

Understanding and supporting the processes that bring a team to a point of reflection (i.e., that bring work processes or tools to the focus of attention) are outside the scope of this thesis. This is also true for understanding the relation between work breakdowns and reflection. Particular research interest has been given to improvement of the processes that motivate reflection with the aim of obtaining more reflective teams (see [32]). The focus of this thesis is activities that occur after acknowledging a breakdown, on the phenomena that unfolds when some members of a team experience a work breakdown that can no longer be ignored.

At the beginning of the breakdown *Collective Code Ownership* , as described in Section 2.1.5, the developers of the C-Browser experience problems integrating the new release of the framework. The problem is perceived by the developers as a load of work beyond the effort predicted for the task. The immediate impact of the problem is that the development of some requirements needs to be postponed to the next scrum. The developers of the C-Browser share the opinion that this issue is not the result of an accidental circumstance but a symptom of a problem with the potential to recur. As it is not in their hands to understand and solve the problem, they raise awareness about it in the next weekly meeting.

The team members that experience the breakdown can tell about the symptoms they observe, can tell how the breakdown affects their ability to work, and can assess the probability of the breakdown to recur. They include this information in a breakdown report.

The severity of the breakdown depends on the impact perceived by those directly affected by the breakdown, and on the importance of their contribution to the team's goal. The breakdown report is an initial indicator of the severity of the problem; the complete picture of the severity is available only after the problem behind breakdown is defined, taking the perspectives of other team members into account.

Triggering breakdown handling involves raising awareness about the problem in the team. As a result of triggering the handling of a breakdown, all team members must become aware of the existence of an unhandled work breakdown. The details of the breakdown as it was experienced by those who report it must be available to all team members.

There is no restriction to the form or number of breakdowns. For example, the first day of work with a new set of tools may result in several breakdowns with varying degrees of severity being reported. The team will possibly discover that some of these breakdowns have a common origin and can be dealt with simultaneously. Support for triggering should allow the coexistence of numerous breakdown reports and should allow aggregating closely related breakdown reports.

How many breakdowns are handled at a time, and how they are chosen for handling depends on the team's priorities and appreciation of the importance of the breakdowns. For example, in the scenario, the breakdown of slow integration is handled immediately because it has strong impact on the critical path of the project plan. Support for triggering should provide means to classify reports according to an initial assessment of severity, and to select breakdowns for proceeding with their definition.

2.3.3 Definition of the Breakdown

Requirement 3: Provide teamwork support to define the breakdown and to decide on continuing breakdown handling.

The breakdown report is the input for the definition of a breakdown. The report usually considers only the perspective of a limited portion of the community. Before advancing to any decision, other team members need to validate the report.

The outcomes of the definition phase are: an agreed evaluation of the relevance of the reported problem; and a team decision to proceed with, postpone, or ignore the handling of the breakdown.

Any attempt to solve a problem is preceded by an assessment of the impact/relevance of the problem. In particular, an assessment of the impact includes the costs the team will incur if the breakdown is not handled.

Deciding to proceed involves accepting the cost of involving the group in the process of reflection. If the team already performed any breakdown handling, an estimation of the effort is based on the team's experience from previous occasions. For the first breakdown, the team needs to resort to best guesses. In any case, estimations need to be accompanied with an indication of how certain they are. For estimations based on the observation of past events, the indication of certainty reflects how many past events have been considered. For estimations based on an estimator's best guess, the value of certainty reflects how much the estimator trusts the given estimation to be correct.

Deciding to postpone breakdown handling implies that the group recognizes its relevance but anyway accepts to sacrifice the cost of any recurrence of this breakdown until this cost justifies the effort of handling.

A decision about how to continue handling a breakdown can be the responsibility of a single person entitled to do so. This is the case for most traditional teams, in which the manager takes the responsibility for strategic decisions. The opinion of the team is a valuable resource used by the manager to make a decision.

There are distributed teams where important decisions are taken by the group as a whole, for example, participative teams. Additionally, there are cases

in which the costs mainly fall on the team members' shoulders, for example if not handling the breakdown results in less enjoyable work hours. For these teams or breakdowns, deciding on breakdown handling is a group task.

It is possible that the group decides not to engage in handling the breakdown, however, its existence and cost is recognized. The cost may be too high for the likely payoff. The team must stay alert for any recurrence and be prepared to change to the estimated cost and benefit and reevaluate its decision.

Ignoring the symptoms implies that the group has agreed that the observed symptoms are not an indication of a problem in teamwork that needs to be dealt with immediately. This could be explained as the group recognizing that the observation had been incorrect, or that the observed phenomena is part of normal behavior and does not require a change in teamwork or is not actionable.

2.3.4 Diagnosis of the Breakdown

Requirement 4: Provide teamwork support to document and to evaluate the forces that shape the breakdown.

Once the group decides it will handle a recognized problem, a phase of diagnosis starts. What is known about the symptoms serves as the starting point.

The goal of this phase is to learn enough about the cause and effect of the problem that hides behind the observed symptoms so that appropriate corrective action can be planned and taken on the basis of sound understanding.

It is hard to formally estimate up-front how much information is enough. This is in some way a matter of experience. The group as a whole needs to decide that enough is known to proceed to treatment. For example, each member could evaluate whether enough is known from his/her perspective in order to explore and evaluate solution alternatives.

Participating in the Diagnosis

A challenge the team faces at this point is to find a balance between having gathered enough useful information, and the cost of gathering more. If, for example, every perspective has to be diagnosed in depth, the cost could be too high.

A breakdown may recur, possibly taking a new form. During diagnosis the team must analyze the data available from past breakdowns and attempt to reuse what applies to the new case. If the new breakdowns is a variation from a previous one, the group needs to identify the differences to produce a tailored solution. If the breakdown is the recurrence of a previous breakdown for which the attempted solution failed, the team needs to focus on the reasons for the failure to produce a better solution.

Not all perspectives are valuable in all situations. For example, if the problem has an origin in some aspect of the product or process, it can be expected that the perspectives responsible for that aspect would have more valuable contributions to make than those that are less related. In order to balance the cost and the value of the results, the perspectives with more to provide need to be identified and all team members that belong to these perspectives need to be invited to contribute.

Structuring Contributions to the Diagnosis

The literature about systems to support decision making [10] points out that discussions tend to be unstructured, dominated by rhetoric, and lack a tangible model able to represent all important aspects for any one participant. Moreover, there are incompatible levels of argumentation and abstraction.

To structure the characterization of the main elements in the problem and solution of a pattern, the documentation style of patterns in software engineering [26] emphasizes identifying "forces"[1]. Similarly, forces are useful in characterizing breakdowns. Forces play a double role; first, they are used to convince others about the existence and relevance of the problem; second, they serve as the acceptance criteria for the solution (they are the requirements that can be checked against any solution). In this way, forces have an important part in enabling participation in problem solving and evaluation.

Forces can be classified according to three types: contextual fact, argument, and system of forces.

A force stated as a contextual fact documents an invariant in the context of the breakdown. A contextual fact cannot be changed. It acts as a restriction, an opportunity, or both. In the breakdown *Coding Conventions* of the scenario (see Section 2.1.5 on Page 13), Force 5 is stated to express a restriction for the CVS system. In contrast, Force 6 is stated with the intention of presenting an opportunity, namely that many IDEs provide functionality to automatically format code.

A force stated as an argument relates choices for a tailoring hook in teamwork (see discussion in Section 2.3.1) to their impact on a common value. In the breakdown *Coding Conventions*, Force 2 is stated as an argument that relates the use of idioms [2] to difficulties in communications with other subgroups. In the diagnosis of a breakdown, an argument can be inspected to see if the breakdown is caused by a choice made for the tailoring hook. In the phase of design of solutions for the breakdown, an argument can be used to generate solution alternatives (i.e., by experimenting with different choices for the tailoring hook).

A system of forces groups contextual facts, arguments, and further systems of forces. It documents a trade-off between forces. A system of forces is introduced to make an interdependency between existing forces explicit. Force 2 in the scenario states that idioms can make communication difficult in an environment with several organizational cultures. Force 3 states that the use of idioms can empower communication within one organizational culture. Force 7 states that enforcing one unique style can help communication, but it may require that some subgroups agree to abandon the use of idioms and/or to adopt the idioms of other subgroups. Force 7 is a system of forces, an explicit trade-off between Force 3 and Force 2.

Evaluating the State of a Force

During diagnosis, team members produce an evaluation of the state of forces. The state of a force can either be "resolved" or "unresolved". From all forces

[1] The term force has been defined in Section 2.1.5. Forces are concerns that make the problem a problem, and must be taken into consideration in any solution. They document the trade-offs, goals and constraints, and motivating factors/concerns that make-up the problem and its solution

[2] Being able to choose idioms to be used team-wide is a tailoring hook.

that are important for the context of the breakdown, some of them are not being resolved in the current situation and need attention. They are deemed the source of the problem. Moreover, in the context of the breakdown, there are already resolved forces that must stay resolved. The team also evaluates forces during the design phase, to assess the effect of a solution alternative, and during the evaluation phase, to check the impact of the chosen and implemented solution.

Arguments are evaluated by considering the current choice for a tailoring hook, effective at the moment of a breakdown. If this choice is seen by team members as a cause of the breakdown, the argument is defined as unresolved. Otherwise, it is resolved. If team members see a contextual fact as a cause of the breakdown, then it is unresolved; otherwise it is considered resolved. A system of forces is resolved if all constituent forces are resolved, and it is unresolved if any one constituent force is unresolved.

During the design phase, a solution alternative aims at finding a choice for the tailoring hooks of an argument that causes it to be resolved. If a force is part of a system of forces, the choice should not cause any other force in the system to become unresolved.

Contextual facts are not inherently unchangeable. They are sometimes a value previously fixed for a force in a situation external to the breakdown (e.g., availability of a resource). What once was variable gets imported to the breakdown as an invariant, thus limiting the freedom to explore alternatives of change. Whenever possible, forces should be expressed as arguments or as systems of arguments (a system of forces containing only arguments or further systems of arguments).

Figure 2.7 provides an overview of the forces involved in the breakdowns *Coding Conventions* and *Collective Code Ownership* of the scenario (presented in Section 2.1.5). Forces were re-written to fit a one page layout of the figure. Each box in the figure lists the forces in a breakdown. The box at the top (labeled "Breakdown 1") represents a previous breakdown which has not been discussed in the scenario. In each box, forces are organized as facts, arguments, and systems of forces. Moreover, at the bottom of each box is the list of the decisions that were taken to handle the breakdown. The figure additionally shows how the decisions that are taken to handle a breakdown represent facts (invariants) that later breakdowns must honor.

Evaluating the state of a system of forces is based on subjective appreciation, and therefore cannot be fully automated. Deciding on this matter is done as a collaborative effort. In order to evaluate the state of forces, team members are asked and their independent appraisals are consistently aggregated. Each team member judges the force from his/her own perspective using available evidence.

The fact that forces are expressed in natural language represents a drawback of using forces as an element for the problem analysis. Being able to identify and describe forces is a valuable skill that requires practice. Moreover, the value of a force cannot always be formally measured, although it can be judged by how much it helps to understand the problem and to determine a solution.

For each involved team member, the forces that make-up a particular situation are implicitly known. In order to use these forces in a team process of problem identification and resolution, they must be made explicit.

During treatment of the breakdown, forces play a role for the evaluation of alternatives. In cases where a balance of all involved forces cannot be achieved,

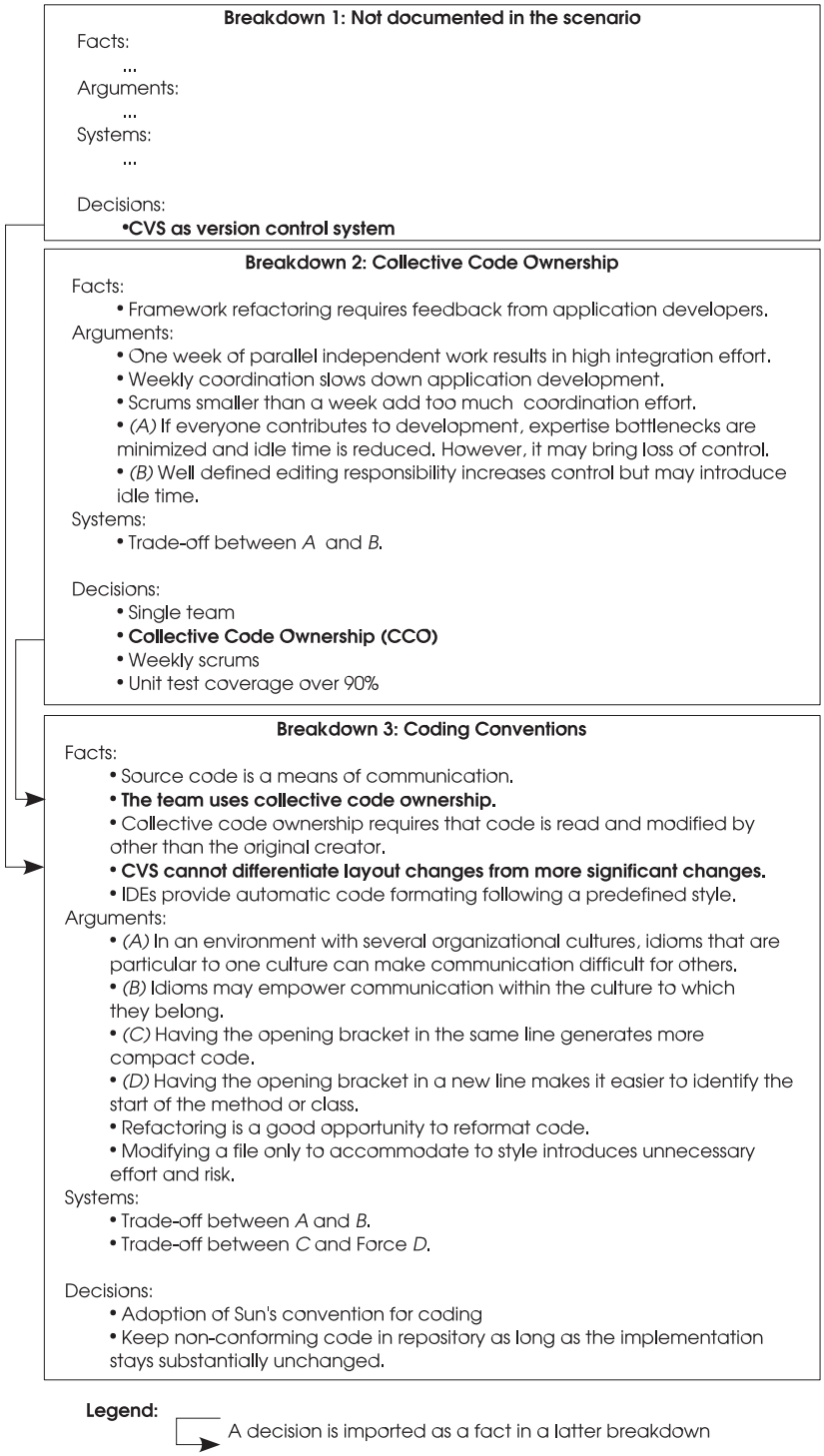

Figure 2.7: A decision to balance forces in the context becomes an invariant for the breakdown.

a compromise solution needs to be found. Successful negotiation depends on understanding the consequences of the possible different sets of alternatives. In this case, this means understanding how forces (the balance of forces) is related to the way work is done (i.e., the product, the team, the process).

Team members understand teamwork and the product of teamwork by being immersed in its everyday practice. However, in cases such as training or analysis, team members make use of documentation that explicitly describes teamwork and the product of teamwork. Documentation serves as an agreed basis on which team members can build. As illustrated in Section 2.2, this thesis builds on the use of diagramming conventions as a means to graphically document teamwork and its product or outcome. Understanding how forces relate to teamwork implies understanding and documenting how forces relate to these graphical representations.

The choice of forces is particular to a given problem. Moreover, a forces can be part of the diagnosis of several breakdowns. Forces can potentially stay unchanged for the lifespan of the organization.

Each breakdown that is handled brings new insights about the forces that shape the organization. Each phase of diagnosis is an opportunity for the organization to learn something more about itself. This continuous learning contributes to reducing the cost of handling breakdowns (there is less to be investigated). However, it requires that some form of memory support is provided.

As the organization learns and evolves, its forces may change. The existence and relevance of forces may change. In correspondence, the organizational memory needs to reflect this evolution.

2.3.5 Design of a Solution

> **Requirement 5: Provide teamwork support to generate alternatives that model possible new forms of teamwork, to select an alternative for implementation, and to plan evaluation.**

The goal of the phase of design is to plan the actions that are needed (if any) to resolve the forces that are the cause of the work breakdown, while at the same time keeping all other forces in their existing resolved state.

To resolve forces, team members can introduce changes in any of the different aspects of teamwork that are open for change (i.e., tailoring hooks). If the forces that need to be balanced depend only on external aspects (i.e., those not under the control of the team), there is nothing that can be done to change them (other than bring the issues to the attention of those who can).

Introducing changes to the way work is done requires exploration, assessment, and agreement.

During exploration, the team identifies alternatives for a new form of performing work. Each alternative needs to be assessed for feasibility and impact. At the end, the group needs to agree on the implementation of one alternative. The alternatives become the focus of collaboration. As such, they need to be explicitly represented in forms that allow the group to collaborate.

Each change to any of the aspects of the product or process defines a new form of teamwork. It is to be expected that, for a non-trivial breakdown, many aspects need to be examined and changed.

The extent to which a team member can contribute to the construction of these alternatives depends on how close his/her perspective is to the aspects that need tuning, or to the aspects that will be affected by the change. In the presence of symmetry of ignorance, constructing an alternative requires the participation of members with different perspectives. The distributed nature of the team introduces new challenges for the coordination of their efforts to create alternatives of work.

The team conducts an evaluation of the forces as discussed in Section 2.3.4 for each of the generated alternatives. The expected effect of an alternative is that it resolves all known forces in the domain of the breakdown, without negatively affecting other aspects of work (e.g., forces that have been resolved for previous breakdowns). However, the actions taken to eliminate the cause of the observed symptoms, could have side effects. A side effect could be a trigger for new cycles of reflection, each of them adding cost and potentially generating new cycles of reflection. Negative side effects need to be identified and minimized to the extent possible.

After conducting the evaluation of the forces for each of the designed alternatives, the team needs to choose one alternative for implementation. Alternatives are compared regarding how they impact forces, and regarding implementation cost and other applicable requirements such as budget limitations. Let $fResolved()$ be a function that returns the set of forces that the team evaluates as resolved for a given alternative. An alternative "A" is better than an alternative "B" with respect to fResolved(), if fResolved(B) is strictly included in fResolved(A). "A" is as good as "B", if fResolved(A) equals fResolved(B). Let $estimatedCost()$ be a function that returns the cost estimated by the team for the implementation of a given alternative. An alternative "A" is better than an alternative "B" with respect to estimatedCost(), if estimatedCost(A) is strictly less than estimatedCost(B). Let $criteria_i()$ be a function that returns the estimation provided by the team for how a given alternative fulfills the $criteria_i$. Assuming that there is a way to determine if an alternative "A" is as good or better than an alternative "B" with respect to $criteria_i()$, then an alternative "A" is a non-dominated alternative if *there is no alternative "B"* such that "B" is as good as "A" regarding all $criteria_i()$, fResolved(), and estimatedCost(), and, in addition, "B" is better than "A" regarding at least one of all $criteria_i()$, fResolved(), and estimatedCost(). The choice of an alternative is made from the set of all non-dominated alternatives.

The extent to which forces are resolved by an alternative and the cost of implementing the alternative are elements that affect the feasibility of its implementation. For example, an unfair distribution of effort and benefit can cause teamwork to fail [28]. Participation becomes central to the acceptance of changes to teamwork.

The time needed for the effects of tailoring to be perceivable depends on the nature of the problem that is approached, and on the changes that are chosen as a way to approach it. Once these factors are known, the team can plan evaluation. The plans for evaluation consist of a schedule of dates where all team members will evaluate the forces and see if they have reached their planned status.

2.3.6 Treatment of the Breakdown

Requirement 6: Provide support to spawn the tailoring project and resume breakdown handling when tailoring finishes, and to document changes.

The goal of this phase is to change teamwork to conform to the alternative agreed upon during design.

A "tailoring project" is created with the goal of doing what is necessary to make teamwork as in the agreed alternative. In this thesis, the tailoring project is seen as an independent process that spawns from breakdown handling. While the tailoring project is running, breakdown handling is suspended, and so is the actual work (at least, the part of work directly affected by the breakdown).

Support for executing the changes to teamwork that are required in order to implement the chosen alternative falls outside the scope of this thesis. The complexity of this task depends on the nature of the changes that need to be done. For example, changes to the infrastructure may require hiring external development teams. Moreover, work cannot, in most cases, stop until changes have taken place. This confronts the tailors with the challenge of changing work, while it takes place. These challenges are the focus of future research investigations.

When the tailoring project finishes, the corresponding breakdown handling process is resumed. As already discussed, collaboration for breakdown handling relies on the existence of up-to-date representations of teamwork and the product of teamwork. After the tailoring project has introduced changes, these representations of teamwork and the product need to be updated to reflect the new status.

2.3.7 Follow-Up Evaluation

Requirement 7: Provide teamwork support for conducting follow-up evaluation of the results of change.

The goal of breakdown handling is to eliminate the cause of the problem that started the process. These changes have been carefully planned. However, their effect can only be assessed but never ensured. Although the core of the process of reflection can be said to end after the corresponding actions have been taken (i.e., change is implemented), its result is still to be seen.

Executing and maintaining corrective actions requires effort. Such effort is only worthwhile if corrective actions are effective. In order to ensure this, the planned evaluation needs to take place. On the scheduled dates, all team members will assess the status of the forces. Their individual observations need to be consolidated. The result needs to be compared with the expected status by the given date.

If the results of the evaluation show that the solution had the expected impact on the forces, the solution is marked as successful, the breakdown is closed, and all related information is persistently stored. On the other hand, if the results of the evaluation show that the solution did not have the expected effect, two paths of actions are possible. If there are signs that indicate that the solution is working but it needs more time, the team may decide to re-schedule evaluation. Or, the breakdown solution is marked as failed, the breakdown is

closed, and all information is persistently stored. Breakdowns for which handling failed reappear as new breakdowns that are treated in a new cycle of breakdown handling. As part of handling of such a new breakdown, the team may decide to tweak the failed solution based on a deeper definition and design, or to undo the changes prescribed by the failed solution and implement a new one.

2.3.8 Summary of Requirements

Collaborative breakdown handling is the process of tailoring teamwork by solving work breakdowns collaboratively. Collaborative breakdown handling requires teamwork support, both at the level of everyday work and at the level of tailoring (Requirement 1). In particular, it requires specific teamwork support for each of the six phases of the process. It requires teamwork support for triggering breakdown handling (Requirement 2). It requires teamwork support to define the breakdown and to decide on continuing breakdown handling (Requirement 3). It requires teamwork support to document and to evaluate the forces that shape the breakdown (Requirement 4). It requires teamwork support to generate alternatives that model possible new forms of teamwork, to select an alternative for implementation, and to plan evaluation (Requirement 5). It requires support to spawn the tailoring project and resume breakdown handling when tailoring finishes, and to document changes (Requirement 6). It requires teamwork support for conducting follow-up evaluation of the results of change (Requirement 7).

Chapter 3

State of the Art

This chapter presents an overview of the state of the art in the area of support for collaborative tailoring. Relevant related work is organized in three sections. Section 3.1 provides an overview of the most relevant contributions towards the construction of groupware that can be tailored. Section 3.2 presents work that contributes to understand tailoring as a process. Finally, Section 3.3 summarizes the most relevant contributions regarding collaboration in the context of tailoring.

3.1 Tailorable Groupware

In 1992 Thomas Malone [50] introduced the idea of radical tailorability as an approach to match cooperative work tools to their contexts of use. Radically tailorable systems allow end users to create a wide range of different groupware applications by progressively modifying a working system. The approach, instantiated in a system called OVAL, is based on four key building blocks. Structured objects represent domain elements such as people, tasks, messages and meetings. User customizable views display collections of objects and can be used to edit individual object's properties. Rule based agents automatically perform tasks on behalf of users without requiring human intervention. Links represent relationships among objects. Radical tailoring is done by defining new types of objects, adding fields to existing object types, selecting and configuring views, creating agents and rules, and inserting new links. This approach to tailorability is reminiscent of computer programming generally, and object-based, rule-based programming specifically. It is rare to find much flexibility in a system that does not manifest itself as some kind of programming.

Malone was not the first to find tailorability as a way to build successful groupware. A year before, Greenberg identified tailoring [1] (for individuals and for the group) as an approach to increase groupware acceptance and to be able to keep pace with evolving user needs. He built SHARE, an example of a personalizable application sharing system [2]. In SHARE, a system component

[1] Greenberg uses the term personalization, that he defines as the ability to tailor a system's behavior to match the particular needs of users or groups of users.

[2] An application sharing systems allows single user applications to be shared through the network.

provides functionality to tailor the floor control policy used to share applications. A handful of other contemporary systems included similar provisions for tailorability. Information LENS [49], an information manager for news and e-mail, allowed users to created tailored groupware by customizing templates and creating filtering rules. CRUISER [62], a virtual hallway based on video to support casual interaction, allows people to configure privacy settings and, as a consequence, to tailor interaction options. Virtual Learning Community [37] is an asynchronous conferencing system. The conference facilitator can tailor some of the system's settings to achieve a variety of collaboration settings. For example, the facilitator can tailor the system to force equal participation and discourage lurking.

Recent work by Robert Slagter [64] proposes a design methodology for groupware based on the principle of composing generic groupware services. A groupware application provides a Groupware Service (GS). The groupware service is composed of several Groupware Service Modules (GSM). End users can select and compose GSMs in a GS to obtain the desired behavior. In a finer granularity, GSM consist of Groupware Service Module Elements that represent elementary units of externally observable, groupware behavior (e.g., starting a conference). The key criteria behind the success of this approach to enable end-user tailoring is that users can find and learn about module elements (i.e., understand the groupware behavior associated with a GSME) and use them to describe and prescribe groupware.

Implementing tailorability in groupware was the topic of two workshops held in 1998 and 1999. The workshop reports [52, 66] document the workshops' findings. The organizers report that tailoring groupware differs from mainstream tailoring in that proposed changes must be accepted by all members of the group (or at least a critical mass or accepted decision-makers for the group). As a result, the question of how to regard tailoring as a collaborative activity was found to be a relevant open issue. Tailoring must involve end users as the primary source of information about the work to be performed and the groupware that has been used. Successful tailoring is only possible if it contributes to improving the groupware as an integral part of the work process. As a participant, Volker Wulf introduced the concept of "scopes of validity" to refer to a particular situation or scenario to which a set of changes (the tailoring) apply. The major problem in being able to specify scopes of validity is to provide flexibility while simultaneously keeping consistency. Wulf points out that meta-tailoring facilities need to be provided to be able to handle inconsistencies and conflicts. This observation is in-line with the discussion presented in Section 2.3.1 of this thesis regarding the need to support "tailoring of the tailoring support" and the need to provide tailoring hooks that support tailoring.

State of the art in tailorable groupware focuses on building groupware that can be changed through a variety of technical means. The main objective of these approaches is to achieve richness and flexibility in how systems can be tailored (where presumably the more aspects of a system that can be tailored and the more alternatives for tailoring each aspect are available, the better). Although researchers realized the need to involve communities of users in collaboration for tailoring, existing approaches for tailorable groupware fall short in providing specific community support functions thus, failing to cover the core concern of the seven requirements presented in this thesis. Tailoring is only covered as a technical issue (looking at the technical system) without consid-

ering the team processes that form the context of tailoring (the corresponding social system). Most approaches to tailorable groupware focus on end-user tailorability. End-user tailorability is a valuable resource to cover requirement 6 (support to spawn the tailoring project), as it makes possible the participation of end-users in the tailoring project. Participation in the tailoring project is important to ensure that the implemented changes match the intention of the users. Moreover, by directly participating in tailoring, team members gain a better understanding about the tailoring hooks in the teamwork. Understanding the tailoring hooks is important for fruitful participation in the design of the solution, and therefore, for covering requirement 5 (teamwork support for the generation of solutions).

3.2 Understanding Tailoring

Anders Mørch [42] sees tailoring as a form to bridge the gap between presentation objects (available to the users) and implementation code (hidden from the users). He classifies tailoring in relation to the types of activities that are performed using three levels of tailoring. The first level of tailoring is customization. To customize is to modify the appearance of presentation objects, or to edit their attribute values selecting from a set of predefined options. Customization is done through templates or forms that show users the configurable attributes and the possible values. The second level of tailoring is integration. Integration allows bridging the gap between presentation and implementation by allowing users to add functionality (both presentation and implementation). It is not necessary to access the underlying implementation code. Users link predefined components from a set of available ones. Each component encapsulates a unit of system behavior required to perform a well-defined task (e.g., save a file). The work done by Slagter (see previous section) mainly focuses on this type of tailoring. Extension is the third level of of tailoring. Tailoring as extension is to improve the functionality of an application by adding new code. Depending on the complexity of the programming facilities that are available, extension may be done by end-users or may require the participation of developers.

Mørch considers that the previously mentioned gap between presentation objects and implementation code reflects the distance that exists between the work of groupware developers and the work of end users. Thus, to bring developers and users closer to each other would help bridge the gap. In Mørch and Mehandjiev [53], the authors propose a way to improve collaboration between users and developers. They propose building and tailoring applications by composing application units and by documenting the design rationale with multiple representations. Applications are decomposed in applications units. Each application unit is specified using different representations. User-oriented representations capture high-level views of the systems, aspects of the domain, and aspects of how to use the system. Design representations are more technical, addressing specific design decisions and the context in which those decisions were made. Good tailorable systems are build with a rich set of user-oriented representations, some of which are executable since they describe functionality in a way that can be translated to program code.

Richard Bentley and Paul Dourish [13] talk about the "customization gulf" which is characterized by two inter-related problems: level of customization pos-

sible and the language of customization. They propose the idea of incremental customization where different levels of customization require different levels of expertise, to the point where users can only express what they want to more advance users or to the developers. They aim at systems that can be customized and used in unpredicted ways (that support innovation). The more aspects of the systems that can be customized the better (e.g., being able to alter the concurrency control mechanisms). They argue that all the aspects that are fixed in most groupware systems should be open to customization.

Wendy Mackay [47] takes an ethnographic stance towards tailoring. She postulates that the use of information technology is a co-adaptive phenomenon; the use of technology affects human behavior and at the same time, human behavior redefines technology. Tailoring allows particular patterns of use to be encoded in the system, later influencing the way the system is used. The presence of shared artifacts (e.g., customization files) is of particular relevance to her approach. Shared artifacts provide the mechanism by which individual behavior can influence global institutional properties and future implementations of the technology. Mackay also discovered common patterns of customization. Users most commonly customize when they are new to the organization and know the least about the technology and its use. Customization is commonly done as a way to explore new environments. Users create customizations as an attempt to capture their current work setting. External effects that cause users to reflect upon their work increase the probability that users will customize. Users that customize like to keep the same work environment. As a consequence, new new software is adopted (e.g., as a result of management decision) they try to retrofit new software to be like the old one. Customization can generate a feeling of ownership that motivates users to oppose any changes in the system that would threaten their customizations. The most common form of customization occurs when users realize the existence of recurring work patterns and encode them in a customization. Mackay proposes that software should be designed to include mechanisms that support reflection about the use of software and mechanisms for sharing customizations. Non-effective/useless tailoring can be harmful; therefore, it is necessary to provide mechanisms to assess the usefulness of customizations in a reflective way (i.e., being able to relate the impact of customization to work practices).

Allan MacLean and colleagues [48] argue that tailorable systems alone are not enough. They should be complemented with a tailoring culture that motivates tailoring. To evolve a tailoring culture one must understand the needs and skills of the organization's members. There are *workers*, who have no interest in the computer system per se. They just want to get work done and have no expectation of being able to tailor the system. There are *tinkerers*, who enjoy exploring a computer system but may not fully understand it. There are *programmers*, who understand the internals of computer systems. Programmers are usually not accessible to ordinary workers. To shorten the separation between workers or tinkerers and programmers, there could be *handymen*. Handymen work closely with users and help with immediate needs. Moreover, handymen can communicate user's needs to developers for more complex tailoring.

State of the art research in understanding tailoring is mainly focused on answering the questions: why is tailoring necessary? and how can it be enabled? MacLean and Mackay take a first look at how tailoring is done as collaboration by talking about a tailoring culture and describing patterns of sharing. Tailor-

ing is teamwork and teamwork is defined in terms of processes, artifacts, tools and communication. The gap in these research efforts is the recognition that the support for tailoring should itself be tailorable (part of requirement 1 of this thesis), and should itself be studied as both a technical and social phenomenon. The work done by Mørch on composing application units and documenting the design rationale partially fulfills requirement 4 (teamwork support to document and to evaluate forces) as design rationale can document the forces that motivated tailoring. However, the work done by Mørch does not support building the comprehensive and flexible landscape of the forces that underly several tailoring efforts needed to cover requirement 3 (teamwork support for the definition of the breakdown) and requirement 4 (teamwork support for the diagnosis of the breakdown). The work done by MacLean and colleagues partially covers requirement 1 (teamwork support for collaborative breakdown handling as a whole) by defining a tailoring organization/culture. However, it lacks a definition of the processes, the tools, and the artifacts to support the efforts of the tailoring organization.

3.3 Collaboration in Tailoring

In his paper "From Tailorism to Tailorability", Helge Kahler [41] presents an interesting discussion regarding the role of negotiation in tailoring. He presents the problem of finding agreed solutions in flexible and constantly adapting organizations. He argues that instead of trying to find an optimal solution, negotiation should be carried out to achieve stable solutions that are agreed upon by the participants. Important dimensions of negotiation for tailoring that need to be considered are the initiator of tailoring, the actor, the affected persons, the subject (i.e., target of the tailoring), the goal of tailoring, when or for how long, and in which part of the organization the change is to be carried out. Kahler indicates that no work has been done in covering how a group tailors a groupware system to the group needs. He indicates that one important aspect is moderation of interests in the groupware tailoring process.

Experimental studies about sharing of customization files, conducted by Mackay and colleagues, demonstrate that tailoring is not an individual activity[46]. Tailoring is a an activity that involves group reflection. As such, tailoring requires support to identify stakeholders and views. It requires mechanisms to collaboratively identify, state, and re-frame problems, and to state forces, values, restrictions, and motivation. It requires mechanisms to safely attempt different solution approaches and to state the strengths and weaknesses of each approach.

Users would create customization files for single user systems and later share them by e-mail. As a result of the studies, the authors recognize that not all sharing of customization files is beneficial. In some cases, given the fact that files were shared without any previous evaluation, sharing would cause errors to be distributed along with the files. It was observed that people in all job categories (except system programmers) preferred to "ask a person" and, only if that yielded no result, they would "copy and experiment" (to borrow someone else's files and edit them). The authors give no explanation for this behavior. One can speculate that asking for help first reduces the work of finding adequate customizations, or that asking a trustworthy colleague ensures quality

and usefulness in the customizations, or that talking directly to the creator of a file reduces the work required to understand its motivation, design and implementation.

The report of the workshop "Tailorable Groupware: Issues, Methods and Architectures" [52] states that individual tailoring is not enough and that support for tailoring the "collaboration objects" of groupware applications is important as well. The work done by Weigang Wang and Jörg Haake with the CHIPS [70] system is a good example of groupware that aims at tailoring the collaboration facilities. CHIPS provides support for coordination of work processes and for the creation of shared workspaces. Users can edit the process description to change the flow of work, the flow of documents, or to change resource allocation. A unique aspect of CHIPS is that process descriptions and shared workspaces can be edited in synchronous collaboration. That is, the tool takes cares of replicating changes and keeping consistency. Processes and workspaces can be annotated. A telepointer facility and a bridge to external video conference services complement the customization tools. In a common scenario, users edit a process description while discussing the changes over the video conference.

Volker Wulf [78] approached the problem of conflicts of interest when tailoring functions of groupware systems. These conflicts arise from the diverging interests of the *activator* (the user who intends to tailor a system's function) and the *affected user* who is affected by the activator's choices. The solution proposed by Wulf is to extend groupware systems with a "negotiability" mechanism for each tailorable groupware function to technically support negotiation. Whenever the activator plans to change a tailorable function to an option X of the set of available options, the affected user can agree, disagree or counter-propose a different alternative. Upon agreement, the choice proposed by the activator is activated. Upon disagreement, the system function takes its default configuration. If the affected user makes a counter-proposal, negotiation continues. The negotiability mechanism can be configured with a maximum number of negotiation loops. In case the roles of activator and affected user are fulfilled by several individuals, additional tools, such as voting, may be needed.

The work done Katharina Just-Hahn and colleagues focuses on the provision of negotiation support during tailoring of workflows [40, 31]. Their work aimed at answering the questions: who can perform a given change in the workflow?, and who needs to be involved in the negotiation of the change? They propose a methodology, called Step-By-Step. The methodology is modeled as a workflow. To answer the question of who can make a certain modification, the workflow is complemented with information that indicates for each activity the types of modifications that are possible and the conditions that must be met. The persons or organizational units that need to participate in a change negotiation, or that need to be notified after a change are similarly indicated for each activity in the workflow. The Step-by-Step model requires that all modifications are recorded along with authors, rational of the modification, and duration. This information is valuable during later modifications. Step-by-Step builds on Wulf's model of negotiability and extends it to consider the following possibilities for reaction: to accept only until withdrawal, to rank several proposals, and to change the steps followed to negotiate. This later extension represents tailoring of the tailoring process.

State of the art in collaborative tailoring clearly shows that tailoring involves multiple collaborating stakeholders and must be supported by groupware.

Groupware to support tailoring should enable communication (e.g., to initiate tailoring), collaboration (e.g., to create a definition of the needs of tailoring that considers all perspectives), co-operation and coordination, (e.g., to create alternative solutions), and negotiation (e.g., to decide on multiple paths of action). The CHIPS system can be used to create a flexible process to coordinate the tailoring activities. Moreover, the CHIPS system is itself tailorable, what makes it a good candidate to support collaborative tailoring at the level of everyday work and at the level of tailoring. However, the CHIPS system lacks specific support for eachof the phases of breakdown handling, for example, it lacks support for documenting forces (needed to cover requirements 3 and 4). The work done by Katharina Just-Hahn and colleagues supports recording the rationale of the changes for later referencing and supports negotiation for the introduction of changes. However, it lacks support for collaboratively exploring the causes of the problems that motivate tailoring (requirement 4). The above mentioned research contributions are also limited as they are only applicable to the system being studied.

3.4 Summary

Chapter 2.3 lists seven requirements. As discussed in the previous sections, existing related work partially covers some of these requirements. However, there is no solution that covers them all. To provide such a complete solution is the aim of this thesis. Requirement 1 argues for coordination support for collaborative tailoring, which should itself be tailorable. Existing related work that provides tailorable support for work coordination lacks of the specific data structures needed to hold information about the breakdown handling process. Requirement 2, the provision of support for triggering breakdown handling, implies the provision of support to collaboratively report the occurrence of work breakdowns, support to collaboratively estimate its severity, support to raise awareness to other team members, support to aggregate closely related breakdown for their joint treatment, and support to classify breakdown according to severity. There aren't any groupware systems that provide this support. Requirement 3, the provision of support for the definition of breakdowns and for deciding on continuation, implies the provision of support to collaborative assess the impact of the breakdown, and support for group decision about handling the breakdown. Although there aren't any systems to specifically support these activities in the context of tailoring, this thesis draws from results from the area of GDSS (Group Decision Support Systems) to build specific tools. Requirement 4, the provision of support for documenting and evaluating the forces that shape the breakdown, implies support to collaboratively document the forces, support to assure and motivate the participation of all important stakeholders, the possibility to consider information about forces which were relevant in past related breakdowns, support to anchor forces on agreed representations of teamwork, and support to collaboratively identify causes. Existing related work only provides the possibility of documenting rationale of changes. Requirement 5, support to create and select solutions and to plan evaluation, implies support to collaboratively create solutions, to assure and motivate participation of all relevant stakeholders, support to assess the effect and the side effects of solutions, support to collaboratively select a solution for implementa-

tion, and support for planning evaluation. Existing related work contributes, through end-user tailoring, to more informed participation in the generation of solutions. As for requirement 3, results from research in GDSS can be applied to build the tools needed to collaboratively assess the impact of solutions and to select one solution for implementation. Requirement 6, support for spawning the tailoring project, implies support to launch the tailoring project and to update representations of teamwork when the tailoring project finishes. This requirement is not covered by existing related work. Finally, requirement 7, the provision of collaboration support for evaluating the result of the implemented solution has neither been covered by existing related work.

Chapter 4

Approach

4.1 Overview

We are changing gears now —from philosophical to testing whether the philosophy can be embodied in a tangible usable system. We start with a specification for a design in this chapter, then move in the next to describe a reference implementation. Experience with the reference implementation lends observable credibility to both feasibility and value of the design.

Peter and Trudy Johnson-Lenz coined the term groupware as intentional group processes plus software to support them [38]. In their paper titled *Rhythms, Boundaries, and Containers: Creative Dynamics of Asynchronous Group Life* [39], they review two prevailing approaches to groupware: (1) groupware as the mechanism that makes groups work through the use of explicit forms and procedures, and (2) groupware as a context or open space that allows groups to self organize. They claim that computer-supported groups need groupware that provides more than procedures and open space. They argue for tailorable groupware capable of embodying forms to serve their changing needs and evolving purposes. They aim at a balanced combination of both approaches. Effective groupware is somewhere in the middle of these two approaches. Form (approach 1) and context (approach 2) are part of a dynamic creative process: form continually emerges from and evolves in response to its living context.

The approach to support collaborative tailoring of teamwork presented in this thesis follows the philosophy proposed by Peter and Trudy Johnson-Lenz. It consists of a method for collaborative breakdown handling, a selection of specific groupware tools to be used for the deliberation activities defined by the method, and guidance in the form of scaffoldings for the application of the method. The method, the specific tools that support the method, and the scaffolding for the application of the method, have been designed with a balance of open space and structure that reflect what is known about collaborative tailoring of teamwork at the time of this writing. Moreover, the method, the tools, and the scaffolding can be tailored. The proposed support for collaborative tailoring of teamwork is delivered as a stand-alone groupware system for collaborative tailoring. The system can be deployed along existing groupware systems, thus extending them with support for collaborative tailoring. However, the extent to which they can be tailored dependens on the tailoring facilities that they provide. Figure 4.1

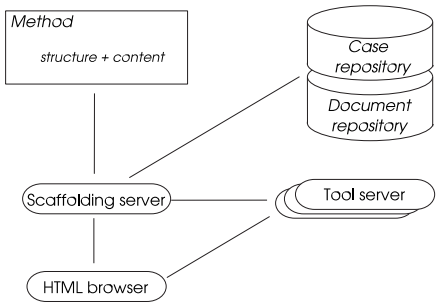

Figure 4.1: System's main elements

shows the main elements of the system.

The method of collaborative breakdown handling follows the structure presented during problem analysis. It consists of six phases, namely, triggering, definition, diagnosis, design, treatment, and follow-up evaluation. Each phase consists of several activities. For each activity the method details the expected outcomes, suggests useful tools and artifacts, and organizes participation and communication. Figure 4.2 provides an overview of the activities that compose each phase.

The term "case" is used to refer to the application of the method to handle a particular breakdown. A case repository maintains information about the progress of each case, and links to retrieve the relevant artifacts and tools. When a case is initiated, the system creates a new entry in the case repository to record activity information about the case. Similarly, all required documents are created and initialized.

The structure of the method specifies the ordering of activities, the structure of the required artifacts and their relation to activities, and the use of tools. It is used by the system to create a hypermedia documenting the scaffolding for each case, to create and initialize the documents that the method produces, and to link the tools. The content of the hypermedia is mostly an on-line, hypermedia version of the content of sections 4.3 to 4.8 of the thesis. It provides detailed guidance for the users, for example, to indicate how to perform a task or how to use deliberation tools to collaboratively obtain the required outcomes.

Some activities in collaborative breakdown handling require the use of specific groupware tools. For example, this is the case for activities that result in the aggregation of the individual contributions from several team members (e.g., effort estimations), activities that require negotiation and decision (e.g., deciding how to proceed), and activities that require communication (e.g., deliberation to produce a solution alternative). The thesis presents a collection of specific groupware tools targeting activities for which there is no generic groupware tool that applies, or where having a specific tool results in better collaboration. An example of such a tool is the "Breakdown Landscape" that enables the collaborative construction and maintenance of a knowledge base that relates breakdowns and teamwork. Tools reside and are accessed from an independent tool server.

The scaffolding is delivered as a hypermedia document that combines the method description, the progress information for each case, the documents for

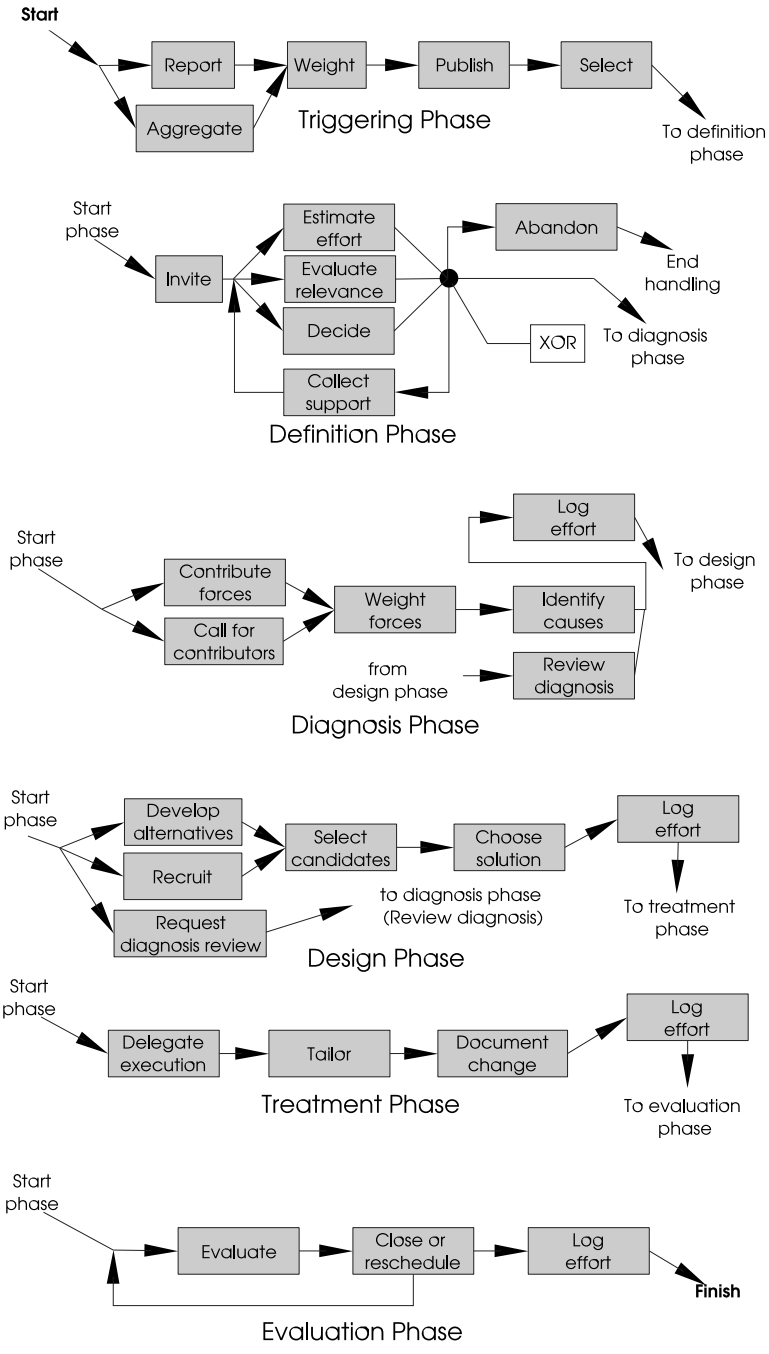

Figure 4.2: Activities in each phase of the breakdown handling process

each case, and links to the external tools. Users access the scaffolding and the tools through a web-browser. The server processes user input and updates the state of the cases. Changes to documents are directly uploaded to the document repository without involving the scaffolding server.

Section 4.2 presents the details of the scaffolding server that delivers the method, links tools and the artifacts and that helps coordinate the application of the method. Sections 4.3 to 4.8 present the method in detail. Supporting groupware tools are introduced as the need for them becomes clear.

4.2 Guidance and Coordination Support

To assure that team members contribute adequately in time and form to the handling of the breakdown, in this thesis a method is suggested. The method serves as a manual of operations and procedures. It is documented at a level of detail that enables any newcomer to contribute fruitfully to breakdown handling.

The method is delivered electronically via the guidance and coordination system described in Section 4.2.2. The choice of structure used to document the method aims at a straight-forward delivery in the form of editable hypermedia. Moreover, the chosen structure enables the creation and maintenance of the data objects that hold information about breakdown handling cases. This is needed to coordinate the work of the distributed team members.

The choice of editable hypermedia as the mechanism for capturing and coordinating the tailoring processes is motivated by the need for flexibility (e.g., to allow tailoring, specially by end users) and the need to put users in control of the tailoring process. Other alternatives for process modeling and enactment such as Workflow Management Systems (WFMS) aim at strictly defining work processes whose correct execution can be enforced and partly automated by a workflow engine. They provide more support for automation expense of reduced flexibility. Moreover, most workflow management systems require expert knowledge to document processes. The approach of editable hypermedia implemented in Scaki focuses on flexibility and simplicity however, it requires more manual work (specially for process enactment).

The method documented in this thesis has been developed based on literature on problem solving in groups. It also reflects the results of early experiments of collaborative tailoring. The organization of the method description and the mechanisms chosen to store and to deliver it aim at enabling end-user tailoring. It is expected that its usage will result in improved versions of the method. Therefore, tailoring of the tailoring method is supported as well.

4.2.1 The Specification of the Method

The method is specified in terms of activities, artifacts, tools, and team member participation. Object Oriented Programming has been chosen as the specification formalism. The class diagram of Figure 4.3 shows the main elements of the method. The diagram has been adapted from the model proposed by the Workflow Management Coalition to specify workflows ([33], page 30).

The process of breakdown handling is decomposed in atomic activities. The method presented in this thesis provides detailed instructions to achieve the intent of each activity. When a tool is needed in order to perform an activity, the method references the specification of the suggested tool (the "uses" relation in the the diagram). Activities commonly require the creation, modification, or use of artifacts. When an activity involves an artifact, the method references the specification of the artifact (the "develops" relation in the diagram). Artifacts flow from the activities that create them, to activities that use and/or modify them. Artifact flows determine dependencies among activities. There are also synchronization dependencies among activities that do not relate to artifact flows. Dependencies among activities are documented in the activation conditions part of an activity. Similarly, the conditions that need to be met to treat an activity as completed are documented (completion conditions). If

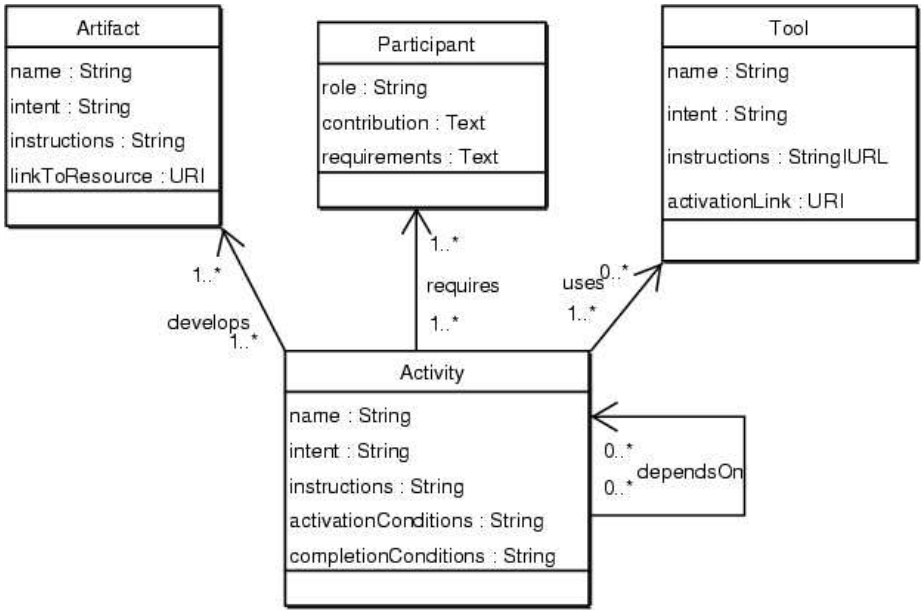

Figure 4.3: Object model of a hypermedia scaffolding

a template for the construction of an artifact is available, the method provides the URI (Unique Resource Identifier) to download it (the "linkToResource" attribute in the diagram). The method also provides instructions for the use of tools. In some cases, these instructions are replaced by a reference to the user manual of the tool. When possible, the method provides the URI that can be used to start the tool.

Finally, the method documents participation in terms of the roles that need to be fulfilled by team members in each of the activities. The assignment of team members to roles is done when the method is used in a particular case. To be eligible for participation in a given role, a team member needs to fulfill certain requirements which are textually documented. For example, a role may require that the participant belongs to a given shared perspective, or has certain skills. It is possible that many team members fulfill a single role.

The method is presented in sections 4.3 to 4.8. Each section focuses on one of the six phases of breakdown handling. The remainder of this current section presents the system that is used to deliver the method on-line to the distributed team members, and that supports team members in coordinating their contributions to breakdown handling.

4.2.2 The Scaffolding Server

Breakdowns differ in complexity. In the scenario, the breakdown presented in section 2.1.5 (Coding Conventions) encounters a problem that requires the participation of only one perspective (developers). Moreover, participants go through definition and diagnosis without much need for coordination or collaboration support. Only at the point of deciding for an alternative, do they

need to capture explicit commitment from all developers. Coordination and process guidance support must be provided to enable the team to deviate from the suggested process, and to use alternative tools. Team members cannot be expected to immediately know all details about the activities, tools, and artifacts involved in breakdown handling. Their interest and expertise focus on the everyday work practice. Therefore, the amount of detail documented about the breakdown handling process and related elements should be enough to serve as a manual of operation. A groupware system to support collaborative breakdown handling documents the chosen method and tracks the application of the method to cases. Support for tracking the application of the method in a case must make it possible to access related artifacts and tools, and to perform activities outside of the planned schedule if participants consider it adequate. The mechanism used to describe the process must be rich and simple so team members can understand it. Moreover, it should be possible for users to collaboratively tailor the process description in response to breakdowns and then have the guidelines appropriately modified.

There exists a wide range of coordination support mechanisms. There are workflow systems that provide computer-controlled execution of formally specified processes. Traditional workflow management systems require expert knowledge to document processes and impose a rigid structure in how to describe and execute processes[33]. The need for more flexible process coordination support motivated the creation of several flexible work coordination systems. These systems aim, for example, at supporting emergent workflows. The CHIPS system [70] is an example of such a flexible work coordination system. Neither traditional workflow systems, nor the more flexible alternatives, support the provision of detailed work guides.

In education, scaffolding [69] refers to the provision of a supporting framework, for example in the form of step-by-step instructions, that students can use to complete a complex assignment. Scaffolding support is gradually removed (or ignored by the students) as students gain practice. This thesis uses the concept of "scaffolding" to characterize the groupware system that stores and delivers the method of collaborative breakdown handling and guides a work team in the application of the method.

Collaborative Hypermedia

Nodes and links in hypermedia have already been found useful in modeling work processes [70, 71]. Hypermedia nodes can model activities, tools, artifacts and people, whereas the links between the nodes can model any relation among these elements. In addition, the popularity of hypermedia as a mechanism to create and publish shared knowledge has grown with the advent of easy to use collaborative hypermedia technology such as WikiWikiWebs.

WikiWikiWebs were first implemented and introduced by Ward Cunningham as a lightweight collaborative composition system for the web, to host the work of the Portland Patterns community. WikiWikiWebs are discussed in detail in [45] and on-line in [75]. This section provides a short overview of the main concepts.

A WikiWikiWeb is a server of hypermedia content rendered in HTML. A link labeled "edit this page" gives access, in the browser itself, to a form in which the source code of the page is presented and can be edited. A save button under

the form submits the changed source code to the server, that in turn returns the newly rendered page. The source code of a page is basically plain text with some simple markup to do formatting and to create hyperlinks. Any group of run-together words (e.g., ScaffoldingWiki) defines a hyperlink to another page within the same WikiWikiWeb. The run-together words represent the title of the target page. Titles must be unique. If the target page does not yet exist, the run-together words are displayed followed by a question mark which is a hyperlink (e.g., ScaffoldingWiki?) [1]. The user who first follows the hyperlink is presented with a form to edit the page. Saving the form creates the new page. Then, the run-together words become the anchor of the hyperlink that leads to the new page. Text formating markup allows the creation of simple logical elements such as bulleted lists, separators, and definitions.

Any number of users can simultaneously create and modify pages in the WikiWikiWeb. The philosophy of a WikiWikiWeb is to trust users in that they will edit responsibly. Pages are versioned on every save. All versions of a page can be accessed and compared via a link in the page footer.

A WikiWikiWeb is implemented as a central server, accessed via the HTTP protocol by regular web-browsers. Each page in a WikiWikiWeb has a unique URL composed of the server's URL and the page's title. WikiWikiWebs are easy to deploy and use.

Specializing WikiWikiWeb for Scaffolding Hypermedia

The idea of WikiWikiWeb can be specialized to support the delivery of scaffolding. Consider the UML class diagram in Figure 4.4 as an object-oriented model for the information contained in a scaffolding hypermedia. This model extends the one presented in Figure 4.3 with information regarding the application of the method in a case. The elements of the scaffolding, namely, activities, tools, artifacts, participants, team members, and shared perspectives, map to pages (i.e., nodes) of the hypermedia.

Each page is identified with a unique title. For activities, tools, artifacts, shared perspectives and team members the title corresponds to the name. For participants, the title corresponds to the role.

Properties of type Text in the diagram contain WikiWikiWeb markup. Including a reference to another scaffolding element as part of the text in a property creates a relation between the objects. The nature of the relation is detailed in natural language in the text that makes the reference. For example, the text for the intent property of the activity named "Reporting" could read "Create the BreakdownReport". This implies a relation between the Reporting activity and the BreakdownReport artifact. If a text property makes a reference to a non-existing element, after saving the changes, the user is given the possibility to choose the type of the new element from a list.

The relations in the UML diagram document the intended semantics of references. References from activities to activities indicate that there is some form of execution dependency between them. References from activities to tools document usage. References from activities to artifacts document that the artifact is developed or needed in the activity. References from activities to participants document participation. References from participants to team members docu-

[1] In this document, underlined text represents hyperlinks

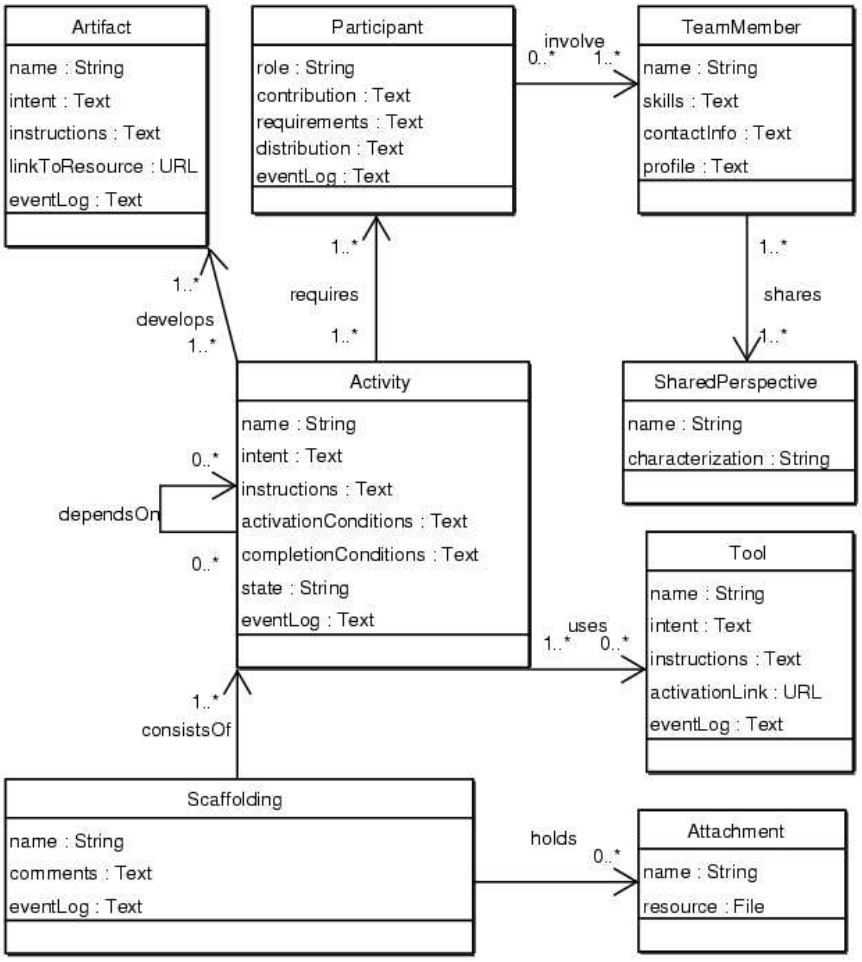

Figure 4.4: Object model of a hypermedia scaffolding

ment involvement. References from team members to perspectives indicate that the team members shares the perspectives.

The definition of a process starts with the creation of an empty scaffolding. A cover page introduces the scaffolding with a title and a description of purpose. Additionally, it provides an index with all the elements in the scaffolding classified by type. A special section provides references to the starting points. Immediately after creation, all fields in the cover page are empty. The user can edit the description of purpose and the title. More important, the user can introduce references to starting points. These can be either activities, tools, artifacts or participants. However, in most cases, the user will reference the initial activities of the process. When the changes to the cover page are saved, the user is presented with a list to choose the adequate type of object for each of the named starting points. Choosing from the list creates the new element and opens it in edit mode. This action results in the scaffolding being populated with the first group of elements. The scaffolding is populated starting from the cover page and by further creating hyperlinks from existing elements to new elements. Every element in the scaffolding is either accessible through the starting points section of the cover page, and/or from a property of another element.

Each element in the scaffolding is shown to the user following the structure presented in Figure 4.5. The details of the element are presented within the placeholder labeled "type dependent content" in the figure. The "edit" button is used to change the interface to edit mode, which allows the user to modify the properties of the element. In edit mode, the placeholder contains text boxes in which to edit the properties of the element.

In presentation mode, anchors for hyperlinks that point to scaffolding elements are followed by a small icon indicating the type of the destination element. Activities, tools, artifacts, and participants that include references to the presented element are listed in individual sections at the bottom of the page.

A section titled "Event Log" at the end of the page is used to list important events that occurred for the element. The system will add entries to the event log automatically (e.g., every time an activity changes its state). Users can create entries via the input field and the button located under the event log. They can use this mechanism to communicate to other users important situations regarding the element, for example, to document daily progress regarding an activity.

The form used to edit an activity includes text fields to enter the activity's name, its intent, a detailed description, and the conditions required to start and to finish the activity. In addition, there is the possibility to indicate the status of the activity, where status is one of *Inactive, InProgress, Canceled, or Completed*. The intent part of an activity provides a short overview of the goal of the activity. Together with the name, the intent part is used to rapidly identify activities.

The description part of an activity should provide all details needed by a newcomer to be able to perform the activity. If an artifact is needed because it provides relevant information for the activity, or if an artifact must be created or updated, a reference to it must be included in the description. The text containing the reference should make the relation to the artifact clear. The same is true for the usage of tools and for the participation of participants.

The activation conditions part of an activity succinctly details all required

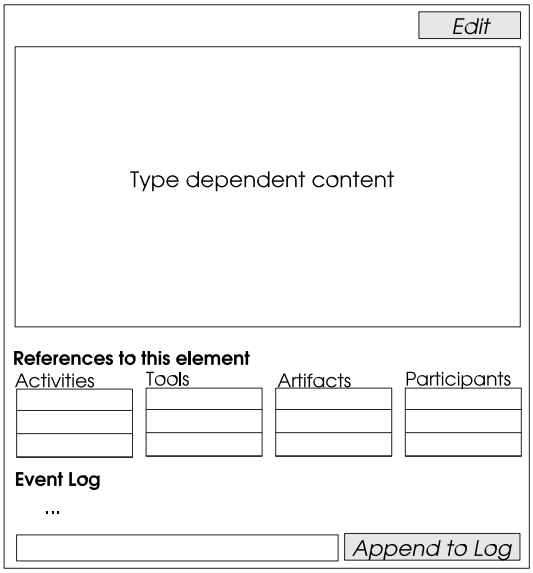

Figure 4.5: Presentation of a scaffolding's element

aspects of the state of the case that are the precondition for the activity to be started. These preconditions will be usually expressed in relation to other activities (e.g., whether they were finished, or have produced significant results). The completion conditions part of an activity documents the conditions that must be fulfilled in order to be able to say that the activity has been completed. For example, the completion conditions part of an activity "Publish" could state: "This activity is completed when the in-box tool confirms that publishing has been successful and notifications have been sent."

The form used to edit a tool includes text fields to enter the tool's name, the tool's intended use, a detailed description of how to use the tool or a link to the on-line user manual, and a link to install and/or activate the tool.

Artifacts are edited by providing a name, an intent, and a detailed description of the structure of the artifact and the rules that apply to its usage and development. A central property of an artifact is the hyperlink reference to the real resource. The artifact element of the scaffolding acts as a placeholder for meta-information about the real artifact available at another location. This level of indirection allows resources that originate in other systems accessible through the network to be integrated into the scaffolding. In the simplest case, the real resource is an untyped page within the hypermedia. Untyped means that the page does not correspond to an element type of the scaffolding; thus it is displayed and edited as a regular WikiWikiWeb page.

A participant node documents a role that need to be fulfilled. The edit form can include a requirement that a person must fulfill in order to fit the role. A general description of the participant contribution to breakdown handling is provided. The exact details of what the participant needs to do are provided as part of the description of the activities that reference the participant. The edit form also indicates team member involvement in roles (i.e., work distribu-

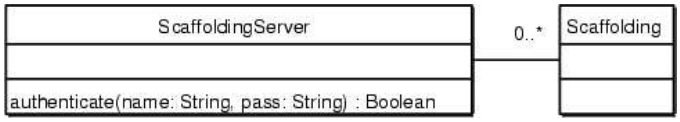

Figure 4.6: The Scaffolding server class

tion). References to the involved team members are included as references to the corresponding team member nodes.

Participation in collaborative tailoring requires, for many activities, an association between team members and shared perspectives (see Section 2.2.1). The node type *shared perspective* is introduced to document shared perspectives. Each shared perspective is identified by a name. The characterization of a shared perspective provides hints, in natural language, for team members to decide whether they share the perspective or not. For example, the characterization of a perspective can make reference to the role of team members in the organization, to the skills of team members, to professional background, or to a given work philosophy.

A team member node documents the name, contact information, and the skills of a team member. There is only one team member node for each team member. Additionally, the edit form for team members provides a text field to enter references to shared perspectives (i.e., their profile). These are the shared perspectives that the team member believes to share.

Shared perspectives are seldom referenced from nodes other than team members. Team members are seldom referenced from nodes other than participants. Therefore, the presentation mode of shared perspectives and team members do not classify references in lists according to the type. Instead, incoming references are presented as a comma-separated list of hyperlinks. The icon that accompanies every hyperlink in the scaffolding wiki provides a hint of the type of node to which the link points.

Access to the scaffolding pages is password protected. Access control is provided through an external authentication and session tracking service such as the Apache Session module [54]. The session tracking service supports single login policies through cookies. Cookies play the role of passports that further web-pages and applications can use to grant access without contacting the authentication service, and without asking the user again for user name and password. All pages in any of the scaffoldings check that the passport is present. Otherwise, the content of the page is not presented. Instead, a login form is presented. The authentication functionality is channeled through the `authenticate` method of the class ScaffoldingServer[2] shown in Figure 4.6. This method encapsulates the interaction with the authentication service and provides a single entry point for login. Figure 4.7 shows the class Passport whose properties are locally stored in the client machine. A logout page handles removing the passport.

User accounts are created by a system administrator. Team member nodes are created together with the user account to keep the scaffolding and the user

[2]The ScaffoldingServer is a Facade object [26] used to describe the public behavior of the scaffolding server

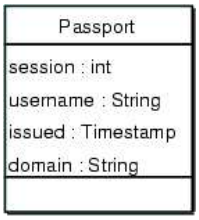

Figure 4.7: Passport object

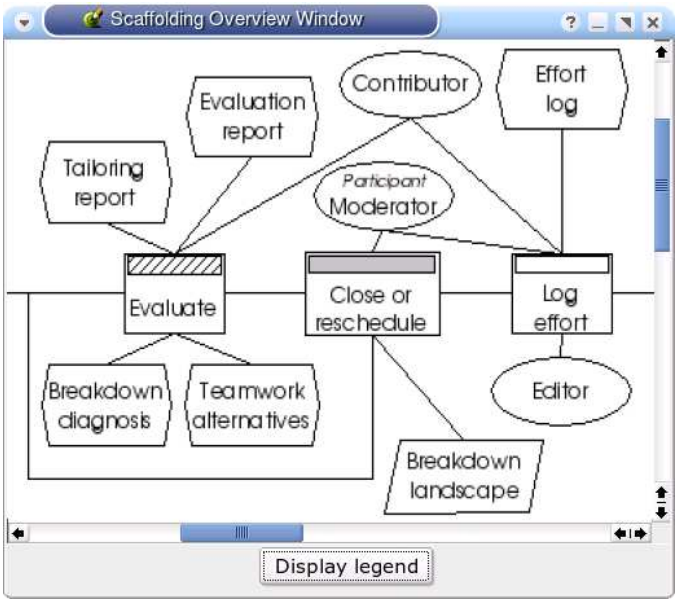

Figure 4.8: Graphical scaffolding overview window

database used by the authentication service consistent.

Enactment of a Scaffolding

To contribute to a case, users browse the scaffolding and learn about what needs to be done. At any time they can sign-up to be notified of changes in pages of their interest. Before the project starts, users usually sign-up to be notified of changes in those activities in which they participate.

To get an overview of the work that is to be done and to check what progress has been made, users activate from the cover page the graphical overview that opens in a separate window (see Figure 4.8). The graphical scaffolding overview displays the elements in the scaffolding as a two-dimensional graph view. Scaffolding elements are represented as nodes in the graph and relations are the edges of the graph. The name (the role, for the case of participants) of the scaffolding element is used to label the node. An attribute of the node (e.g., shape) is used to indicate the type of element. Moreover, an attribute of activity nodes

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT graph (node* link*)>
<!ELEMENT node EMPTY>
<!ATTLIST node
        label CDATA #REQUIRED
        url CDATA #REQUIRED
        id ID #REQUIRED
        type (ACTIVITY | ARTIFACT | PARTICIPANT | TOOL)
node-data CDATA #REQUIRED>
<!ELEMENT edge EMPTY>
<!ATTLIST edge
        from REFID #REQUIRED
        to REFID #REQUIRED>
```

Figure 4.9: Document Type Declaration for the result of the generateOverview method of class Scaffolding

(e.g., color) is used to indicate their state. A button labeled "Display legend" gives access to help on how to interpret the image. The graphical overview also serves as a navigation control. Clicking on a node in the view, causes the browser that launched the view to navigate to the element represented by the clicked node. To obtain the graphical overview, the class Scaffolding is extended with the method `generateOverview`. The method returns an XML representation of a graph that mirrors the scaffolding. The information contained in the XML string follows the DTD presented in Figure 4.9. Each node contains information about the type of scaffolding element it represents, and the URL of the page in the scaffolding hypermedia for the element. The XML string is fed to a new browser window containing an applet capable of laying out and displaying the graph in 2D.

The activities in the list of starting points provide a place to start looking for work. One of the users, the one declared as the project conductor, is responsible for changing the status of these activities to "in progress". Users who have signed-up to be notified of changes in these activities receive a notification. Other users, who prefer a proactive way of working, periodically check the state of the project to find out if their participation is required.

Team members identify the activities they need to contribute to by tracing references backwards from their team member nodes, to participant nodes, to activity nodes. Once users know where they need to contribute, they navigate to the hypermedia nodes that document these activities to look for the details and to do their work. Once the completion conditions of the activity are met, they edit the activity and change the status to "completed". The scaffolding hypermedia provides no support to compute dependencies between activities. It is up to the users to decide how changes in status of activities affect other activities.

The scaffolding hypermedia additionally provides a mechanism to query the list of pages that changed. The result is a report that indicates for each of the recent past days the list of pages that have been changed. This history of changes can be used by proactive users as a guide of where to explore for news.

As a result of some activities, users need to modify artifacts. Artifacts can be easily accessed from hyperlinks embedded in the description of the activities. The event log of the artifacts can be used to share notes regarding the changes made to the artifact.

If a tool is needed in order to contribute to an activity, the node describing it

```
public URL deepCopy(String newName) {
  Scaffolding copy := new Scaffolding(newName);
  for each Activity A in this.consistsOf() {
    A.deepCopyTo(copy);
  }
  for each Attachment T in this.holds() {
    T.copyTo(copy);
  }
  copy.logInstantiationFrom(this);
  this.logNewInstance(copy);
  return copy.asURL();
}
```

Figure 4.10: Pseudocode for the deepCopy operation in the Scaffolding class.

can be easily accessed from the link embedded in the description of the activity. Moreover, if the tool can be started from the web, the node describing it provides the activation link.

Instantiation of a Scaffolding

When a scaffolding is used in a case, its pages are modified to contain case-specific information. The status of the activities reflects the status of the case, the artifacts reflect the content of the artifacts in the case, and the event logs reflect the events that occurred in the case. A scaffolding can describe a way of working that is valid for more than one case. Therefore, it is desired that the experience it documents can be reused. In order to do this, an unused version of the scaffolding needs to be maintained and duplicated every time a case based on it starts. In this thesis, the term "instantiation" refers to the creation of a copy of a master scaffolding for its usage in a particular case. The class Scaffolding in the UML diagram in Figure 4.4 is extended with a method `deepCopy` as pseudocoded in Figure 4.10.

Instantiation of scaffoldings works by creating in the same server an exact copy of a master scaffolding. The copy is accessible in a new name-space identified by a URL composed of the URL of the server followed by an identifier for the new case (e.g., http://scaffolding.myorg.org/new-case/). All elements maintain their original name relative to the URL of the scaffolding. Artifacts from external sources that were stored as attachments in the master are also copied as attachments for the instance. However, artifacts or other on-line resources maintained outside the scaffolding (e.g., accessible via URLs) aren't copied. The references in the instance and the master are to exactly the same resource.

On instantiation, an entry documenting the time of instantiation and a reference to the master scaffolding is automatically appended to an event log presented at the end of the cover page of the instance. The log entry additionally contains a hyperlink to an untyped page where further details about the operation are provided (e.g., the list of references to external resources shared with the master). Additionally, a log entry is appended to the event log in the cover

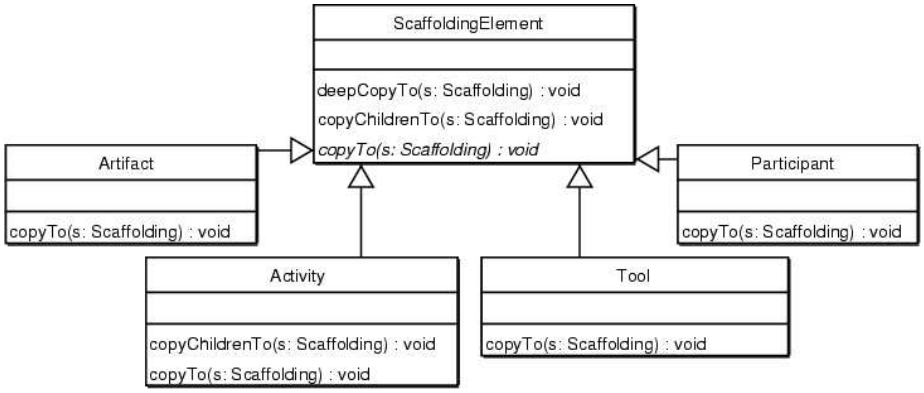

Figure 4.11: A hierarchy of scaffolding elements

page of the master indicating that a new case based on the master has been created, and containing a hyperlink to the new case. The references to the instances from a master scaffolding serve as examples of use.

A hierarchy of scaffolding elements simplifies the definition of the copy operations. Figure 4.11 shows how the class ScaffoldingElement implements the deepCopy method inherited by all subclasses. The method relies on two other methods that subclasses implement or override. The pseudocode of the `deepCopy` method is shown in figure 4.12.

Figure 4.13 provides pseudocode for the class Activity. The class Activity implements the `copyChildren` method to propagate the copying to instances of Participant, Artifact, and Tool. The class Artifact, Participant and Tool do not override the method `copyChildren` because they do not reference further scaffolding elements that need to be copied. The `copyTo` method in class Activity creates a new Activity in the target scaffolding and copies all attributes to it. The classes Tool, Artifact and Participant have a similar `copyTo` method with one variation. Whenever a property is found which points via an URL to an external resource (i.e., not an attachment or untyped page inside the scaffolding), a note is added to the page that holds further details of the copy operation. The note documents that an element is shared between the master and the copy.

Unlike activities, tools, and artifacts, team members and perspectives need to be shared among all instances. If an element representing a new team member is added, it should be usable in all scaffoldings. Similarly, perspectives, and associations between team members and perspectives must be accessible for referencing in all scaffoldings. Therefore, they are created as nodes in an independent WikiWikiWeb namespace called *shared*. (i.e., accessible under http://scaffolding.myorg.org/shared/). References to these shared elements have the form *Shared:nodeName* (e.g., Shared:JohnDoe or Shared:Management). Implementations of the `copyChildren` method should not be propagated to the *shared* namespace.

```
class ScaffoldingElement {
void deepCopy(Scaffolding target) {
  this.copyTo(target);
  this.copyChildrenTo(target);
}

void copyChildrenTo(Scaffolding target) {
  return; //Does nothing.
}

abstract void copyTo(Scaffolding target);

}
```

Figure 4.12: Pseudocode for the deepCopy operation in the ScaffoldingElement class.

```
class Activity extends ScaffoldingElement {
void copyTo(Scaffolding target) {
  Activity copy = target.newActivity(this.name,
  this.intent, this.instructions,
  this.activationConditions, this.eventLog,
  this.completionConditions);
}

void copyChildrenTo(Scaffolding target) {
  for each Participant P in this.requires() {
    P.deepCopyTo(target);
  }
  for each Artifact A in this.develops() {
    A.deepCopyTo(target);
  }
  for each Tool T in this.uses() {
    T.deepCopyTo(target);
  }
}

}
```

Figure 4.13: Pseudocode for the deepCopy operation in the Activity class.

Tailoring Hooks

As a consequence of the need to change the method in response to level B breakdowns, scaffolding support for breakdown handling must provide the tools to collaboratively create, edit, and publish scaffoldings. WikiWikiWebs are by nature a tool to collaboratively edit content. Therefore, the method becomes a shared document that team members can edit in collaboration. The scaffolding server provides features to lock pages, to review changes, to access previous versions of all pages, and to notify team members on change of pages.

The scaffolding integrates tools and artifact templates through hyperlinks. This allows coarse level tailoring by simply replacing tools and templates when needed. If artifact templates are created as page templates within the WikiWikiWeb, finer granularity tailoring (i.e. section by section) is also possible. If artifact templates are imported from external sources, the ability to tailor them depends on the source. Finer granularity in the tailoring of tools needs to be provided by the tool itself.

Instantiation via deep copy as described in the previous section has the benefit of making the copy completely independent from the master. Everything can be tailored before and during use to match the needs of the case. For example, the descriptions of the activities can be edited to fit the background of the participants. As each case works on a completely independent copy of the scaffolding, changes to instances are not propagated. In order to foster the propagation of successful experiences, a mechanism to extract a master from any tailored instance is required.

Extraction of Best Practices

Extracting the practice contained in a scaffolding used in a project implies creating a master scaffolding that mirrors its structure and content, and that can be used for further instantiation. The status of all activities in the master should be set to inactive, all event logs should be cleared, and all modified artifacts should be put to an initial state.

Similarly to the `deepCopy` operation used for instantiation, an `extractMaster` operation is defined. Figure 4.14 provides pseudocode for the `extractMaster` in the class Scaffolding. After extraction, an entry documenting the operation is added to the event log of the new master scaffolding. The entry contains a hyperlink to the project scaffolding that served as the source. A page with additional information about the operation lists the resources that may require manual processing, and the resources that are shared with the source scaffolding.

Figure 4.15 and Figure 4.16 specify in pseudocode the `extractMasterTo` operation for the classes ScaffoldingElement and Activity. To extract the master of an activity, a new activity in instantiated. By default, activities are instantiated with inactive state. Moreover, the new instance is passed all attributes of the original activity except the content of the event log, which is initialized to an empty string. The method `extractMasterTo` is similarly implemented in the classes Participant, Artifact, Tool.

Attachments, external resources (URL outside the namespace of the Wiki), and untyped pages all pose a limitation when using the copy mechanism for extracting practice. The system has no knowledge about the structure and content of these artifacts; therefore it is not possible to implement a `extractMasterTo`

```
public URL extractMaster(String newName) {
  Scaffolding master := new Scaffolding(newName);
  for each Activity A in this.consistsOf() {
    A.extractMasterTo(master);
  }
  for each Attachment T in this.holds() {
    T.copyTo(master);
  }
  master.logExtractionFrom(this);
  return master.asURL();
}
```

Figure 4.14: Pseudocode for the extractMaster operation in the Scaffolding class.

```
class ScaffoldingElement {
void extractMasterTo(Scaffolding masterScf) {
  this.copyAsMasterTo(masterScf);
  this.extractChildrenTo(masterScf);
}

void extractChildrenTo(Scaffolding masterScf) {
  return; //Does nothing.
}

abstract void copyAsMasterTo(Scaffolding masterScf);
}
```

Figure 4.15: Pseudocode for the extractMaster operation in the ScaffoldingElement class.

```
class Activity extends ScaffoldingElement {
void copyAsMasterTo(Scaffolding masterScf) {
  Activity copy = masterScf.newActivity(this.name,
  this.intent, this.instructions,
  this.activationConditions, "",
  this.completionConditions);
}

void extractChildrenTo(Scaffolding masterScf) {
  for each Participant P in this.requires() {
    P.extractMasterTo(masterScf);
  }
  for each Artifact A in this.develops() {
    A.extractMasterTo(masterScf);
  }
  for each Tool T in this.uses() {
    T.extractMasterTo(masterScf);
  }
}}
```

Figure 4.16: Pseudocode for the extractMaster operation in the Activity class.

method that clears specific sections. Untyped pages and resources from external sources stored as attachments are copied without any change. Artifacts or other on-line resources maintained outside aren't copied, but referenced. After extraction, the user must check all copied artifacts and all shared resources to see if the content needs to be (re)set to initial values.

4.2.3 Summary

The scaffolding approach described in this section supports the informed and coordinated participation of team members in breakdown handling. It is applicable to handle breakdowns in everyday work. Moreover, it is also applicable to handle breakdowns that occur during breakdown handling, which this requires that the method specified in the upcoming sections is also valid in such cases. The scaffolding approach provides tailoring hooks to enable tailoring of the process in any form, to replace tools, to tailor and replace artifacts, and to tailor participation in the process. This represents a first step for the fulfillment of requirement 1, namely, the provision of support for collaborative breakdown handling at any level.

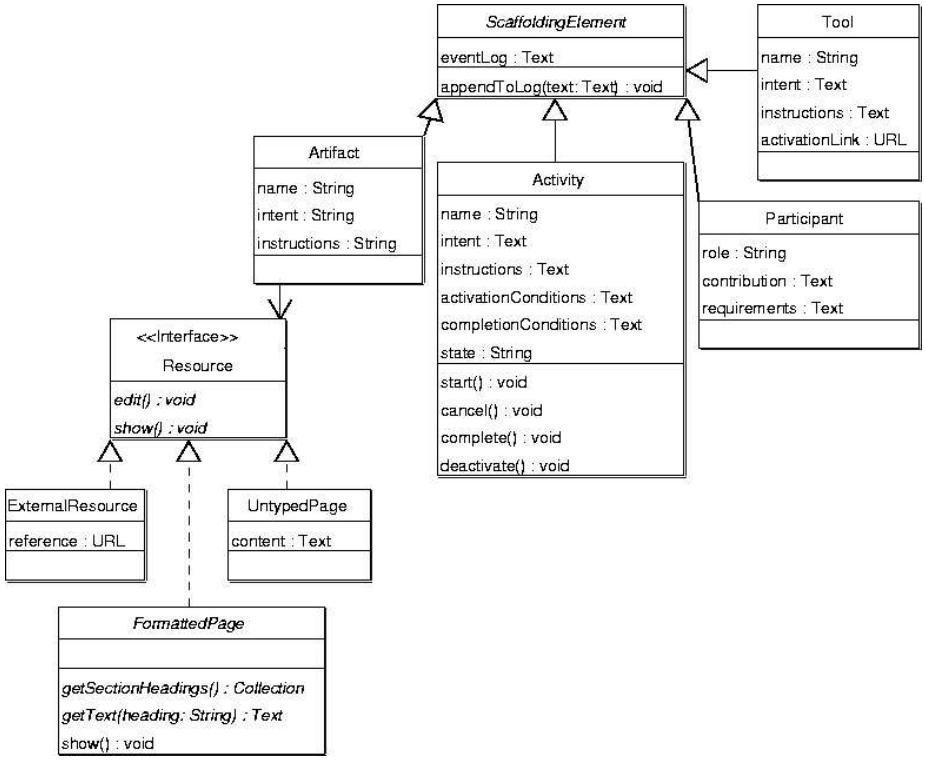

Figure 4.17: Extended ScaffoldingElement hierarchy

4.3 Triggering Breakdown Handling

4.3.1 Presentation of the Method

Sections 4.3 to 4.8 presented the breakdown handling method proposed in this thesis. The objective of these sections is twofold. They document the instructions that will be delivered to users as a manual of operations and procedures. This manual includes instructions to perform the activities, to use the tools, to create artifacts, and to organize participation. Additionally, they specify the data structures that are required to store and deliver artifacts, and specify the data structures and functionality required by the individual tools.

Each section starts with a graphical overview of the phase, covering organization of activities, the use of tools, the creation of artifacts, and participation. One subsection is dedicated to each of the elements in the phase. The structure of each subsection varies with the type of element it describes. Subsections present the text that corresponds to the static properties of the corresponding element class in Figure 4.17. A static property of a scaffolding element does not change when the scaffolding is deployed, instantiated or enacted in a case. The event log property, the state of an activity node, and the distribution of a participant node are examples of non-static properties.

Participant subsections are further divided into **role**, **contribution** and **requirements**. The abstract class ScaffoldingElement defines the method `appendToLog`.

This method guarantees that users can only append to the log but never remove. Moreover, the method automatically timestamps each entry.

Activity subsections are further divided into **name**, **intent**, **instructions**, **activation conditions**, and **completion conditions**. The content of the subsection *state* can only be set through one of the state change methods in the Activity class. This restriction ensures that the value matches one of the possible states. The method `start` sets the content to "InProgress". The method `cancel` sets the content to "Canceled". The method `complete` sets the content to "Completed". The method `deactivate` sets the content to "Inactive".

Tool subsections are further divided into **name**, **intent** and **instructions**. The instructions subsection indicates how to use the tool. Moreover it also specifies the underlying data structures and operations needed to build the tool. Tools that are used along several breakdown handling phases, are incrementally introduced.

Artifact subsections are further divided into **name**, **intent** and **instructions**. The instructions subsection additionally describes the structure of the artifact. In the UML model presented in Section 4.2, Artifact instances had a URL reference to the real resource (the artifact itself). The resource could be an external document or an untyped page in the scaffolding hypermedia. To facilitate the creation of artifact templates within the scaffolding hypermedia, the object model is extended with the interface Resource and the abstract class FormattedPage. ExternalResource and UntypedPage are implementations of the Resource interface. They can be edited and shown. FormattedPage implements Resource to specify two operations that templates should implement. The two operations are used to provide a default implementation of `show`. The operation `getSectionHeadings` returns a collection with all section headings in the template. The operation `getText` returns the content of the section whose heading matches the argument. In the following sections, when a template for an artifact in the method is introduced, it is presented as an implementation of the FormattedPage interface.

4.3.2 Overview of the Triggering Phase

The goal of the triggering phase is to create awareness among team members about the existence of breakdowns that need to be handled.

Figure 4.18 provides an overview of the triggering phase. The notation in the figure will be used in the next sections to provide an overview of a phase. Rectangles represent activities. Hexagons represent tools. Parallelograms represent artifacts. Ovals represent participants. The arrows represent the flow of control. Solid lines that connect activities to tools document usage. Solid lines that connect activities to artifacts indicate that the artifact is created or needed in the activity. Solid lines that connect activities to tools indicate that the tool is needed to perform the activity.

The triggering phase consists of five core activities: report, weight, publish, select, and aggregate. The key artifact in this phase is the breakdown report. It is created during the report activity, and it is used in the other four activities. Participation in the triggering phase is done in the roles of reporter and manager. Two tools are suggested. The Breakdown Landscape is used to capture the relation among forces and teamwork. The breakdown in-box holds the list of breakdowns that wait to be handled. During the weight activity, the report is

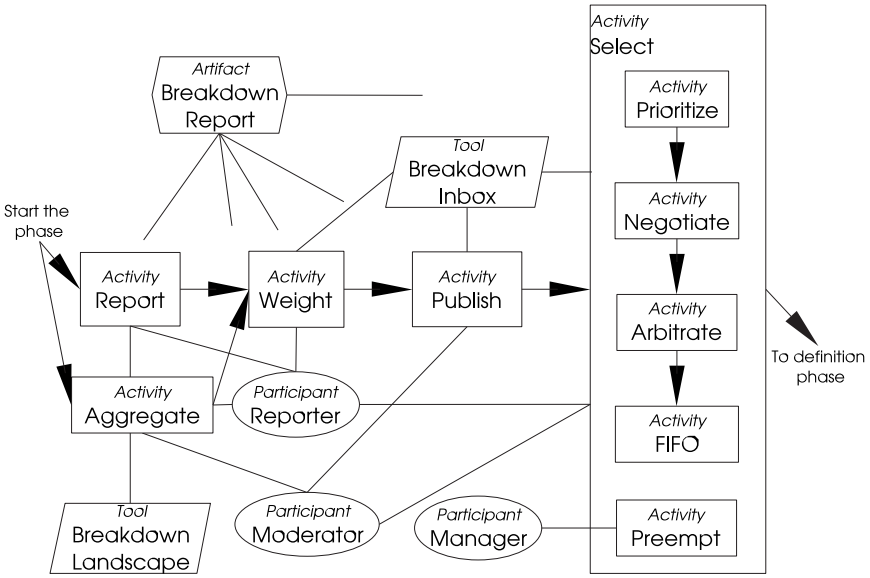

Figure 4.18: Triggering phase of breakdown handling. The artifact breakdown report is related to all activities in this phase.

extended with a preliminary evaluation of the impact of the breakdown. The publish activity releases the report to all team members, making them aware of the breakdown. When team members are ready to handle a breakdown, they explore the list of reported breakdowns in order to select one breakdown for handling. Once a breakdown is selected, the triggering phase ends and the definition phase starts. At any moment during the triggering phase, reporters of related breakdowns can collaborate to create an aggregated report that covers all related breakdowns.

4.3.3 Participant: Reporter

Contribution

The team members that experience the breakdown take the role of *reporter*. The role of reporter is assigned during the report activity of the triggering phase, and never reassigned. Reporters have the final word regarding the definition of the problem, and the evaluation of the solution.

Requirements

Any team member can be a reporter, regardless of skills and perspectives.

4.3.4 Participant: Moderator

Contribution

The moderator is the official speaker for the breakdown. The moderator can always transfer the role to another team member.

The moderator is responsible for tracking the progress of the breakdown handling process. The moderator is responsible for changing the status of activities to *InProgress*, when activation conditions are met; to *Completed*, when the completion conditions are met; and to *Cancelled*, when prescribed by the method. The moderator takes care that breakdown handling moves forward.

Besides the general responsibilities of the moderator previously stated, the moderator is explicitly requested to contribute to certain activities such as the publish activity of the triggering phase.

Requirements

The team member who is the first to contribute to handling of the current breakdown (i.e., the first reporter) additionally takes the role of moderator.

4.3.5 Activity: Report

Intent

To create a detailed report of the perceived breakdown, and initiate breakdown handling.

Activation conditions

It is always possible to report a breakdown. Initiating a report activity implicitly involves initiating a new breakdown handling process.

Instructions

Reporting starts with the creation of an empty breakdown report (Artifact in Section 4.3.6) [3].

Upon creation of a report, the scaffolding system instantiates a breakdown handling process. All artifacts in the scaffolding are initialized using the given templates, and all roles are created but not assigned.

The team members that report add references from the reporter node to their team member pages. If the team members' details have not been entered in advance[4], they need to be entered at this point. Besides team member name, contact info, and skills, it is necessary to provide references from team member pages to at least one shared perspective node. Shared perspective nodes are created on demand.

[3] When the method is deployed electronically via the scaffolding server, these cross-references become WikiWikiWeb hyperlinks. For example, the previous reference "... breakdown report (Artifact in Section 4.3.6) ..." becomes breakdown report, pointing to the node BreakdownReport which holds the content presented in Section 4.3.6.

[4] Team member pages are created when user accounts are opened (see Section 4.2.2). However, no data other than user name is entered.

Isolating the key elements of teamwork, and of the products that relate to a breakdown, is valuable for later phases of breakdown handling. Therefore, the reporters (Participant in Section 4.3.3) are asked to explicitly identify these elements. The context and symptoms section of the report must explicitly indicate the role of activities, tools, artifacts, and communication in the breakdown. The BreakdownLandscapeTool (Tool in Section 4.3.7) is used to create explicit references from the breakdown report to the relevant elements in teamwork and the product.

Completion Conditions

This activity is completed when the reporters consider that they have filled in the sections of the report related to *context*, *symptoms*, and all elements in the *key elements* are cross referenced from the context and symptoms. The moderator, in agreement with the other reporters, should mark the activity as completed.

4.3.6 Artifact: Breakdown Report

Intent

The breakdown report documents the perceived problem, its perceived weight, and its relation to teamwork. The report describes the breakdown so all team members can be aware of it. It forms the basis for further discussion.

Instructions

The breakdown report is a document with four mandatory sections, namely, the context, the symptoms observed by the reporters, key elements of teamwork, and an initial indication of the importance of handling the breakdown (i.e., its weight).

The breakdown report is created from a template provided by the scaffolding service described in Section 4.2.2. The template contains one text area for each of the four sections. The user enters text in the text areas according to a predefined format. When the artifact is saved, the text fields are parsed and the data objects in Figure 4.19 are updated. The `getText` method is implemented to send the appropriate `get` message (e.g., `getContext`) that returns the text to be shown. When the page is saved after an edit, the `parse` messages are sent to transform the entered text into the underlying data objects.

The reporter documents the context and the symptoms of the breakdown in natural language. The context answers the questions: what was the reporter doing or attempting when the breakdown occurred? and what was the status of work at the moment of the occurrence of the breakdown? The description of symptoms answers the questions: what did the reported experience that interrupted the normal flow of work? and what was expected instead? Symptoms are documented in the form of a bulleted list. Consecutive text lines starting with a hyphen become elements in the list.

The description of context and symptoms should make references to tasks, tools, artifacts or communication forms. The Breakdown Landscape tool can be used to create a selection of elements of teamwork that can be inserted in

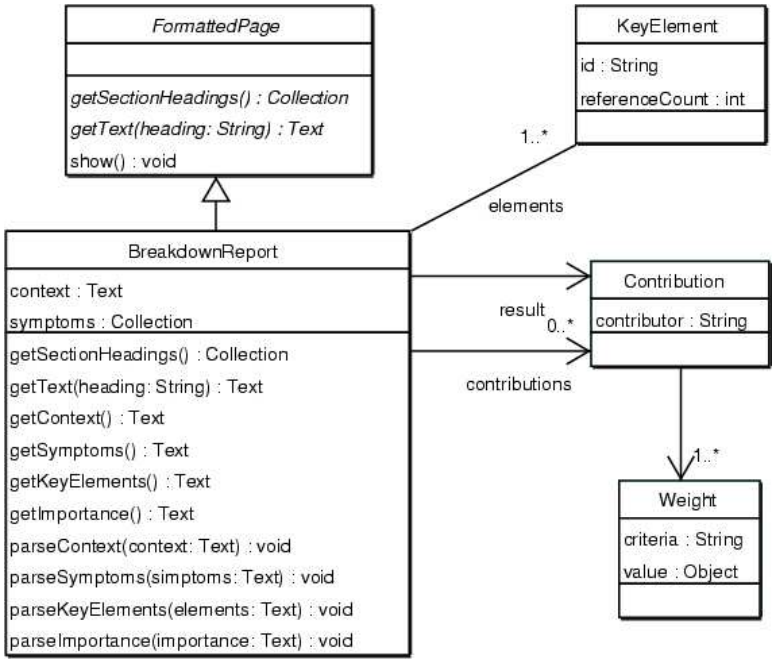

Figure 4.19: FormattedPage subclass for the breakdown report artifact

the *key elements* section of the report as a numbered list. The reporter can cross reference, from the context and symptoms sections, elements of the list of key elements using $\backslash ref\{id\}$ cross-references, where id is the identification of the element in the Breakdown Landscape. Moreover, when the report is saved, elements in the list that are not referenced are highlighted.

The `getKeyElements` method returns a Text with a numbered list constructed from the collection of key elements. WikiWiki formatting is used to highlight the elements that have no references. The methods `parseContext`, `parseSymptoms`, and `parseKeyElements` update the list of key elements and reference count of the key elements from the text entered in the corresponding fields. The implementation of `getContext` and `getSymptoms` replace $\backslash ref\{id\}$ cross-references to cross-references using the order of the element in the numbered list of key elements (i.e., as in bibliographic citations).

The importance of the breakdown is indicated criteria by criteria. Each individual contribution is provided in a separated line following the pattern *criteria-1:weight-for-criteria-1i;... criteria-n:weight-for-criteria-n #name-of-contributor*. The list of contributions ends with a line that aggregates (e.g., calculates the mode) all individual contributions, criteria by criteria. This last line follows the same pattern, with "Aggregates result" as the author name. The content of this section is prepared with the In-box Tool (see Section 4.3.13) and directly submitted to the scaffolding server. The criteria and possible weights are determined when the tool is configured. If the in-box is not used and the section is filled in manually, the `parseImportance` checks that the format is correct and updates the object model. However, this would require that the moderator

manually calculates the aggregated result.

4.3.7 Tool: Breakdown Landscape

Intent

The Breakdown Landscape of an organization holds a set of graphical representations of teamwork in the form it currently takes place, and of the product. These representations are the basis for documenting how forces in breakdowns relate to teamwork and the product. During breakdown handling, the Breakdown Landscape can be used to embed references to elements of teamwork and the product in documents, to document forces, and to perform queries regarding forces, perspectives, teamwork, and breakdowns.

Instructions

The Breakdown Landscape tool implements functionality useful for all phases of breakdown handling. The following sections present the functionality of this tool needed for the triggering phase. Later sections complete the tool's specification.

Diagrams of Teamwork and Product

Whittaker and colleagues [74] proposed to classify contributions to an on-line conversation in artifacts, prose and diesis. Artifacts include data structures such as tables, diagrams, and matrices. Prose is all textual conversation not part of an artifact. Diesis refers to actions that reference existing materials, for example, pointing, voting, and drawing. Whittaker and colleagues observed that artifacts carry the content of the discussion, whereas prose and diesis are used as part of the process of conversation in activities such as discussion, clarification and negotiation about the content. Experiments showed that even in the presence of a speech channel, participants continued creating artifacts. Collaborative tailoring requires that team members engage in on-line conversations supported by groupware tools. For example, team members participate in conversations to identify the causes and the involved forces of a given breakdown. Artifacts (e.g., an organizational diagram or an UML diagram of the product) can be used to focus conversations that are part of breakdown handling. The Breakdown Landscape tool holds the set of diagrams (i.e., artifacts) that team members use as artifacts to focus communication.

There exists a rich spectrum of tools that can be used to create artifacts that represent different aspects of teamwork and the product. Some of these tools are based on standard graphical languages such as Petri nets [68], or on UML extensions [60]. Other tools propose their own graphical language. In many cases, the diagrams are interpreted by the tool (or by other systems) to provide some form of work management support. This is the case of diagrams created with graphical workflow editors that are interpreted by workflow management systems. Using the diagrams as input for computation requires a high degree of detail and formality on the graphical language.

There are general diagramming tools commonly used to create descriptions of work that are to be interpreted only by humans. The languages provided may resemble a standard, formal notation. However, these tools have lighter (if any)

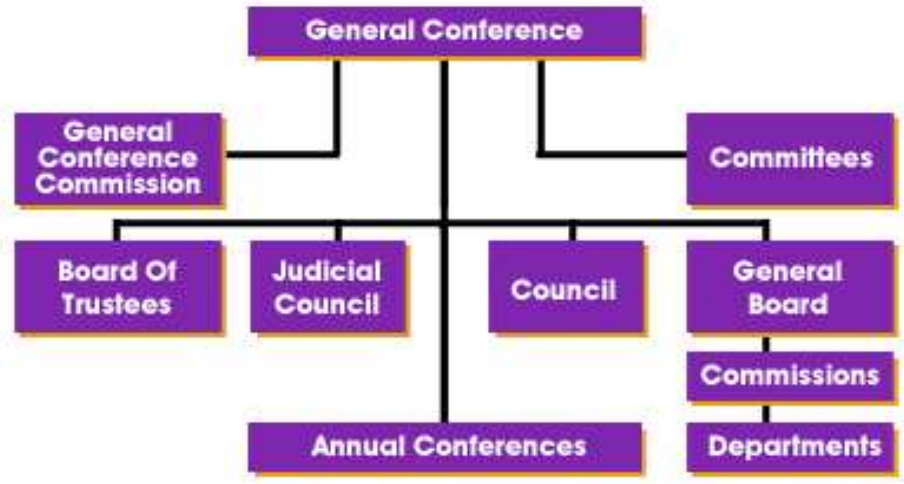

Figure 4.20: Example organizational chart

requirements for details, completion, and formality of the created diagrams. The main goal of a diagramming editor is to provide the graphical constructs of one or more languages, and to assist the user by simplifying the creation of commonly needed arrangements. Microsoft's Visio [5] is a widespread tool that belongs to this family and that supports a variety of frequently used languages. Some example languages supported by Visio allow the creation of diagrams to represent office arrangements, product flows, and organizational charts. Figure 4.20 shows an example diagram of an organizational chart.

From the research arena of visual languages there are tools that allow the definition of formal visual languages and the generation of the corresponding visual editors. GenGed [11] is an example of this type of editors. GenGed additionally supports the simulation of the behavior of visual models. Similarly, the Boz system [18] allows the definition of editors for arbitrary diagramming conventions. The Boz system additionally provides mechanisms to indicate how diagrams are to be interpreted and represented in the form of formal data constructs.

In a group of professionals, or in a work team informal diagramming conventions frequently emerge. Diagrams are created by hand, either on paper or by means of very generic drawing programs (e.g., using basic geometric constructs). The interpretation of these languages is based on agreed upon conventions and shared background knowledge (see [18] for further discussion on the use of diagramming conventions). These languages are used in scenarios with low requirements for formalism. Details are commonly left out of the diagrams in order to simplify them. The risk of misinterpretation associated with the lack of formality of the diagrams and the simplification of details is reduced through verbal clarification and agreement.

Any of the above discussed graphical editors can create the diagrams needed by the Breakdown Landscape. However, they must be capable of exporting the

[5] Visio is a registered trademark of Microsoft Corp.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT teamwork-descriptor (object-representation*)>
<!ATTLIST teamwork-descriptor
        id ID #REQUIRED
        title CDATA #REQUIRED
        image-uri CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
<!ELEMENT object-representation (shape-path+)>
<!ATTLIST object-representation
        object IDREF #REQUIRED>
<!ELEMENT shape-path (point+)>
<!ELEMENT point EMPTY>
<!ATTLIST point
        x-coordinate CDATA #REQUIRED
        y-coordinate CDATA #REQUIRED>
<!ELEMENT object EMPTY>
<!ATTLIST object
        id ID #REQUIRED
        name CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
```

Figure 4.21: Document Type Declaration used by the Breakdown Landscape tool

diagrams in a format that includes meta-information about the elements being represented as required by the Breakdown Landscape tool.

Representations of Teamwork and Product

In a Breakdown Landscape any diagram of teamwork serves as the basic map that is enriched with information about breakdowns. The map is interpreted as an open space where the graphical constructs that represent important abstractions are delimited by two-dimensional areas. For example, in the organizational chart in the left of Figure 4.20, each organizational unit is represented as a rectangular box which in turn occupies a rectangular area.

The information contained in a diagram of teamwork is prepared for its usage in the Breakdown Landscape tool as an image plus meta-information about the image's content. This meta-information consists of a list of the important abstractions represented in the diagram and the area each of them occupies in the image. Most of the previously mentioned diagramming tools are capable of exporting the diagrams in some standard graphical format (e.g., jpeg, or gif). In order to enable use of arbitrary diagramming tools, while at the same time keeping the Breakdown Landscape tool simple, the meta-information about the image's content must be described in conformance to a well-defined convention. An image together with the associated meta-information form a teamwork representation.

The document type definition (DTD) specified in Figure 4.21 defines the format used by the Breakdown Landscape tool to import representations of teamwork or products. The XML documents created from this definition are referred to as representation descriptors. In a descriptor, a representation is defined with a unique identifier. The diagram is specified as an Internet URI (Uniform Resource Identifier). The diagram must be a rasterized image (pixel matrix, non-scalable) to ensure consistency with the areas indicated for each of the objects in the representation. Optionally, an URI can be provided to indicate where to find reference material for the representation (e.g., a detailed

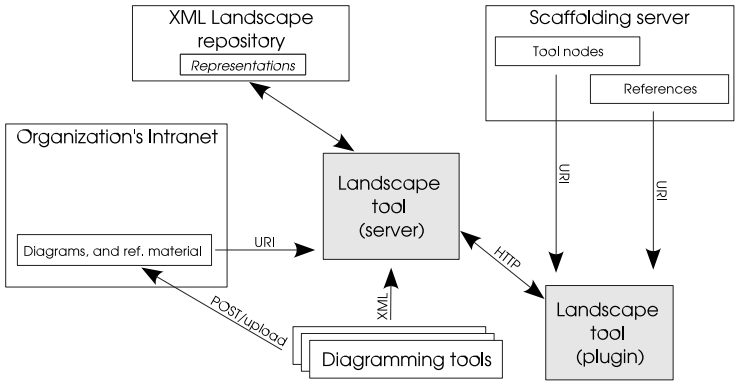

Figure 4.22: Initial architecture overview of the landscaping tool

textual description). Each of the objects represented in the diagram is specified by indicating its unique identifier, an optional URI to reference material, and the list of polygonal areas in the image that are covered by the object. Each of these polygonal areas is described as a sequence of two-dimensional points which are interpreted as the vertexes of a polygon. The possibility of indicating more than one area for a single object is provided as an elementary mechanism to specify composed objects. Unique identifiers are not necessarily cryptic values. On the contrary, it is recommended to provide brief but meaningful strings. The name of the object, and the title of the representation are good candidates for identifiers if it can be ensured that they are unique.

The generation of the descriptor of a representation that conforms to the indicated convention can be implemented by extending the diagram editors with specific functionality for that purpose. MS Visio, for example, foresees the possibility of programming extensions through macros or plug-ins. Implementing such an extension requires the ability to save the image where it can be accessed via a URI, to identify the bounding areas for each of the objects, to extract the properties of these objects (i.e., id and reference URI), and to generate the XML file in conformance with the DTD.

To enable the use of editors that cannot be extended, an additional tool that allows the specification of areas over the rasterized image generated by the editor can be used. Such a tool would work similarly to an editor of client-side HTML image maps ([59], chapter 13) . The disadvantage of this approach is that the map (thus the representation descriptor) must be manually updated every time the associated diagram changes.

Architecture

Figure 4.22 provides an overview of the initial[6] architecture of the landscaping tool. It shows the breakdown landscape tool (with gray background) with its main components, and around it the systems that interface with the tool. The Breakdown Landscape tool has two components, the server and the plug-in.

[6]Other features of the tool are discussed in later sections

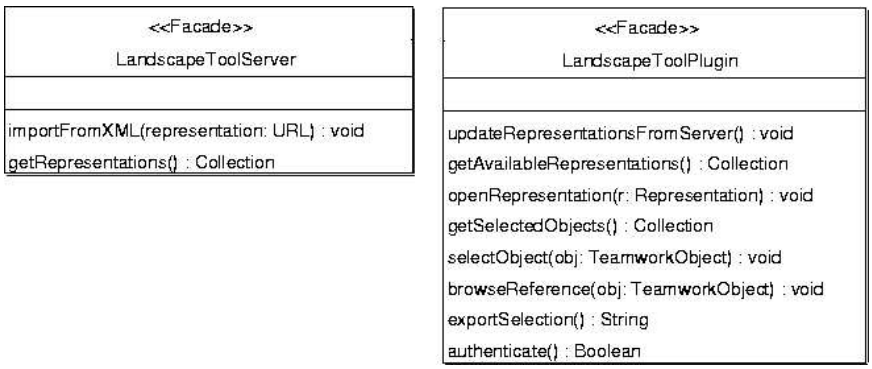

Figure 4.23: Landscape tool: functionality provided on the the server and on the client

The tool is designed following the philosophy proposed earlier, which is to provide small (core) units of data and functionality that can be used to achieve a particular teamwork objective. The landscape (as a document) is the data unit on which the tool is based.

Figure 4.23 specifies the tool functionality. The Facade pattern [26] is used to simplify the specification. One Facade class specifies the operations published by the server, and one class specifies the operations published by the client.

The `importFromXML` method on the server component provides functionality to import new representations that were created with external editors. The server stores the meta-information about the representations in the XML landscape repository. During import, a copy of the diagrams is created and stored in the repository to make sure the diagrams remain available. If a representation has links to reference material (e.g., detailed explanations of tasks), the reference material is made available through the organization's HTTP servers. If the availability of reference material changes, representations needs to be re-imported to update the repository.

The plug-in component delivers the functionality required to explore and use representations during breakdown handling. The plug-in is deployed by embedding it in the corresponding tool nodes of the scaffolding. Thus, when team members reach the report activity, the tool can be started from the associated tool page. A method `authenticate` checks that a passport indicating that the user has authenticated is present. A method `getRepresentations` on the server returns the collection of representation objects used by the client. When the plug-in opens, the method `updateRepresentations` on the plug-in retrieves the representations from the server and updates its local state.

For the triggering phase, the landscape provides functionality to browse all available representations and to simultaneously open several representations. As shown in Figure 4.24, each representation is displayed in an individual tab. There are menu actions to browse the list of available representations (corresponding to the `getAvailableRepresentations` method) and to open them (through the `openRepresentation` method). A menu action corresponding to the `browseReference` method supports navigation in a web-browser to the reference material of a representation or of an individual element in a representation.

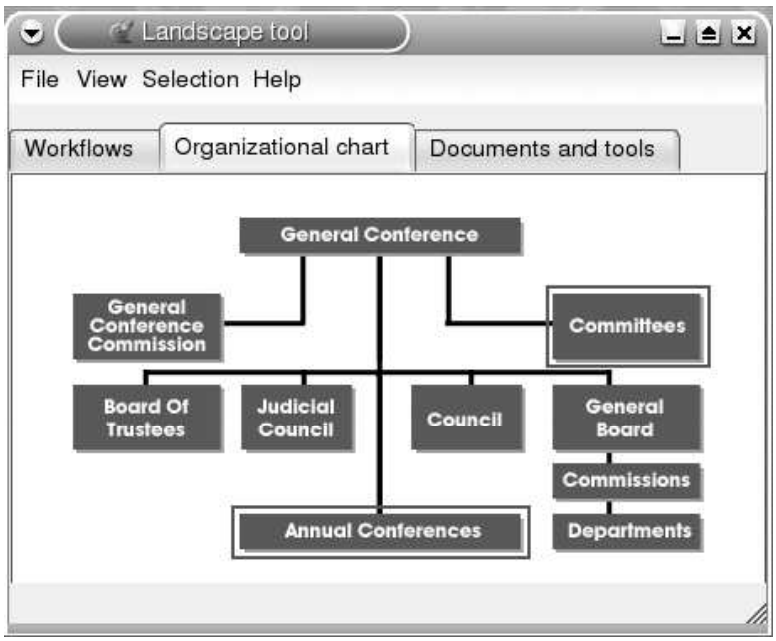

Figure 4.24: Landscape tool: referencing elements from representations

Elements in the representations can be selected. Clicking on an element visible in a representation adds the element to the current selection (through the `selectObject` method). The selection can span multiple representations. Elements in the list of selected objects, which can be obtained through the `getSelectedObjects` method, are highlighted in all representations where they are present. There are menu actions available to operate on the selection, for example to clear it, to invert it, and to remove single elements by changing the effect of mouse clicks.

Once the user has selected the elements from the existing representation that are considered important for a breakdown report, a menu action copies a string version of the selection to the system clipboard. The method `exportSelection` returns the string that is copied to the system clipboard. Then, the content of the clipboard is pasted in the key elements section of the report. Each element in the list appears in a new line in the String.

The exported list ends with a HTML hyperlink that can be used to open the landscape tool and display the selection. The hyperlink is a request to the Landscape tool's server, where the selected elements are passed as arguments as in this example:

<a href="http://svr-addr/landscape.cgi?op=browse&selected=id1+id2+id3">

Reification of Support for Level B Work

An organization commonly works with two breakdown landscapes. One of them is used to analyze work on level A, the other to analyze work on level B (i.e., breakdown handling). The representations used to describe level A work are specific for each organization. They need to be initialized on deployment and updated as needed by the

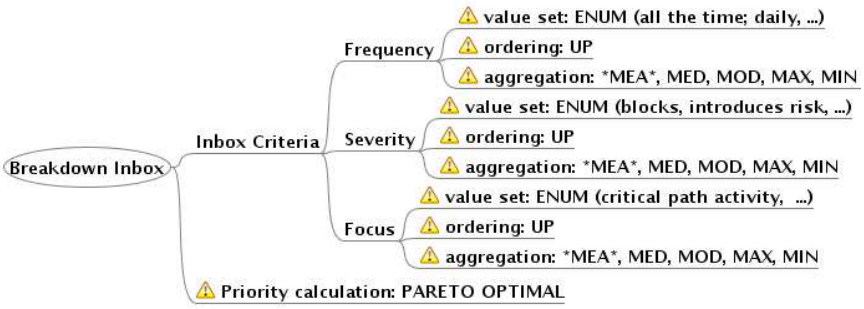

Figure 4.25: Default representation for the Breakdown In-box tool

team. However, many of the representations of work on level B can be obtained from this thesis and the proposed tools and artifacts. There is a process representation that reflects the method, there is an organizational representation that lists and describes the available perspectives, there is a representation of the structure of each artifact, and there is a representation of each tool, specially detailing all tailoring hooks. To simplify the creation and maintenance of these representations, they are generated automatically by the tools.

The scaffolding server presented in Section 4.2.2 provides a graphical overview window that shows, in the form of a two dimensional graph, artifacts, tools, participants, and activities. A menu action in the graphical overview window allows the overview to be exported as a representation. The two-dimensional graph is exported as an image, and the information about the elements in the image is exported as a data file that complies with the DTD of Figure 4.21 in Section 4.3.7.

The scaffolding server cares for the generation of default representations for the scaffolding. Similarly, each tool must provide functionality to export a default representation of itself. The default representation provides an overview of the tailoring hooks of the tool and the choices effective at the moment of the export. Figure 4.25 provides an example of a default representation for the Breakdown In-box tool presented in Section 4.3.13. Tailoring hooks, labeled with a triangle in the figure, are grouped in system features, aspects or components. For example, the frequency aspect of the inbox criteria has three tailoring hooks, namely value, ordering and aggregation. In addtition to the default representations, tools developers can attach UML use case diagrams to the tools (if available) to be exported as representations that users can interpret.

Tailoring Hooks

When the tool is deployed, the system administrators import an initial set of representations. These representations are mined from the diagrams used by the management to describe the organization, and from diagrams obtained in interviews with some representative team members. Ideally, informal drawings used in past meetings are also used to inspire representations.

Any set of representations is valid as long as team members find it accurate. Changes in the organization and/or its context that were not driven by the breakdown handling support system may cause a mismatch between representations and reality. Eventually, such mismatch is perceived by team members as a level B breakdown that needs handling. Handling the breakdown should result in an updated set of representations. The teamwork landscape allows new representations to be imported and existing ones to be removed.

When a new representation is created, attention must be payed to the existence of elements of teamwork in the new representation that were already present in any of the existing representations. An element that is present in several representations must be identified in all of them with the same *id*.

4.3.8 Activity: Weight

Intent

The intent of the weight activity is to obtain an initial indication of the importance of the breakdown that can serve to prioritize breakdowns.

Activation Conditions

The weight activity can start as soon as the report activity (Activity in Section 4.3.5) is completed and the aggregate activity (Activity in Section 4.3.10) is completed or has not been started.

Instructions

This activity results in an initial value for the importance of the breakdown. The value is obtained with the help of the Breakdown In-box tool (Tool in Section 4.3.13) and is documented in the corresponding section of the breakdown report (Artifact in Section 4.3.6).

All team members that fulfill the role of reporter (Participant in Section 4.3.3) of this breakdown must provide a weight. There are N criteria for weighting the impact of a breakdown. The criteria have been defined in advance by the system administrators in agreement with the organization's management. For each criteria there are bounds to its value (enumerated types, or range of integers or real numbers). Reporters must provide a value for each criteria. The weight for a criteria indicates how severely the breakdown impacts that criteria. The list of the contributions from all team members is accompanied with an aggregated result. This is obtained, for example, by averaging the contributed values for each criteria. The final choice of of aggregation mechanism depends on the type of values that can be entered (see Section 4.3.13 for further discussion). The aggregated value needs to be updated after each individual contribution. If the in-box tool is used (expected case), the calculation is done automatically. Otherwise, the moderator needs to aggregate the results manually.

Completion Conditions

This activity is completed when the last reporter has contributed weights and has indicated this fact in the event-log.

4.3.9 Activity: Publish

Intent

A breakdown report was being created and is now ready. The intent of the publish activity is to inform other team members about the existence of a new breakdown report.

Activation Conditions

The publish activity can be started as soon as the weight activity (Activity in Section 4.3.8) has been completed.

Instructions

Only the moderator (Participant in Section 4.3.4) contributes to publish. The breakdown report (Artifact in Section 4.3.6) has been completed and other team members need to be informed about a new breakdown being ready for selection. The moderator sends an e-mail to all team members (possibly via a distribution list). The e-mail must include the URL of the breakdown report, and the URL of the starting page of the scaffolding for the breakdown.

Completion Conditions

This activity is completed when the e-mail is sent.

4.3.10 Activity: Aggregate

Intent

The intent of this activity is to combine the current report with other tightly related, incomplete breakdown reports.

Activation Conditions

Aggregation is possible as soon as the report activity starts (Activity in Section 4.3.5), and as long as it is in progress. Aggregation is no longer possible if the report activity is completed.

Instructions

All team members that contribute as reporters (Participant in Section 4.3.3) should also contribute to this activity. Their responsibility is to pay attention to all other breakdowns in the process of being reported. As soon as they realize the existence of another breakdown report in progress that could be tightly related, they express it using the event-log of this activity. The comment should include the URL of the other report, an indication of the observed similarity, and contact information of the reporter making the comment.

The moderator (Participant in Section 4.3.4) must periodically check this activity for changes in the event-log (or sign-up to be notified of changes). When a comment is appended, the moderator must contact the moderator of the potentially related breakdown in an attempt to aggregate both reports into one. The moderator must react fast to observations of similarities because aggregation is only possible as long as the related breakdown reports are not completed.

If moderators of the related breakdown reports (Artifact in Section 4.3.6) agree on the value of aggregating the two reports, they proceed accordingly. They decide which report to take as the basis, and integrate the information contained in the other one. The report activity of the aggregated breakdown continues normally whereas the report activity of the other breakdown is canceled, and a note is added to the event log of the start page of its scaffolding.

When an aggregated breakdown is created, the moderator of the report taken as the basis stays as moderator. All reporters of the canceled breakdown report are assigned as reporters of the aggregated report.

Completion Conditions

The aggregation activity should be marked as completed as soon as the report activity is completed.

4.3.11 Activity: Select

Intent

The intent of the select activity is to select a breakdown for breakdown handling.

Activation Conditions

Only one breakdown handling process can go beyond the select activity (i.e., beyond the triggering phase) at a time. Therefore, this activity can be started as soon as the weight activity (Activity in Section 4.3.8) is finished and no other breakdown handling effort is in progress and has started or completed the select activity.

Instructions

Selecting a breakdown for handling should consider the following two objectives: 1) to reduce the chance that lengthy negotiation is needed to decide which breakdown to handle first, and 2) to reach a decision that team members accept and that motivates them to contribute. To achieve these objectives, four techniques are applied in sequence. If one fails, the next one is attempted. At any point, the team manager (Participant in Section 4.3.12) can preempt the process and dictate that a given breakdown is handled next.

First, a try is made to *prioritize* the breakdowns using an ordering suggested by the breakdown in-box. This ordering is based on the weights provided in the breakdown report (Artifact in Section 4.3.6) and a prioritization function configured in the tool. If there is only one breakdown with priority one (i.e., top priority), this breakdown is selected for handling. Otherwise, the next technique is attempted.

Second, the moderators (Participant in Section 4.3.4) of all top priority breakdown reports *negotiate* and try to reach consensus. If they agree on a breakdown to be handled next, the breakdown is selected for handling. Otherwise, the next technique is used.

The third technique requires that the moderators of all top priority breakdowns agree, and name another team member to *arbitrate* and propose a decision. If arbitration is not possible or yields no result, the fourth technique is used.

Fourth, if none of the participative techniques succeeded to select a breakdown for handling, a *FIFO* rule is used and the oldest breakdown is selected.

Some vendors of existing electronic moderation software (e.g., GroupSystems' EasyWinWin [17]) propose ranking or voting as a mechanism to choose from several alternatives. The Breakdown In-box tool differs from voting and ranking in that it attempts to reach a decision through consensus and making explicit the criteria used for the decision. Moreover, it is possible that the external member chosen as the referee for arbitration, suggests to reach a decision using voting and/or ranking tools (e.g., in cases where team members use arbitrary and subjective criteria and it is clear that all team members will maintain their opposed positions).

When a breakdown is selected for handling, the triggering phase for the selected breakdown ends. The moderator of the selected breakdown marks the activity as completed.

Completion Conditions

The activity is completed when a breakdown from the in-box is selected for breakdown handling.

4.3.12 Participant: Manager

Contribution

The manager is a team member with the authority to veto and prescribe team action regarding tailoring of teamwork. The method documents the situations that may require the participation of the manager. However, by definition, the manager can dictate alternative ways of action than the one suggested by the method.

Requirements

The manager is any team member with the required authority.

4.3.13 Tool: Breakdown In-box

Intent

The breakdown in-box serves as an organizer of breakdown reports and it provides support for selecting breakdowns for handling.

Instructions

Organizing Breakdowns

As an organizer of breakdown reports, the breakdown in-box classifies breakdowns reports according to three possible states: being reported, to be handled, or handled. Figure 4.26 presents the design of the graphical user interface of the tool. Breakdown reports are displayed in lists in one of the tabs according to status. The in-box tab shows only completed breakdown reports for breakdowns that have not gone past the triggering phase. The reporting tab shows all breakdowns still being written. The handling tab lists the breakdown being handled and all breakdowns that have already been handled. The figure shows the 'in-box' tab.

Lists show the names of the breakdowns and summary of the weights given by users as part of the report. Selecting a breakdown in any of the lists allows the user, through a menu action, to navigate to the corresponding breakdown report page or to the breakdown scaffolding in the scaffolding server.

Multiple Criteria Decision Making

The breakdown in-box is designed to work stand-alone. It can be started over the Internet, for example, as an Applet. Figure 4.27 provides an overview of the object model behind the tool.

The purpose of the in-box view is to help team members choose among newly reported breakdowns for handling by using weights. Weights are given independently by some team members (reporters) in relation to predefined criteria, and are aggregated by the tool. The aggregated weights for each of the criteria are used to try to identify the breakdown that is undoubtedly top priority.

Before deployment, the tool is configured with any number of criteria to use for weighting. Each criteria is associated with an ordered set of possible values. The ordering of the set indicates that X precedes Y, if a breakdown that weights X should be handled before the breakdown that weights Y (if only this criteria was considered). By default, the tool is configured with the criteria *Frequency* (with value set *all the time, daily, monthly, seldom, could recur*), *Severity* (with value set *blocks, introduces risk, delays, conditions*), and *Focus* (with value set *critical path activity, mandatory activity, optional activity*).

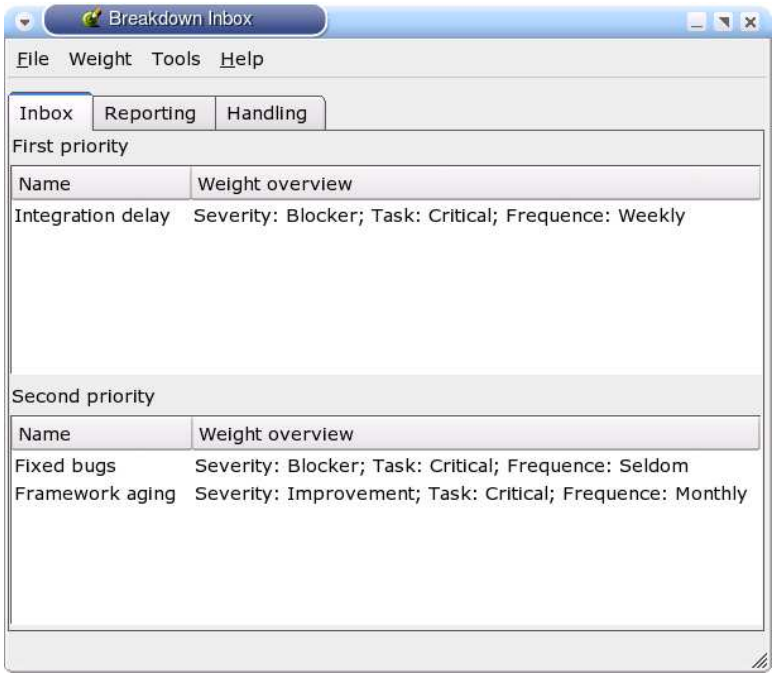

Figure 4.26: The Breakdown In-box: Breakdown Priorities

Three types of criteria can be included (different subclasses of class Criteria in the diagram), two of them with discrete values, namely enumeration criteria and integer criteria, and one with continous values, namely real criteria. For enumeration criteria, all possible values must be given in an ordered set. The criteria Frequency mentioned in the paragraph is an example of an enumeration criteria. An integer criteria can only take integer values in a range set between a minimum value and a maximum value. A real criteria can only take real number values also within a given range. Each criteria is configured with a strategy (see Strategy pattern in [26]) for aggregation. Minimum and maximum are strategies that can be used with all three types of criteria. Mode only applies to integer and enumeration criteria. Mean and Median only apply to real criteria.

The reporting tab provides a menu action to contribute a weight. A team member that is expected to contribute can select a breakdown from the list to define its weight. An input form shows the possible values for each criteria. The values the team member considers adequate can be entered and the form submitted. As a result, the in-box tool updates the breakdown information with the new contribution and updates the aggregated value. Only the most recent contribution of each team member is kept.

The tool holds a set of reported breakdowns with associated contributions from users regarding the importance of each breakdown. Users contribute their opinions regarding the weight of a breakdown for each of the configured criteria. The `updateResult` method in the class Breakdown creates or updates the aggregated result (an instance of Contribution), using the corresponding strategy to aggregate the values contributed for each criteria.

The method `getTopPriorityBreakdowns` of class BreakdownInboxTool applies a priority calculation algorithm to calculate the set of top priority breakdowns that are shown in the "First Priority" list of the user interface. If the list shows only one

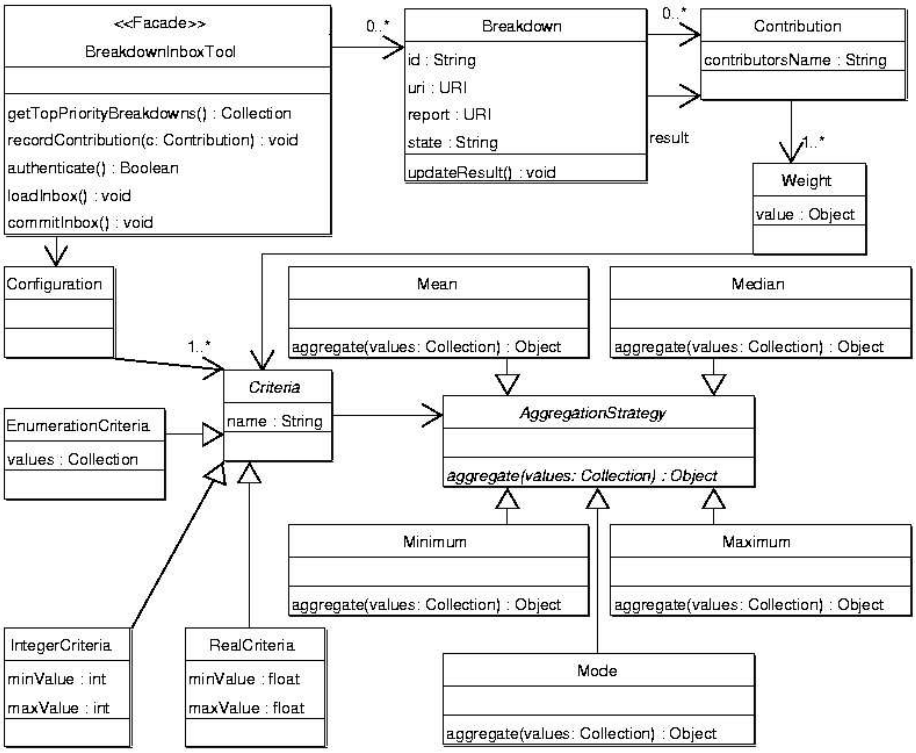

Figure 4.27: UML class diagram of the Breakdown In-box tool

breakdown, this is undoubtedly the top priority breakdown that must be handled next. If several breakdowns are top priority, the team must find an additional mechanism to decide among them.

The priority calculation algorithm tries to optimize the weights for all criteria. Let c_i, with i between 1 and some N, be the criteria used to weight breakdowns. Let $asImportantAs_i(b1, b2)$ be a function that returns *true* if the weight given for criteria c_i to breakdown $b1$ exceeds or is equal to the value given for the same criteria to breakdown $b2$. Let $moreImportantThan_i(b1, b2)$ be a function that returns *true* if the weight given for criteria c_i to breakdown $b1$ exceeds the value given for the same criteria to breakdown $b2$. A breakdown b_x is in the priority list, if there is no other breakdown b_y such that $asImportantAs_i(b_y, b_x)$ for all i between 1 and N, and $moreImportantThan_i(b_y, b_x)$ for some i. Figure 4.28 provides the pseudocode for the method `getTopPriorityBreakdowns` that encapsulates the algorithm[7].

The data model is stored in XML files that comply with the DTD in Figure 4.29. Files can be stored in a common file repository. Users download and lock the file, open it with the tool, contribute, and commit the file back to the repository (methods `loadInbox` and `commitInbox` of class BreakdownInboxTool).

An extension to the scaffolding server described in Section 4.2.2 enables the integration of the breakdown in-box. Through this extension, the in-box tool uses the scaffolding server as a data repository instead of XML files. If the tool is configured to work against the scaffolding server, methods `loadInbox` and `commitInbox` connect to the server instead of working on a file. Moreover, the method `authenticate` checks that a passport, indicating that the user has authenticated, is present.

To support this integration, the scaffolding server is extended to serve the URI *http://domain-name/breakdown-inbox* with POST and GET requests. Figure 4.30 shows these two requests as methods of the ScaffoldingServer class. The GET request translates to the method `getInboxAsXML`. The post request translates to the method `updateInboxFromXML`.

Method `getInboxAsXML` retrieves all available breakdown reports, all contributions made to the importance section of each breakdown report, and the state of each breakdown. The tool's configuration (i.e., available criteria) is obtained from the scaffolding's node that corresponds to the breakdown in-box tool. With this information the server assembles the XML string that is returned.

Breakdown reports are obtained by retrieving all instances of the BreakdownReport class. Each BreakdownReport instance has information about the recorded contributions (i.e., references to instances of class Contribution . To determine the state of the breakdown (REPORTING, SELECTING, HANDLING), the scaffolding server checks the state of the select activity (see Section 4.3.11). If the state of the select activity is Inactive, it means that the breakdown is still being reported, thus the state of the breakdown is REPORTING. If the state of the select activity is InProgress, it means that this breakdown is ready for selection (i.e., SELECTING). Finally, if the state of the activity is Completed, it means that the breakdown has been selected and is being handled or has been completed. Therefore, for the in-box tool, the state is HANDLING.

To store and later retrieve the configuration of the in-box tool in the scaffolding, the class InboxTool is created as a subclass of class Tool. InboxTool extends Tool with an attribute to hold the configuration. Therefore, the scaffolding page that represents the InboxTool has a specific section to hold the configuration data. When the page is shown, this data is hidden. When the page is edited, the configuration data is displayed in XML text and can be changed. When the page is saved, the method

[7]To simplify the pseudocode, the method dominates has been implemented in class BreakdownInboxTool. However, it is a Contribution's responsibility to know if it dominates another contribution, therefore it should be a method in class Contribution.

```
public Collection getTopPriorityBreakdowns() {
  Collection top = new Collection();
  for each Breakdown b1 in breakdowns do
  {
    dominated = false;
    for each Breakdown b2 in breakdowns do
    {
      if (dominates(b2, b1)
        dominated = true;
    }
    if (! dominated)
      top.add(b1);
  }
  return top;
}

private boolean dominates(Breakdown b1, b2) {
  betterInOne = false;
  asGoodInAll = true;
  howMany = b1.getResultCriteria().getLenght();
  for index in 1 to howMany do
  {
    c1 = b1.getResult().getCriteria(index);
    c2 = b2.getResult().getCriteria(index);
    betterInOne := betterInOne or (c1 > c2));
    asGoodInAll := asGoodInAll and (c1 >= c2)];
  }
  return betterInOne and asGoodInAll;
}
```

Figure 4.28: Pseudocode for the getTopPriorityBreakdowns method in the BreakdownInboxTool class.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT breakdown-inbox (criteria* breakdown* )>
<!ATTLIST breakdown-inbox
        id ID #REQUIRED
        description CDATA #IMPLIED
        uri CDATA #REQUIRED>
<!ELEMENT criteria EMPTY>
<!ATTLIST criteria
        object ID #REQUIRED
        name CDATA #REQUIRED
        aggregation (MEA | MED | MOD | MAX | MIN)
        type (INT | REAL | ENUM)
        enum-values CDATA #IMPLIED
        ordering (UP | DOWN)>
<!ELEMENT breakdown (contribution* result) >
<!ATTLIST breakdown
        id ID #REQUIRED
        state (REPORTING | SELECTING | HANDLING) #REQUIRED
        scaffolding-uri CDATA #REQUIRED
        report-uri CDATA #REQUIRED>
<!ELEMENT contribution (weight+)>
<!ATTLIST contribution
        contributor CDATA #REQUIRED>
<!ELEMENT result (weight+)>
<!ELEMENT weight EMPTY>
<!ATTLIST weight
        criteria REFID #REQUIRED
        value CDATA #IMPLIED>
```

Figure 4.29: Document Type Declaration used by the breakdown in-box tool

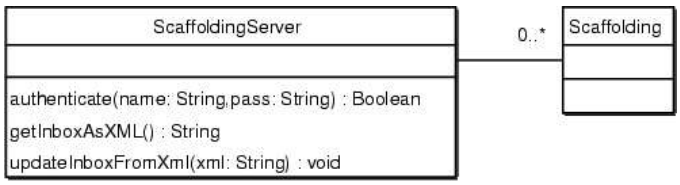

Figure 4.30: Extensions to the scaffolding server to integrate the breakdown in-box tool

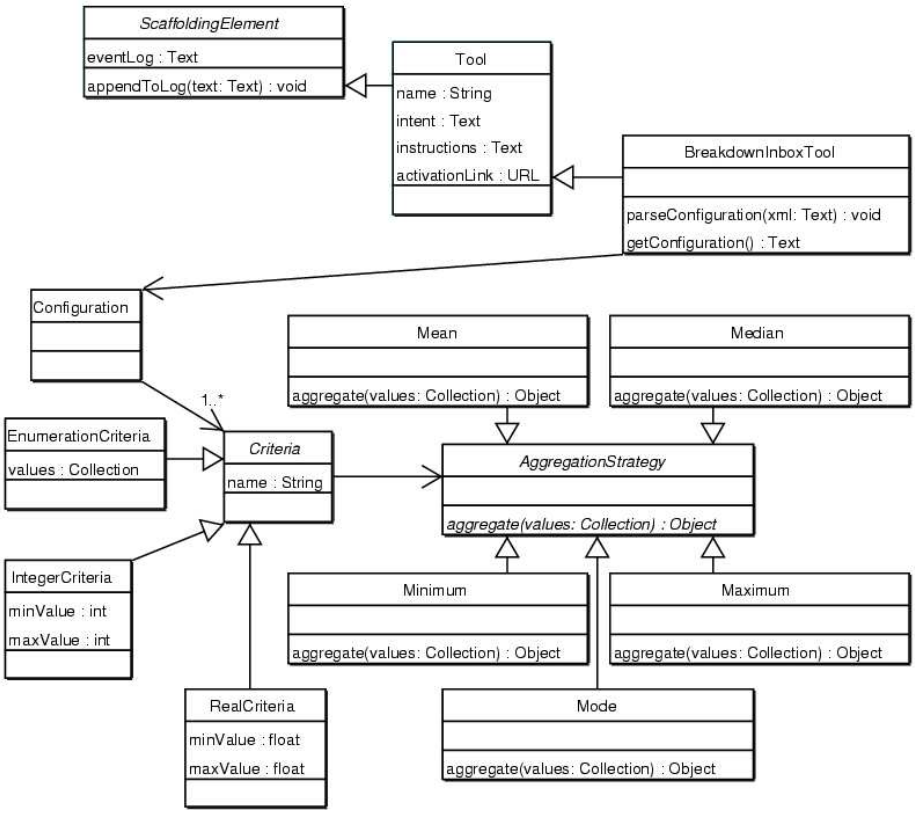

Figure 4.31: Class InboxTool stores the tool's configuration

`parseConfiguration` parses the configuration data and updates the corresponding objects. The method `getConfigurationAsXml` returns the tool's configuration as XML.

The method `updateInboxFromXML` takes as an argument the XML string submitted from the tool to the server and updates information stored in the scaffolding. The XML file is parsed. All new or changed contributions of weight are reflected in the corresponding breakdown report artifacts of the scaffolding's object model.

Tailoring Hooks

A common recurrent breakdown with the use of the breakdown in-box is the inability to decide based on the priority list. This happens if the priority list too often has several elements instead of one. The choice of criteria, values for the criteria, and mechanism used to aggregate values from independent participants impact the results of the priority calculation algorithm shown on page 83.

Each organization may need a different configuration, which could be obtained by careful experimentation before deployment as the result of handling level B breakdowns. For example, using too many criteria for weighting increases the possibility that several breakdowns end in the priority list. Whereas, using too few criteria reduces the contribution of the tool as an objective, multi-criteria decision tool (e.g., having only one criteria results in simple ranking).

The tool can be configured with the set of criteria to use, the values allowed for each criteria, and the ordering. The mechanism used to aggregate contributions can

be configured for each criteria to one of mean, median, mode, maximum, or minimum. The configuration is provided in XML files or, alternatively, as XML text in the node of the scaffolding that corresponds to the tool. To tailor the tool, the corresponding XML needs to be edited. This work is usually done by expert users or administrators. The extensions to the scaffolding that allow the integration of the in-box tool infer the state of breakdowns by looking at the state of an activity named "Select" in the scaffolding. To provide flexibility in naming activities, the extensions to the scaffolding server have a corresponding configuration file that indicates alternative activity names.

4.3.14 Summary

Requirement 2 in Section 2.3.2 argued for teamwork support for triggering breakdown, which is covered by the activities, tools, artifacts and participation described in this section. The breakdown report documents the breakdown as it is perceived by those who encounter it. An initial picture of severity is created by the collaborating reporters, which can be used, together with the breakdown in-box, to choose a breakdown for handling. Several mechanisms for selection are suggested to keep the effort as low as possible. During breakdown report, breakdowns can be aggregated in an attempt to concentrate forces and avoid tackling the same problem more than once. E-mail notifications about new breakdowns assure that the breakdown observed by some team members is known to everyone in the team. To react to level B breakdowns, both tools presented in this section provide tailoring hooks for their most important features. Activities, artifacts, use of tools, and participation in this phase have been stated independently of the work level. Therefore, support for triggering applies both to breakdowns in everyday work (as a level B capability) and to breakdowns that occur during tailoring (as a level C capability).

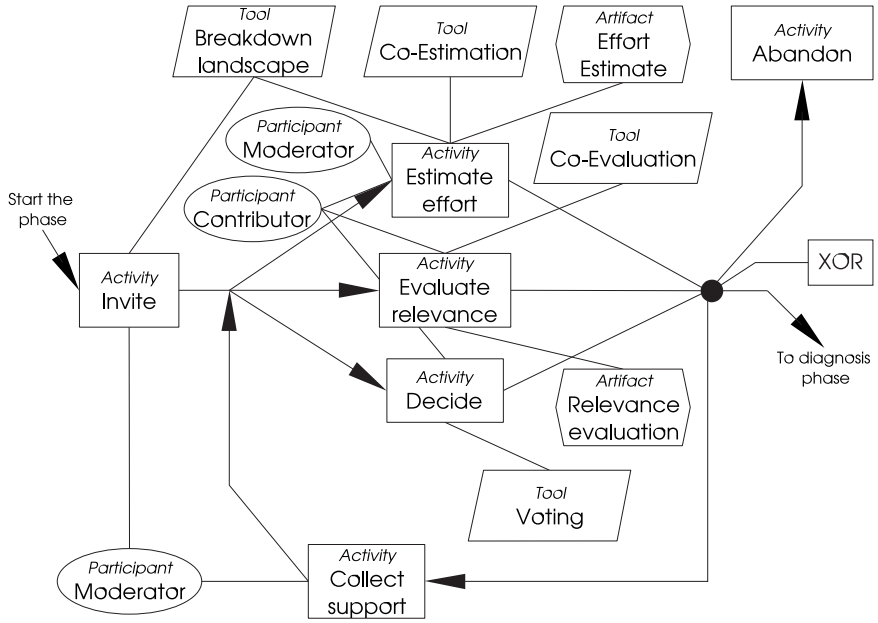

Figure 4.32: Definition phase of breakdown handling

4.4 Defining the Breakdown

4.4.1 Overview of the Definition Phase

The goal of the definition phase is to decide whether to handle the breakdown, ignore it, or postpone handling. This decision is taken on the grounds of a jointly constructed understanding of the relevance of the breakdown and of the effort required to handle it.

Figure 4.32 provides an overview of the definition phase. Breakdown definition involves six activities: inviting, estimating effort, evaluating relevance, deciding, abandoning, and collecting support. The box labeled "XOR" at the right of the figure indicates that, at that point, an exclusive choice between alternative continuation paths is made. In this example the decision is made between continuing with the next phase (diagnosis), abandoning, or collecting support. The moderator sets a deadline when each of the activities should come to an end. All team members can contribute in this phase.

4.4.2 Activity: Invite

Intent

The intent of this activity is to invite an initial set of team member to contribute to breakdown handling.

Activation Conditions

This activity can be started as soon as the select activity (Activity in Section 4.3.11) is completed and a breakdown is selected for handling.

Figure 4.33: Presentation of a participant node including accept/reject buttons for invitations

Instructions

Participation is key to successful breakdown handling. However, there is no tool or method that can assure fruitful participation. It is up to team members to contribute, and it is up to the organization's managers to motivate participation. This activity, and the tools that are used to perform it, aim at building an initial set of participants with representatives from all shared perspectives that have a clear stake in handling the breakdown.

The moderator (Participant in Section 4.3.4) can use the Breakdown Landscape (Tool in Section 4.3.7) to query for all shared perspectives that contributed a force that involves an object in the key elements section of the breakdown report (i.e., these perspectives have a stake in handling the breakdown). Team members that belong to any of these perspectives are invited to contribute.

To invite, a hyperlink reference is included in the contributor node (a node of type Participant) (see Section 4.4.3) pointing to the node that provides the details of each invited team member (a TeamMember node). Next time team members explore the scaffolding for changes, they will see they have been assigned for participation as contributors of a new scaffolding. Only one scaffolding node is used for the contributor, which references all team members that contribute. The contributor node provides the means to accept or reject the invitation.

Team members can reject an invitation if they are unable to perform the work (e.g., if they currently do not have time to make serious contributions). Figure 4.33 presents an example of a participant node (taken from the Reporter participant presented in Section 4.3.3). A section labeled "Invitations" in the node lists all invited participants as hyperlinks to their team member nodes. For the team member Guillaume Pontier, the current user, two buttons are available (accept and reject). He has not responded

to the invitation yet. The identity of the current user is retrieved from the passport. For the other users, the list indicates whether they have accepted, rejected or have not responded yet.

The moderator, team members and possibly the management need to negotiate to assure that at least one team member from each involved perspective accepts to contribute. If this requirement is not met, it is up to the management to decide whether to wait, to proceed and accept the responsibility of potentially uninformed or unfair decisions, or to directly abandon the case. To postpone the case is not an alternative. To postpone is an informed decision that requires participation from all perspectives.

Completion Conditions

This activity is completed when at least one team member from each of the perspectives with a stake in handling the breakdown has accepted the invitation. Alternatively, this activity can be completed if the management decides to proceed with the case or to abandon it.

4.4.3 Participant: Contributor

Contribution

The contributor is the central role of breakdown handling. The contributor helps identify the forces that define the breakdown, helps find solution alternatives, and helps evaluate the status of forces before and after a solution has been applied.

Team members participate as contributors, usually after they are invited to do so. However, they can request an invitation from the moderator if they believe they have something valuable to contribute. Team members that have been invited to contribute must stay aware of the progress of breakdown handling, and must contribute when needed. Contributors that are unable to fulfill their responsibilities can ask the moderator to be removed from the case.

A contributor node references (via hyperlinks) the pages of all team members that fulfill the role. To distinguish among team members that have been invited, team members that accepted participation, and team members that rejected the invitation, the class Participant in the hierarchy of scaffolding elements is extended as seen in the class diagram in Figure 4.34. The hyperlinks references to the team member pages are stored in three different collections namely, urlsOfInvitedMembers, urlsOfMembersWhoAccepted, and urlsOfMembersWhoRejected. Two methods, `accept` and `reject` in class Participant (provided in the user interface of the Participant node as a button associated to the selection of the list), allow team members to accept or reject the invitation.

Team Member Nodes

In Section 4.2.2 all nodes in the scaffolding hypermedia are presented. The TeamMember node (introduced on page 54) is used to record the data of a team member. A section labeled "References to this element" presents the comma-separated list of incoming links to the node. As later discussed in Section 4.2.2 (on page 58) TeamMember nodes are unique and shared among all scaffoldings. They are located in a special name-space called *shared*. To learn what work has been assigned to them, team members visit their team member nodes. There, they look for incoming links from Participant nodes and navigate to these nodes to learn about the role they play. With time, team members will become participants in several breakdowns. The list of incoming links to their participant node will grow (e.g., there will be several incoming references from Participant nodes with role "Contributor"). The section of

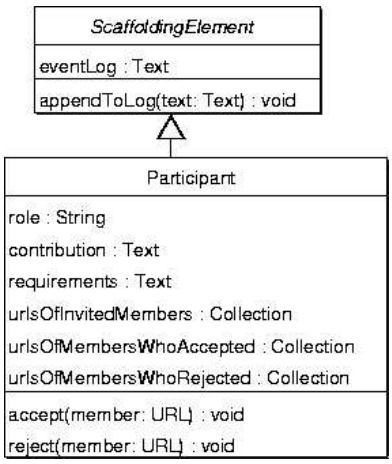

Figure 4.34: Extensions to Participant class to handle invitations

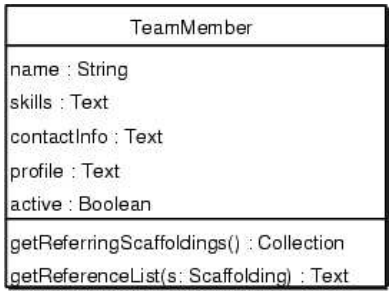

Figure 4.35: Extensions to the TeamMember class to simplify the use of incoming references

incoming references is split in several sections, one for each scaffolding (i.e., one for each breakdown). The method `getReferringScaffoldings` in class TeamMember (see Figure 4.35) retrieves the list of all scaffoldings with references to the team member node. The method `getReferenceList` builds the comma-separated list of incoming references for a given scaffolding.

Team members can leave the company. If they participated in breakdown handling, their team member nodes stay in the scaffolding server as part of the breakdown-handling history of the organization. It is important, before the participation of a team member is required, that only team members that still belong to the company are considered. The attribute `active` of the class TeamMember is true for team members that still belong to the organization.

Requirements

Only team members whose team member page has already been associated with a shared perspective can act as contributors.

4.4.4 Artifact: Effort Estimate

Intent

The Effort Estimate artifact collects the opinion of several perspectives regarding the time needed for handling the breakdown.

Instructions

For each activity in the process of breakdown handling, the effort estimate document presents an estimate of the time that team members will have to dedicate to breakdown handling if breakdown handling is pursued. These estimates are one of the arguments used to decide if the breakdown is handled, postponed, or ignored.

For each activity in the phases of diagnosis, design, treatment, and follow-up evaluation the document provides an optimistic estimate, a pessimistic, and a probable estimate. In the best case, the estimates are obtained from records of past breakdown handling. If no applicable records are available, the document is based on team members' best guesses.

The document presents a table with one activity per row and with its three values. One row for each phase presents the subtotal for the phase. A final row presents the totals.

The document is dynamically updated with the Co-Estimation Tool. The tool results are embedded as a plug-in that continuously checks for changes in the data (the details of this mechanism are presented in Section 4.4.7). Once the estimation is completed, the plug-in can be replaced by a plain text summary of the results. This artifact does not require a specially-formated page but can easily be created from an untyped page.

4.4.5 Activity: Estimate Effort

Intent

The intent of this activity is to produce, from multiple perspectives, an estimate of the effort of handling the breakdown.

Activation Conditions

This activity can be started as soon the activity invite (Activity in Section 4.4.2) is completed, or as soon as the activity collect support (Activity in Section 4.4.14) is completed.

Instructions

Effort estimation involves the creation of the effort estimate document (Artifact in Section 4.4.4). The document is prepared by the moderator and reviewed by all participants who decide to contribute. Effort estimates are indicated in work days.

The moderator (Participant in Section 4.3.4) first initializes the effort estimate document with data obtained from past experiences. If no past experience applies, the document is initially empty. The Breakdown Landscape (Tool in Section 4.3.7) is used to search for all past breakdowns that involve any of the objects in the key elements section of the breakdown report. The moderator decides how to interpret the result of the search. If several breakdowns are found, then only those that match best (e.g., cover all elements in the key set) with the one being estimated are considered. The moderator looks up the effort logs from all considered breakdown handling cases. The minimum value for each activity from all effort logs is taken as the minimun estimate. The maximum value is taken as the maximum estimate. Depending on the similarity of the considered reports, the moderator can decide to set the probable value to the average from all effort logs, the middle between minimum and the maximum, or to the most similar breakdown report. In any case, a note must be appended to the effort estimate that explains the origin of the data.

To complete and validate the estimation of required effort, the moderator, contributors (Participant in Section 4.4.3), and the manager (Participant in Section 4.3.12), can contribute by means of the Co-Estimation Tool (Tool in Section 4.4.6). Contributions from participants are aggregated together with the initial values.

Completion Conditions

Until the deadline set by the moderator, participants are allowed to contribute estimates.

4.4.6 Tool: Co-Estimation Tool

Intent

The intent of the Co-Estimation Tool is to create an effort estimate document for the breakdown.

Instructions

The Co-Estimation Tool is a simple deliberation tool. It uses the generic architecture for loosely coupled deliberation tools documented in the following section (Section 4.4.7). Team members enter and submit effort estimates in a spreadsheet-like user interface. The tool aggregates the contributions from all team members in a result estimate. At any point in time, users can see a preliminary result estimate.

First, the estimate form must be configured (usually by the moderator). The configuration indicates the activities to estimate (rows), the values to provide (columns), and the operation used to aggregate the individual estimates in each column (one of mean, median, mode, maximum, or minimum).

A default configuration is delivered with the tool in an XML file. The default configuration reflects the breakdown handling process documented in this thesis. If

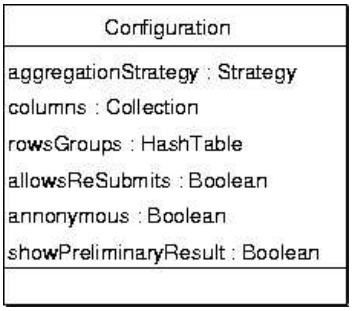

Figure 4.36: Configuration class for the Co-Estimation Tool

the process is changed, the configuration file needs to be updated accordingly. In the default configuration, all minimum values are aggregated by taking the smallest contribution. All maximum values are aggregated by taking the largest contribution. All probable values are aggregated by taking the most frequent contribution. Empty cells are ignored.

Figure 4.36 presents the Configuration class whose instances hold the tool's configuration data. The labels for the columns are stored in a collection. The labels for the rows are stored in a hash table that organizes rows in groups (e.g., activities grouped in phases). The operation used to aggregate values is configured applying the strategy design pattern as done for the breakdown in-box tool specified on page 79. Further configuration options are discussed in the following section where the tailoring hooks are discussed.

Figure 4.37 presents the classes that form the data model of the tool. The result, an instance of class Estimate, references one ValueEstimate instance for each configured cell (i.e., each row/column value). The method `getSubTotal` in class Estimate adds the total estimation for a given column, or for a given column and a given row group. A subclass of Estimate, the ContributedEstimate, represents a user's contributions. A ContributedEstimate indicates whether the user wants to be notified of changes. It has a rationale for the complete contribution and it identifies the contributing user. Moreover, in contrast to ValueEstimates, the ContributedEstimate references instances of ContributedValueEstimates, each of which can have an explanation of rationale.

Figure 4.38 outlines the user interface of the Co-Estimation Tool. The users enter estimates in the cells. The totals are automatically calculated. The users can explain the overall rationale behind the estimate in the text box at the bottom of the window. Additionally, a menu action over each cell allows the user to explain the rationale behind an individual estimate. Cells with rationale comments are indicated with a mark (e.g., a colored triangle in the upper left corner). A menu action "Submit" submits the contribution of the team member. If the check-box "Keep me informed" is marked upon submission, the tool will notify the user when the results change as a consequence of a new contribution.

Another menu action "Show results" loads the preliminary results (i.e., considering all contributions already received by the tool) in the table. Double clicking on a cell opens the details dialog that contains the list of all existing contributions for the cell, ordered by value. For each contribution, the dialog shows the value and the rationale of the contribution. The tool can be configured to show the identity of the contributor.

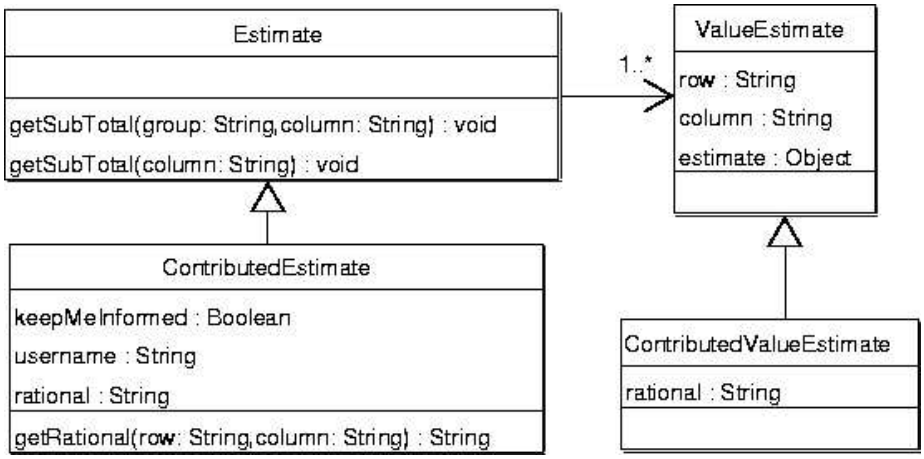

Figure 4.37: Classes representing contributions and results of estimates

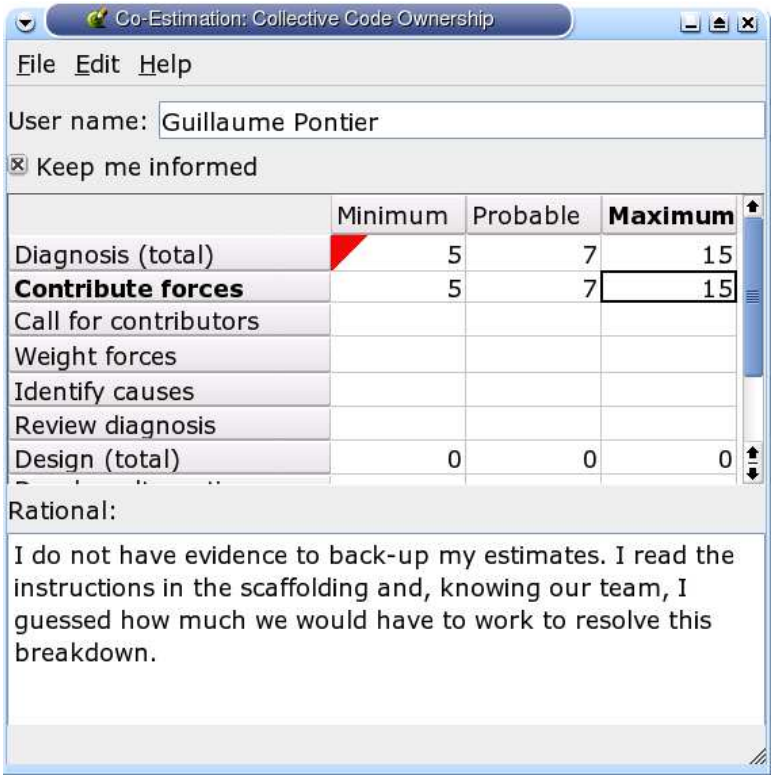

Figure 4.38: Simplified GUI design for Co-Estimation

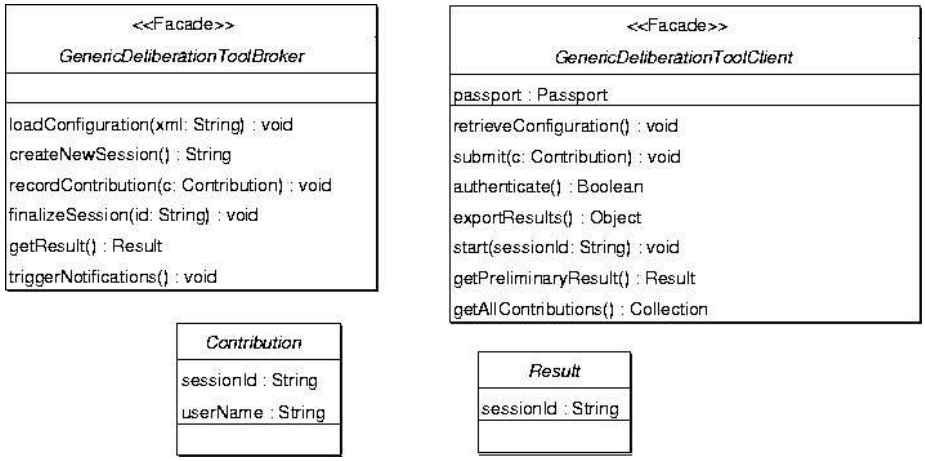

Figure 4.39: Class diagram including key classes of a loosely coupled deliberation tool

Tailoring Hooks

The tool can be configured to allow re-submits. If that is the case, a user can submit a new contribution that will replace any previously given one. The view-results dialog can be disabled for users other than the moderator. Rows, columns, and the aggregation function can be configured. The tool can be configured to keep contributions anonymous.

4.4.7 Generic Architecture for Loosely Coupled Deliberation Tools

Tools similar to the Co-estimation have already been used for work moderation (e.g., [15]) and collaborative learning (e.g., [73]). These tools provide stand-alone support for an atomic deliberation activity, such as voting, ranking, opinion poll, and meeting coordination (e.g., Meet-o-matic [35]). The operation of these tools can be characterized by the following five stages: setup, session initiation, contribution, finalization, and extraction of result. They consist of a central server, and a remote client. Figure 4.39 presents the key classes in a loosely coupled deliberation tools. Class GenericDeliberationToolBroker is a Facade that encapsulates the behavior available on the server. Similarly, class GenericDeliberationToolClient is a Facade that encapsulates the behavior on the client. Both classes are abstract. To instantiate the architecture, concrete subclasses are derived.

During setup, the tool is configured with all required parameters such as the number of allowed votes per user. The broker's method `loadConfiguration` reads the tool configuration from an XML file. The method `retrieveConfiguration` on the client retrieves the configuration from the server when clients connect.

The method `createNewSession` on the server, creates a new session and returns a session identifier. During session initiation, data structures are created and initialized to hold information that identifies the session and to store the users' contributions. When the client starts, a session id is passed to the `start` method to indicate the session that the tool will work in. Moreover, during start, the authenticate method is called to check that a passport (see page 55) is present and to retrieve the user's data from it.

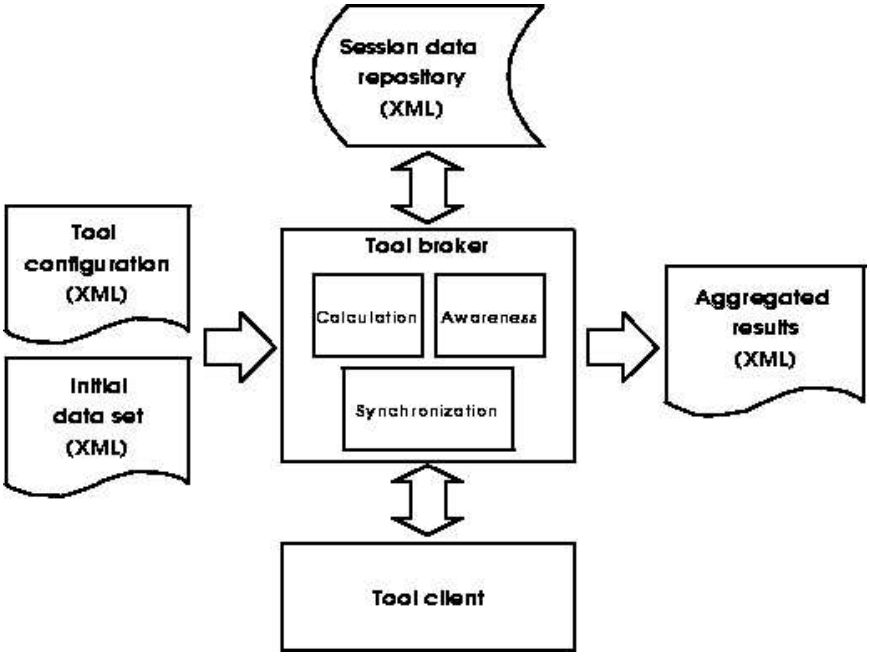

Figure 4.40: General client-server architecture for stand-alone, deliberation tools

During the contribution phase, users asynchronously submit their contributions. The method `submit` on the client assembles the contribution and sends it to the broker. Contributions are represented with concrete subclasses of abstract class Contribution. Each contribution holds the id of the session the tool is working with, and the user name of the contributing user. The `recordContribution` method on the broker stores the contribution. Contributions are stored in the data structures of the corresponding session.

During finalization, the method `finalize` calculates final results and the session is closed for contributions. The method `exportResults` in the client exports the final results to external format (e.g., a textual summary) for publication or as input for other tools. The result is represented with a concrete subclass of the abstract subclass Result. A session id identifies the session to which the tool's result belongs.

A synergy effect can be triggered showing users the preliminary results calculated from all contributions received so far. A tool's configuration option indicates whether preliminary results can be calculated and displayed or not. The method `getPreliminaryResult` in the client retrieves the result from the server and shows it to the user. Additionally, a tool can display the details of individual contributions. This option can also be configured. The client's method `getAllContributions` retrieves all contributions from the broker and shows them to the user.

Figure 4.40 presents a general client-server architecture for stand-alone, loosely coupled deliberation tools. Configuration data is passed on to the tool broker in XML format. Similarly, initial data, for example the elements to rank, is provided. The broker stores the configuration and initial data in an XML repository for later retrieval. As part of the configuration, the broker receives a session identifier. The identifier is used to match the data stored in the repository with the contributions from users.

Users activate the client component of the tool with the session identifier. The

client provides the session identifier to the broker to retrieve the session's data. When the user submits a contribution, it is sent to the broker labeled with the session identifier. The calculation subsystem in the broker updates the preliminary result with the new contribution. Then the broker stores the updated preliminary result and the individual contribution in the repository.

The operation of some tools requires that contributions do not overlap in time. That is, a user can only activate the tool if no other user is still working with it. This is the case of tools where the contribution of a user must take into account all previous contributions. The synchronization subsystem controls access. In its most simple form, the subsystem maintains a lock for the use of the tool.

These tools are conceived principally for asynchronous work. To help coordinate contributions and to achieve a synergy effect among users, an awareness subsystem can be included in the broker. The awareness subsystem takes care of notifying users of changes in the data (see method `triggerNotifications` in the broker's class), and of publishing information about tool usage (e.g., a user is currently contributing, most of the people have already contributed, etc.). The Co-Estimation Tool specified in the previous section does not provide synchronous usage awareness but only e-mail notifications on change.

Integrating Deliberation Tools in the Scaffolding

The setup of a deliberation is performed by the moderator before the activity that uses the tool is enabled. The tool node in the scaffolding that corresponds to the tool already includes an embedded call to the client component. The call uses the breakdown name as the session identifier.

The first time the tool is activated (normally by the moderator), it provides menu actions to pass configuration data and initial data to the broker. The moderator can enter the XML data directly or can provide an URI where the XML data can be found. The configuration data can usually be copied from the node that describes the tool. Some tools may replace the XML data with a simple to use configuration window or with a configuration wizard.

Embedding Deliberation Results in Scaffolding Artifacts

The `exportResults` method of a generic tool can be used to create a summary of the results in an external format such as plain text. This external format is needed if the results are to be incorporated into a scaffolding artifact (e.g., a report). This mechanism to document results of deliberation is simple. Moreover, it does not depend on the tool server being always available to provide the results.

There are activities that depend on partial results being always available as part of an artifact. The need to manually export results and update the corresponding artifacts represents a burden for the usability of the tool. This is the case for the estimate effort activity. The effort estimate artifact (see Section 4.4.4) must always reflect the preliminary result of the estimation activity. In cases like this, a special type of generic deliberation tool client can be used. The embedded deliberation result (see facade class GenericEmbeddedDeliverationResult in Figure 4.41) is a limited deliberation client that only implements functionality to retrieve the preliminary results from the server.

A GenericEmbeddedDeliverationResult must be implemented with a technology that allows seamless integration with web pages and network connectivity (e.g., Java applets). Moreover, embedding results this way implies that the tool server and the data that corresponds to the result's session must stay available. In most cases, the embedded result is replaced by a static (exported) result once the activity is completed and no further changes are expected.

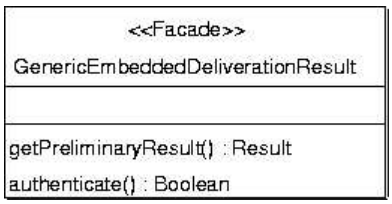

Figure 4.41: Generic class to embed deliberation results in scaffolding artifacts

4.4.8 Tool: Breakdown Landscape (continuation)

Section 4.3.7 specified the Breakdown Landscape tool matching the requirements of the triggering phase as well. This section extends the specification of the tool to cover the requirements of the definition phase. The Breakdown Landscape helps identify important stakeholders during the invite activity, and helps finding relevant past breakdowns during the effort estimation. Later sections present further features of the tool.

Section 2.2 defined the term shared perspective. The identification of shared perspectives is central to coordinate participation in breakdown handling as discussed in Section 2.3. Forces, as defined in Section 2.3.4, are a mechanism to structure communication during breakdown diagnosis and evaluation. They are used to express the reach and impact of the breakdown, and to explore and argue about solution alternatives. When a solution is implemented, forces document decisions that must be contemplated in future tailoring actions. The Breakdown Landscape tool enables the construction, maintenance, and use of a knowledge base (i.e., the Breakdown Landscape) that relates teamwork representations, perspectives and the forces documented during breakdown handling.

The Breakdown Landscape

The Breakdown Landscape holds information from all handled breakdowns and all breakdowns being handled. Each breakdown is identified with a case name identical to the name given to the instance of the scaffolding created for the breakdown (see Section 4.2.2). For each breakdown that is handled in the organization, a list of forces is provided as part of the diagnosis and another list of forces is given as part of the implemented solution. Each of the forces is associated with the perspective that first documented it.

During diagnosis, team members identify all the forces that they consider important for understanding the breakdown and motivating the exploration. These forces may have existed in the landscape before the breakdown occurred, or may have been first documented during diagnosis as systems of forces, as arguments, or as contextual facts. Moreover, as part of the diagnosis, team members evaluate the state of forces and update the landscape. During the design phase, team members document the forces that contribute to the operation of the solution. This list is not necessarily identical to the list of forces identified during the diagnosis phase.

Forces involve objects of teamwork. If an object is involved in a force, it can be influenced by changes in the state of the force (resolved/unresolved). For example, a tool may stop working effectively as a consequence of a related force being unresolved. Moreover, changes to teamwork that affect the object can influence the state of the force. When stating a force, team members explicitly indicate teamwork objects as being involved in the force.

Representations (introduced in Section 4.3.7 with the initial description of this

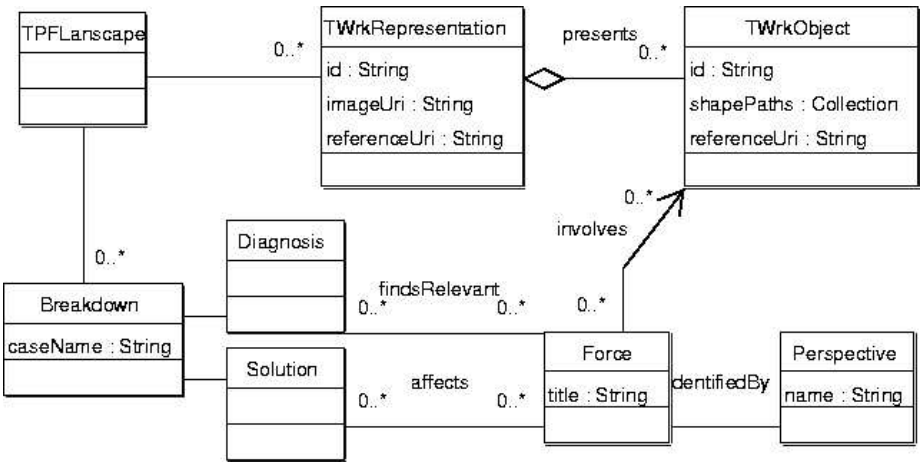

Figure 4.42: UML diagram for the object model of the Breakdown Landscape

tool) define an understanding about teamwork that is shared by team members (at least by those who share a perspective). Representations define a shared vocabulary that can be used to state forces. The elements in the vocabulary are the teamwork objects present in the representations. A teamwork object that is used to state a force (e.g., as the subject or as an object in the statement) is defined as being involved in the force. Force 5 in Section 2.1.5 of the breakdown Coding Conventions of the scenario explicitly references a characteristic of a tool (CVS), thus involving the tool in the breakdown. Section 4.5.2 provides further details of how forces are identified and documented.

A solution of a breakdown proposes changes to teamwork and documents how these changes affect the forces in the breakdown. Changes to teamwork affect teamwork objects. This creates a binding between the forces that the solution is trying to resolve (and which are not yet resolved) and the teamwork objects that are changed by a solution. The teamwork objects affected by a change indicated by the solution are, by definition, involved in all forces that change their state when the solution becomes effective. Section 4.7.5 provides further details of how the Breakdown Landscape is updated in response to changes.

The class diagram in Figure 4.42 models the resulting knowledge network connecting teamwork objects in the diagrams, breakdowns, forces, and perspectives.

A landscape can consist of several representations of teamwork, each of them describing teamwork from a different point of view (e.g., a workflow diagram or an organizational chart). Within a landscape, all representations and all teamwork objects included in these representations must have different identifiers (IDs). Moreover, an object that is included in more than one representation (e.g., a role that appears in both the organizational chart and the workflow diagram) must have the same identifier across all of them.

Exploring the Landscape

During the definition phase, the Breakdown Landscape is used to identify important stake holders and related breakdowns handled in the past. For this purpose, the landscaping tool provides the possibility of querying and exploring the landscape in graphical or textual form. Both modes can be configured with views to present only information that fulfills a condition provided by the user as a query. The graphical

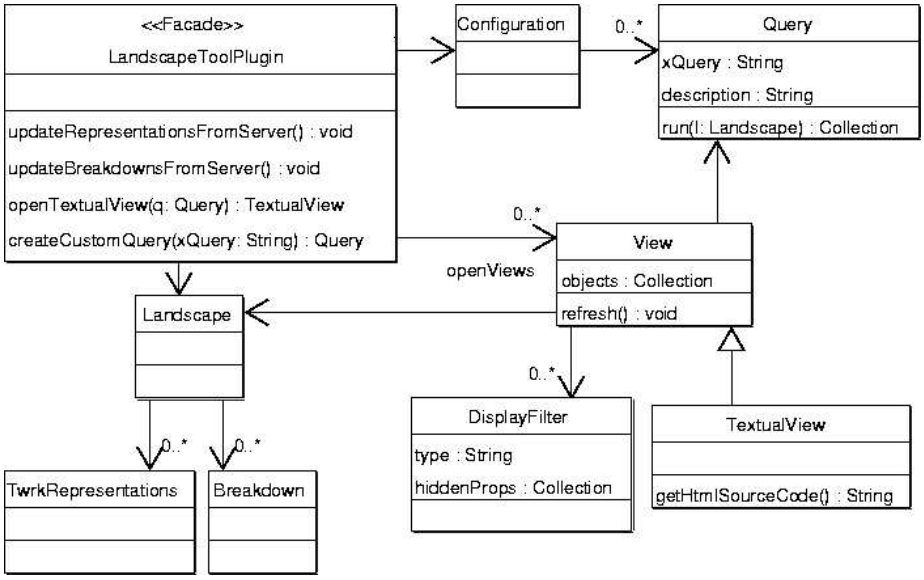

Figure 4.43: Class diagram for the landscape tool plug-in

mode serves the purpose of providing an overview of the arrangement of breakdowns, forces, and perspectives in relation to the objects of teamwork. Graphical views are useful during the diagnosis phase; therefore they are discussed later. The textual mode serves the purpose of providing detailed information, for example to be included in any of the documents that are created during breakdown handling. Textual views are sufficient for the queries used during the breakdown definition phase. The next section presents the queries that are used during definition and specifies the tool's functionality for textual views.

The Breakdown Landscape tool was partially specified in Section 4.3.7 as a pair of facade objects. The facade object LandscapeToolServer represented the server side component, and the facade object LandscapeToolPlugin represented the client side component. To support exploration and queries, the class LandscapeToolPlugin is extended with new methods as shown in Figure 4.43. For clarity, the methods discussed in Section 4.3.7 have been left out of the diagram. Only methods that deal with queries and views are specified.

The data of LandscapeToolPlugin is an instance of the class Landscape presented in Figure 4.42. Through the Landscape, the tool has access to all available representations and to the data available from all reported breakdowns. Representations are retrieved from the server through the method `updateRepresentationsFromServer`. Breakdowns and all the related objects from the model are retrieved from the server through the method `updateBreakdownsFromServer`.

Queries

The Breakdown Landscape tool provides a set of predefined query templates to produce the most commonly used views. The activities proposed in this thesis have been analyzed for needed queries. As a result, the following three queries have been included as predefined queries.

- Given a teamwork object, return all perspectives that contributed at least one force that involves the object. This query can be used to infer perspectives that

are good candidates to participate in the handling of a breakdown for which some key objects are known. This is valuable, for example, for the inviting activity of the definition phase.

- Given a perspective, return all teamwork objects that are involved in a force contributed by that perspective. This query can be used to determine the areas of teamwork known to a perspective. Such information can complement the description of the shared perspective.
- Given a breakdown, return all breakdowns that deal with objects (i.e., that have forces in the diagnosis or the solution that involve objects) involved in forces in the diagnosis of the given breakdown. This query provides a connection to, potentially, related past problems. It is useful as the basis for effort estimates for similar problems.

A query (an instance of class Query) is run on the landscape to yield a collection of teamwork objects. The body (xQuery) of the query is written in the W3C' s XQuery language [14], to be run against the landscape treated as an XML document defined by the DTD presented in Figure 4.55 on page 119. The result of the query must yield an XML document that is also compliant with the mentioned DTD. The resulting XML string is converted again into a collection of objects. The resulting collection of objects can be displayed on a textual or graphical view.

The above mentioned predefined queries are part of the tool's configuration. They are retrieved from the server on startup. Users (possibly with help from advance users) can create custom queries with the plug-in's method `createCustomQuery`. However, these queries are not part of the tool's configuration but are private to the user. To extend the set of predefined queries, the configuration files on the server must be modified by the administrators.

Textual Views

The method `openTextualView` in the plug-in opens a textual view (an instance of TextualView) for a given query (either a predefined query or a custom query). The result of the query is presented in the form of an expandable tree of elements. Objects are represented by expandable nodes, labeled with the type of the object. Expanding the node of a given object results in the display of the properties and relations of the object in the next level of the tree. Primitive values (i.e., numbers, and strings) are displayed as tree leaves that cannot be expanded. Collection attributes and references to other objects in the domain are represented as expandable nodes.

A view consists of a query and a set of display filters. A display filter indicates that for a given class of teamwork objects, the properties that must be hidden. The `refresh` method on the view causes the query to be run (`run` method of class Query). The result of the query, a collection of objects, is stored as an attribute of the view.

The textual view is constructed using standard HTML 2.0 constructs. This allows text to be selected and copied to other documents if the results of queries need to be used for the creation of other artifacts. The method `getHtmlSource` returns the HTML source code for the view. The source code is then rendered in an HTML viewer.

Tailoring Hooks (Continuation)

The HTML version of the graphical and textual views is constructed by processing XML data with XSL transformations. The transformations control the structure and style of the resulting HTML. New transformations can be provided to process the XML data in order to produce HTML-based views targeting specific requirements in content and look.

Queries and views are a central element of the landscaping tool. Users are able to define custom queries that are saved locally for each user. Users can share xQuery

expressions, for example via e-mail. However, to make queries available to all users, the configuration files on the server must be edited. This is usually done by administrators

4.4.9 Activity: Evaluate Relevance

Intent

The intent of this activity is to assess the impact of not handling the breakdown.

Activation Conditions

This activity can be started as soon as the activity invite (Activity in Section 4.4.2) is completed, or as soon as the activity collect support (Activity in Section 4.4.14) is completed.

Instructions

Contributors (Participant in Section 4.4.3) contribute consequences, probability estimates, and cost estimates by means of the Co-evaluating tool (Tool in Section 4.4.11) until the deadline set by the moderator. The tool aggregates the contributions and updates the Relevance Evaluation document (Artifact in Section 4.4.10).

Completion Conditions

This activity is marked as completed on the deadline set by the moderator.

4.4.10 Artifact: Relevance Evaluation

Intent

The intent of the relevance evaluation document is to provide an overview of the possible causes of not handling the breakdown that team members can identify.

Instructions

The relevance of a breakdown is defined as the potential damage it can cause to the organization if it is not handled. Team members use the relevance evaluation document as an argument to decide whether to handle the breakdown or not.

The relevance evaluation document compiles a list of potential consequences of leaving the breakdown unresolved. The document resembles a risk estimation document.

The information in the document is organized around the four variables of software development Scope/Time/Cost/Quality [12]. The Scope section lists all consequences that relate to a limitation to reach the desired scope of the product of work. An example of a scope consequence is "It won't be possible to distill a tool framework from this tool development project". The Time section lists all consequences that are related to delays in the work schedules. An example consequence in this section is "Framework development and tool development can no longer run in parallel, thus delivery delays can be expected". The Costs section lists all consequences that are better described as a monetary cost. An example consequence for this section is "Integration effort needs to be doubled, thus increasing costs." The section Quality lists all consequences that directly impact the quality of the result. An example for this section is "The probability of error in the final product is high."

The description of each consequence must include some indicator of the associated cost. Depending on the type of consequence, the cost can be expressed with the list

of elements that can not be delivered, with an estimate of the expected delays, with an estimate of the expected cost increment, or with some indication of the expected quality failures. For each of the consequences, the document also shows the number of users that believe it can occur. This number is used to sort the lists.

The document is dynamically updated with the Co-evaluation Tool. The tool results are embedded as a plug-in that continuously checks for changes in the data. Once the estimation is completed, the plug-in can be replaced by a plain text summary of the results. This artifact does not require a specially formated page but can be easily created from an untyped page.

4.4.11 Tool: Co-Evaluation Tool

Intent

The intent of the Co-evaluation tool is to help users build an agreed picture of the potential consequences of not handling the breakdown.

Instructions

The Co-Evaluation Tool is a simple deliberation tool based on the generic architecture specified in Section 4.4.7. Team members enter and submit what they consider to be potential consequences of leaving the breakdown unhandled, and/or vote for consequences contributed by other users. The tool compiles a list of all contributed consequences and votes. At any point in time, users can see the results.

Figure 4.44 outlines the user interface of the Co-Evaluation Tool. When the tool opens, the lists are filled with the consequences previously contributed by other users. Each tab pane shows one list of consequences for one of the fours variables Scope/-Time/Cost/Quality. The users can select an element in the list and cast or clear a vote. Items that the user votes are marked with a checkmark. The user can add a new consequence. New consequences are marked with a pencil icon.

The data model of the tool can be seen in Figure 4.45. A consequence (class Consequence) consists of a description, an indication of its potential cost in plain text, and the category the consequence belongs to (one of Scope/Time/Cost/Quality).

Users read through all the lists and issue votes on those consequences that they find probable. Additionally, they contribute new consequences. When their contribution is ready, they can submit with a menu action. On submit, all votes and new consequences are sent as a single contribution. If the checkbox "Keep me informed" is marked upon submission, the tool will notify the user when the results change as a consequence of a new contribution. To model contributions, a class CoEvaluationContribution is derived from the generic Contribution class presented in Section 4.4.7.

The result (a RelevanceEvaluation result derived from the generic Result class presented in Section 4.4.7) is directly built from the set of all contributed consequences and votes. The method `getConsequences` returns the collection of all contributed consequences. The method `getVoteCount` in class Consequence returns the count of all issued votes for the consequence. The action "Show results" opens a window similar to that in Figure 4.44 to show the result. The lists now have an extra column that shows the number of votes received by each consequence. The lists are sorted by descending number of votes.

Tailoring Hooks

The view results dialog can be disabled for users other than the moderator. The tool can be configured to keep contributions anonymous.

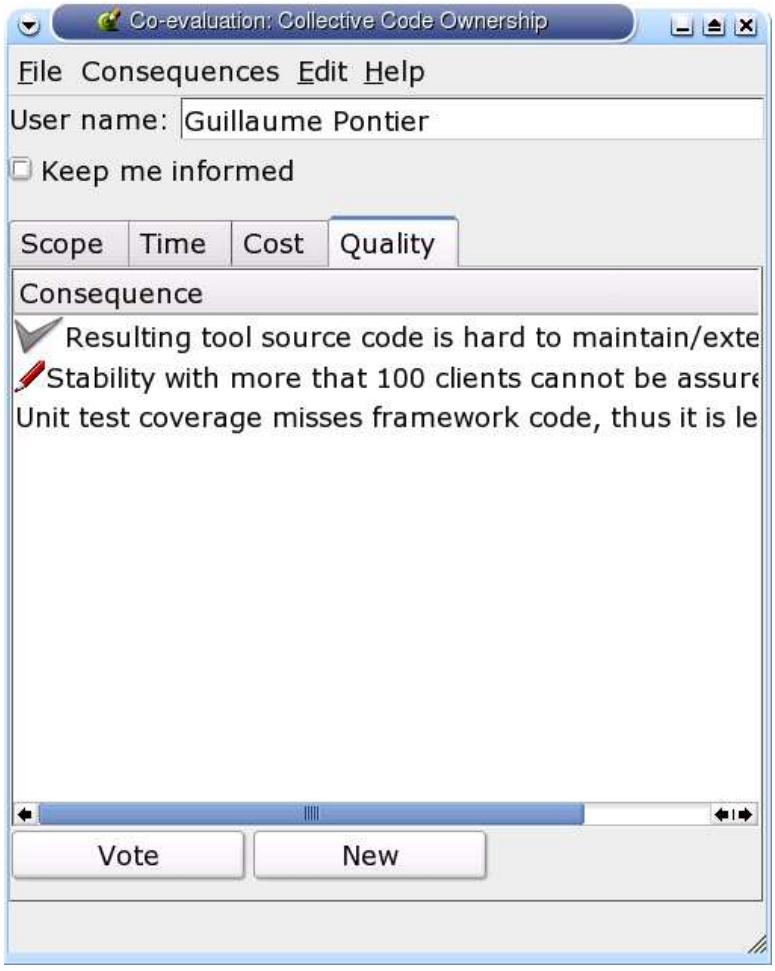

Figure 4.44: Simplified GUI design for Co-Evaluation

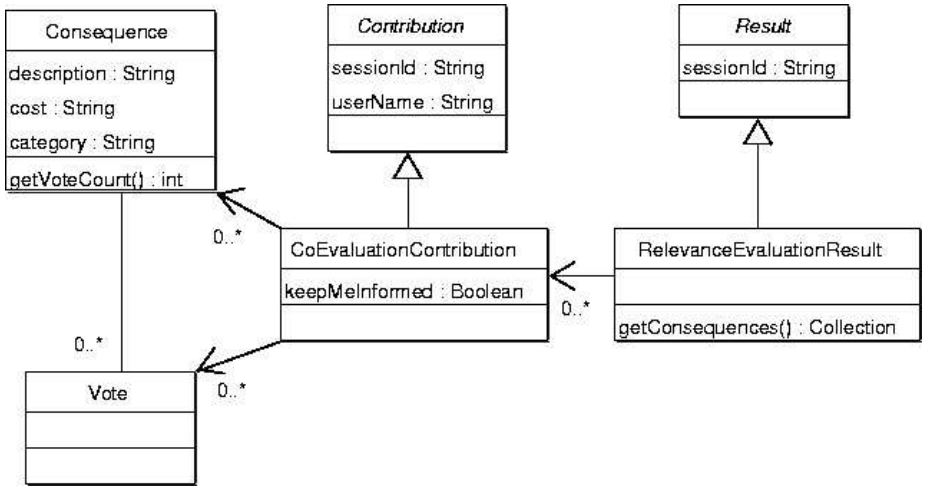

Figure 4.45: Classes representing contributions and results of relevance evaluations

4.4.12 Activity: Decide

Intent

The intent of this activity is to decide wether to handle the breakdown, postpone handling, or ignore the breakdown.

Activation Conditions

This activity can be started as soon the activity invite (Activity in Section 4.4.2) is completed, or as soon as the activity collect support (Activity in Section 4.4.14) is completed.

Instructions

Deciding on breakdown handling is done through an opinion poll. A Voting Tool (Tool in Section 4.4.13) provides the required functionality. Until the deadline set by the moderator, contributors (Participant in Section 4.4.3) can select one of three options: continue handling, postpone handling, or ignore the breakdown. The tool displays the status of the poll in a pie chart (e.g., the distribution of votes). Opinions can be changed at any time before the deadline.

On the deadline, the moderator (Participant in Section 4.3.4) freezes the effort estimation and the relevance evaluation. No further contributions can be made. The choice agreed to by a simple majority[8] (i.e., the choice with more than 50% of the votes) is taken as the group decision. If no choice is agreed to by a simple majority, breakdown handling is postponed.

As previously mentioned, team members are expected to base their choice on the results documented in the effort assessment (Artifact in Section 4.4.4) and the relevant evaluation (Artifact in Section 4.4.10). However, these documents are continuously evolving until the deadline. The goal of this approach is to reduce to the minimum

[8]Simple majority is the most common requirement in voting for a measure to pass, especially in deliberative bodies and small organizations.

the effort of gathering arguments to reach consensus about decisions. In some way, the decision method is a combination of polling and auction that has the additional effect of motivating participation. A participant can use the partial result of the poll as an indicator of how others think about the issue. The effort estimate and the relevance evaluation are the tools for the participant to try to drive the group's decision. In this way, a negative partial result (from the point of view of some team member) would motivate contributions to the effort and relevance documents. A similar motivating effect is found in governmental elections where candidates try to approach non-favorable results of opinion polls with new proposals (and eventually attacks on other candidates). However, in contrast to governmental elections, for the case of breakdown handling, maintaining campaign effort to a minimum is desired.

A consequence of conducting effort estimation, relevance evaluation, and decision making in parallel is that participants need to be kept informed about the latest developments. For this reason, each of the tools used can be set to provide notifications of changes.

Completion Conditions

This activity ends on the deadline set by the moderator.

4.4.13 Tool: Voting Tool

Intent

The intent of the voting tool is to capture the support of team members for one of several choices. The available choices are configured in advance.

Instructions

A voting tool can be implemented as a simple deliberation tool based on the generic architecture specified in Section 4.4.7. Moreover, implementations of stand-alone voting tools fulfilling the requirements of the decide activity can already be found on the Internet (e.g., polls in Yahoo groups [3]).

4.4.14 Activity: Collect Support

Intent

The intent of this activity is to collect support of team members to resume handling of a postponed breakdown.

Activation Conditions

This activity can be started when the activity decide (Activity in Section 4.4.12) is completed and the result of the decision is to postpone breakdown handling.

Instructions

If the decision is to postpone handling (either because the group explicitly decided it or because it was not possible to reach consensus), the case is put on hold. The moderator (Participant in Section 4.3.4) is responsible for collecting signatures in favor of resuming breakdown handling exceeding 50% of the number of total votes emitted during the decision activity. As a consequence of team members leaving the organization, the number of team members may be less than found at the time voting took place. This could complicate being able to collect support. To cope with this

issue, the 50% of signatures is calculated over the number of votes originally emitted minus the number of team members that left the organization since the voting.

Completion Conditions

This activity is completed when the required number of supporting team members is reached.

4.4.15 Activity: Abandon

Intent

The intent of this activity is to clean up and store the case of a breakdown that will not be handled.

Activation Conditions

If the result of the decide activity (Activity in Section 4.4.12) is to ignore the breakdown, the case is closed, and the process of breakdown handling is abandoned.

Instructions

In the event that breakdown handling is abandoned, the moderator (Participant in Section 4.3.4) closes all documents in the case for writing. An abandoned case documents a breakdown that the group does not consider worth pursuing (neither now, nor in the near future). In contrast, a postponed case deals with a breakdown for which the group indicated that the conditions for handling do not yet exist, but could exist in the near future. Nothing stops a team member from reporting an abandoned case again.

Completion Conditions

This activity is completed when the moderator has closed all documents in the case for writing.

4.4.16 Summary

Requirement 3 in Section 2.3.4 argued for teamwork support to define the breakdown and decide on handling it. The Effort Estimate document and the Relevance Evaluation document define the impact and cost of the breakdown. They are created with tools that allow all team members to contribute. These documents are the basis for a decision. The mechanisms used to develop the documents and to make the decision aim at minimizing the effort required for this phase. Only team members that seem to have important contributions to make are invited. At the end of this phase, the team has decided, on the basis of data contributed by the team, how to proceed.

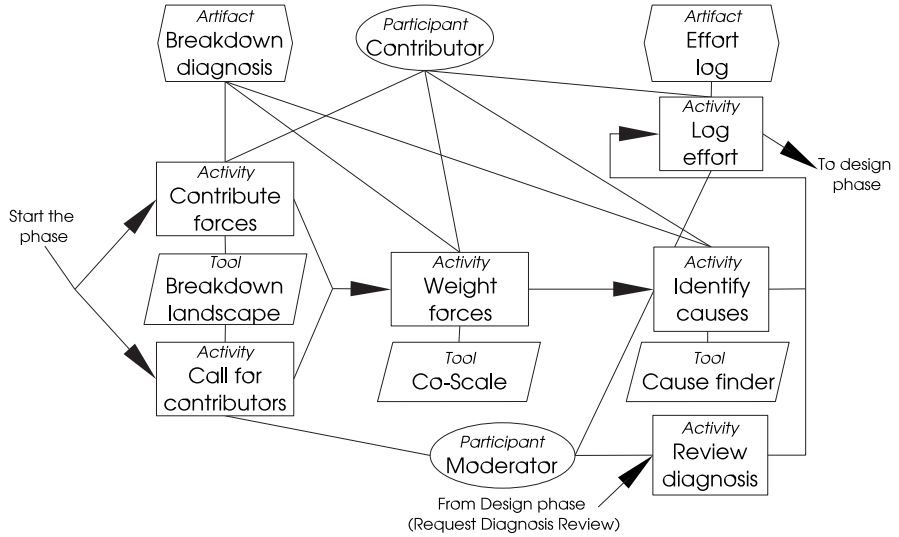

Figure 4.46: Diagnosis phase of breakdown handling

4.5 Conducting the Diagnosis of the Breakdown

4.5.1 Overview of the Diagnosis Phase

The goal of the diagnosis phase is to document the forces that contribute to the occurrence and resolution of the breakdown. The main outcome of this phase is the breakdown diagnosis, a document with the forces that define the breakdown and their associated weights. The breakdown diagnosis is the team's agreed understanding of the problem. During the design phase, several groups of team members will develop solution alternatives. The breakdown diagnosis is the basis for their work. Therefore, after the diagnosis phase is completed, no further changes should be made to the breakdown diagnosis. Experience showed (see Section 6) that during the design phase, team members improve their understanding of the problem. It is also known that attempts to solve wicked problems may result in a re-framing of the problem itself (Schön [63] refers to this phenomenon as reflection in action). This improved understanding results in new forces that any solution should consider, and therefore should be incorporated in the breakdown diagnosis so all teams designing solutions take them into account. If the breakdown diagnosis needs to be changed after the diagnosis phase has been completed, all activities in the phase need to be reconsidered.

Figure 4.46 provides an overview of the diagnosis phase. This phase consists of five activities, namely, contribution of forces, calling for contributors, weighting of forces, testing of forces, reviewing diagnosis, and logging effort. In addition to the breakdown diagnosis, the diagnosis phase results in an updated Breakdown Landscape and an effort log for the phase.

4.5.2 Activity: Contribute Forces

Intent

The intent of this activity is to discover and document the forces that make up the breakdown.

Activation Conditions

This activity can be started when the activity decide (Activity in Section 4.4.12) of the definition phase is completed, and the decision is to proceed with breakdown handling.

Instructions

Identifying and documenting the forces that make-up the breakdown is the central component of collaborative breakdown handling. In this activity, contributors (Participant in Section 4.4.3) contribute to a shared picture of the problem. This picture is a part of the breakdown diagnosis (Artifact in Section 4.5.4).

By means of the Breakdown Landscape tool (Tool in Section 4.3.7) participants can explore the existing forces and select those that are relevant for this case and add them to the diagnosis. If team members recognize the existence of new, undocumented forces, these can be added. The contributions of team members are not bound to their identity but to their perspectives. That is, team members, in the name of perspectives, contribute with forces that refer to teamwork objects found in any of the representations available in the landscape.

If the teamwork objects that are relevant for a force are not present in any of the available representations, a new representation can be created and added to the landscape. As importing is part of the tool's server-side functionality (see Section 4.3.7 for details about creating and importing representations) team members may require assistance from the system administrators to import new representations.

The Breakdown Landscape tool handles asynchronous work. Users can sign up to be notified when contributions are made. Notifications contain a summary of the changes. Being aware of changes is important for the moderator who needs to keep the list of participants up to date as part of the calling for contributors activity. Moreover, as contributions occur asynchronously, awareness about updates is important to produce a synergy whereby team members inspire one another.

The set of available teamwork representations is dynamic because new representations can be added on demand. Each representation can be seen as one way to describe teamwork (or a part of teamwork). Changes to the available representations need to be notified to users. Representations can be added without consensus. As a consequence, there may be situations in which users disagree about the correctness of a representation (e.g., the representation does not match how work is really done). If disagreement cannot be resolved, it becomes a problem in the use of the representations as a tool to collaborate for breakdown handling. It becomes a breakdown during breakdown handling (i.e., a level B breakdown). Breakdown handling of the level A breakdown is interrupted until the level B breakdown is resolved, resulting in a tailored set of agreed representations.

Past experiences can help understand the breakdown. Team members can use the Breakdown Landscape tool to perform queries that retrieve information from related past breakdowns. Graphical views are specially useful to get an overview of all breakdowns, perspectives, or forces that relate to some part of teamwork (i.e., to elements in a representation). In this way, the Breakdown Landscape can be used to point team members to relevant past breakdowns (either solved, postponed, or failed breakdowns). Further information about these breakdowns can be obtained from the scaffoldings that correspond to each case.

A default limit for the duration of this task is given when the scaffolding for breakdown handling is deployed. This limit depends on the organization, for example, on the time team members can spend in tailoring activities. The duration of this activity is configured in the master scaffolding and propagated (copied) to all instances. It can be changed on-demand by the moderator of a given case, in a scaffolding instance that corresponds to that case.

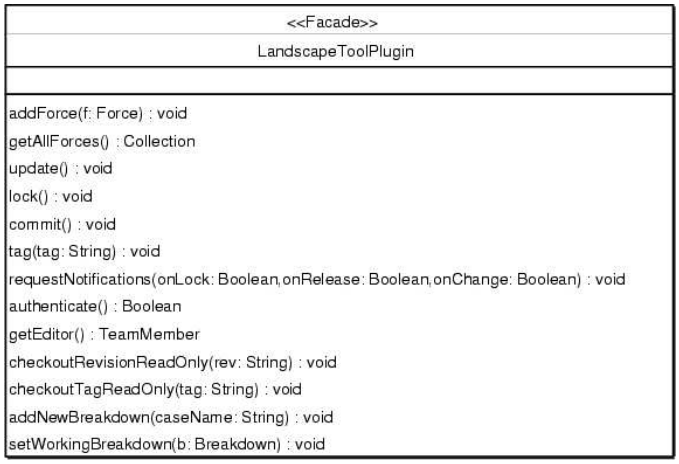

Figure 4.47: Breakdown landscape's client-side object model

To complete the activity, the moderator starts the Breakdown Landscape and performs a query for all forces in the diagnosis of the breakdown. The result, a textual view, is copied to the breakdown diagnosis artifact of the scaffolding. After this activity has been completed, no further forces should be added.

Completion Conditions

This activity is completed when the time limit is reached. However, the moderator can mark the activity as completed before the deadline with the agreement of all team members.

4.5.3 Tool: Breakdown Landscape (Continuation)

Section 4.3.7 specified the functionality of the Breakdown Landscape tool required for the triggering phase. Section 4.4.8 specified the functionality required for the definition phase.This section extends the tool specification with the functionality required for the diagnosis phase.

Documenting Forces

When the Breakdown Landscape is first deployed, it only contains the initial teamwork representations and shared perspectives. During the diagnosis of a breakdown, forces are documented (i.e., added to the landscape). Figure 4.48 depicts the main features of the tool's window that allows the definition of new forces. Figure 4.47 shows how the facade class LandscapeToolPlugin is extended to provide this functionality.

The textual description ("statement") entered in the upper part of the windows shown in Figure 4.48 is the central part of a force (this is the only property documented for the forces found in the scenario). The statement must be concise. It is not necessary to explain or argument for what is stated.

The user can indicate whether the force is a fact, an argument, or a system of forces (selecting the corresponding GUI radio button). In the case of contributing a system of forces, the user is able to add constituent forces by selecting them from the ones that have already been defined. The method `getAllForces` of the plug-in returns all available forces. These are shown in a separate window (not shown in the figure)

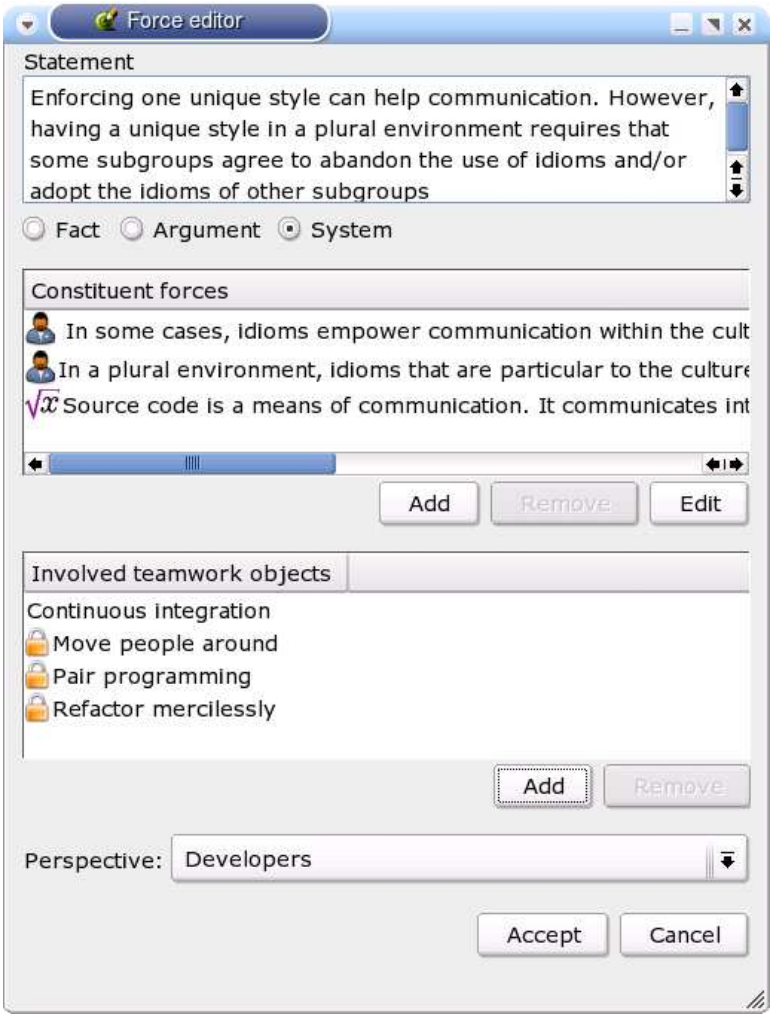

Figure 4.48: GUI design for the "Force editor"

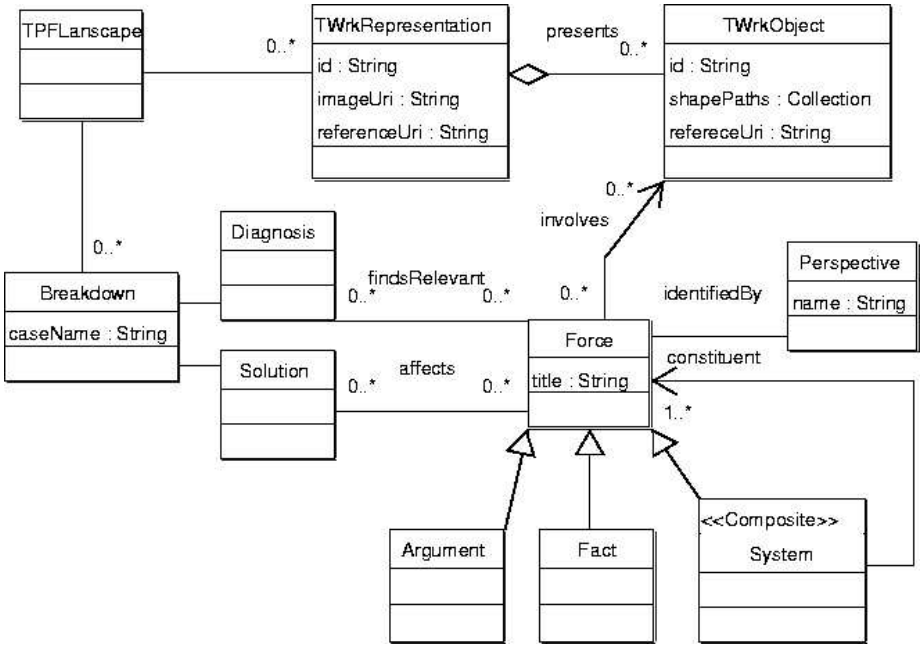

Figure 4.49: UML diagram for the object model of the Breakdown Landscape

and can be selected. The constituent forces of a system of forces can be seen in the list-box labeled "Constituent forces" in the figure. Constituent forces are additionally marked according to their type. An icon with a person on it indicates that the force is an argument. An icon with a formula on it (square root of x) indicates that the force is a fact. An icon with gears on it (not shown in the figure) indicates that the force is a system of forces. These icons are used in all windows that present a list of forces.

Figure 4.49 extends the object model of the Breakdown Landscape presented in Figure 4.42 on page 99 to model the different types of forces. The pattern Composite [26] (documented as the UML stereotype Composite) is used to model systems of forces.

Forces can only be added to the landscape if the user indicates at least one teamwork object that is involved in the force. The user can select from a collection of all defined teamwork objects and add them to the list for the force. The functionality that allows browsing representations and selecting teamwork objects has been described in Section 4.3.7. Involved objects are displayed in the list labeled "Involved teamwork objects" in the figure. A system of forces involves all objects that are involved in any of the constituent forces. It is not possible to remove them from the list of involved objects. They are marked by a lock icon. However, it is possible to indicate additional ones.

Finally, the user can sign the force by choosing the perspective that best represents the contribution from the combo box widget labeled "Perspective" in the figure. By accepting the changes ("Accept" button) the new force is added to the landscape (method `addForce` of the plug-in class).

Forces are added to the Breakdown Landscape as a result of the diagnosis or solution of a breakdown. A menu action "Start new breakdown", corresponding to the `addNewBreadown` method of the tool's facade class, adds a breakdown object to the landscape. New breakdowns are usually created by the moderator as part

of the initialization of tools. To add new forces, team members must first set the working breakdown via the "Set working breakdown" menu that corresponds to the `setWorkingBreakdown` method of the facade class.

It is possible, with a window similar to the one in Figure 4.48, to edit and delete forces. Editing and deleting forces is only allowed for team members that belong to the perspective that contributed the force.

Provisions for On-line Collaboration

The Breakdown Landscape is conceived for asynchronous, document-based collaboration. Users retrieve the latest version of the landscape from a common document repository (via the `update` method of the plug-in class) and work locally on their personal computers. After making changes to the landscape, the user needs to upload the changed landscape back to the common repository (via the `commit` method of the plug-in class).

While a user A makes changes to the current version of the landscape, other users can retrieve the current version of the landscape from the repository, change it, and upload it to the repository. As a consequence, user A continues working (unaware of the actions of the others) on an out-of-date version. If user A is allowed to upload the changed out-of-date version of the landscape to the repository, the changes introduced by the other users would get lost. In order to avoid this type of conflict, users that have the intention to introduce changes to the landscape must first acquire a lock on it (via the `lock` method of the plug-in class). Only one editor lock is granted at a time.

Section 4.3.7 described how the method `authenticate` authenticates and makes the information contained in the user's passport available for the plug-in. This information, in particular the user name, is used to update, to commit, and to acquire locks.

The Breakdown Landscape tool allows team members to request notifications whenever a lock is granted for the landscape, and whenever a lock on the landscape is released. Moreover, the identity of the user who has the lock on the landscape is known to the tool. Team members can query this information (via the `getEditor` method) and can get in contact with the editor in case they consider that synchronous collaboration is necessary. Moreover, the tool can send out notifications every time the data file is committed. The notification contains a summary of the changes. The method `requestNotifications` is used to change the user's preference regarding notifications.

The Breakdown Landscape tool does not provide support for synchronous collaboration. In case that several users have the intention to synchronously modify the landscape, they can share the landscaping tool through any of the currently available application sharing systems (e.g., MS Netmeeting). If users wish to deliberate about the landscape, they can use the results of queries as the focus of on-line, anchored conversations [20].

Queries (Continuation)

Landscape queries that are useful during breakdown definition are presented in Section 4.4.8. The following queries have been found useful for breakdown diagnosis.

- Given a breakdown, return all the forces that are, at most, at a distance d from objects in a given set (e.g., the set of key elements in the breakdown report, or the set of teamwork objects changed by the solution). Forces that involve objects in the set are at distance zero. If a force F is at distance d, other forces that involve an object that is also involved in F are, at most, at distance $d+1$. This query is useful if team members prefer to first focus diagnosis on the key objects indicated in the breakdown report, and then incrementally enlarge the scope of diagnosis. As a heuristic, the scope can be enlarged until the resulting

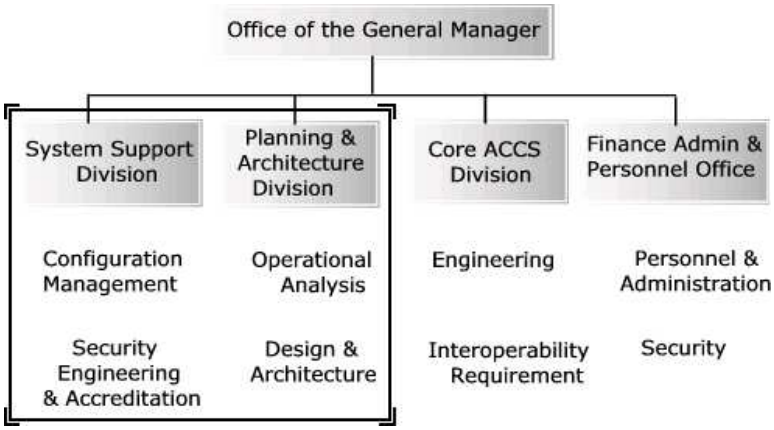

Figure 4.50: Example of a rectangular selection in a representation based on an organizational chart

set of forces for a distance *d* equals that of distance *d+1*. This query is also useful to limit the effort of testing the impact of solutions while these are still under development.

- Given a two-dimensional area of a representation (eventually the complete representation), return all teamwork objects in the area, plus all forces that involve any of these objects, plus the perspectives that documented these forces. This query can be used to explore distributions of forces on a particular area of teamwork. For example, it may be of interest to find all forces and contributing perspectives regarding a given part of the organization in a teamwork representation based on an organizational chart. Figure 4.50 depicts this example. The rectangle with double-line corners represents the selected area of interest.
- Given a two-dimensional area on a representation (eventually the complete representation), return all breakdowns with forces involving teamwork objects in that area. This query provides an overview of the distributions and frequency of problems (breakdowns) in relation to aspects of teamwork. Moreover, it provides pointers to past breakdowns which could provide useful insight about a problem.

Graphical Views

Textual views (see Section 4.4.8) are used to display the result of queries as an HTML-based expandable tree. Textual views can be easily pasted into scaffolding artifacts. Graphical views aim at displaying the results of complex queries that involve the graphical component of representations (i.e., the diagrams).

Section 4.3.7 presents the details of the use of teamwork representations to construct the Breakdown Landscape. A teamwork representation is constructed on top of a diagram of teamwork (e.g., the organizational chart in Figure 4.50). Teamwork elements in the representation occupy a two-dimensional area in the diagram (e.g., each organizational unit in Figure 4.50 occupies a rectangular area).

A graphical view of the result of a query is created by showing, on top of the diagrams of the available teamwork representations, the areas occupied by each of the teamwork objects involved in the result. Figure 4.51 shows a graphical view for the result of a query. In this case, each one of the three representations available

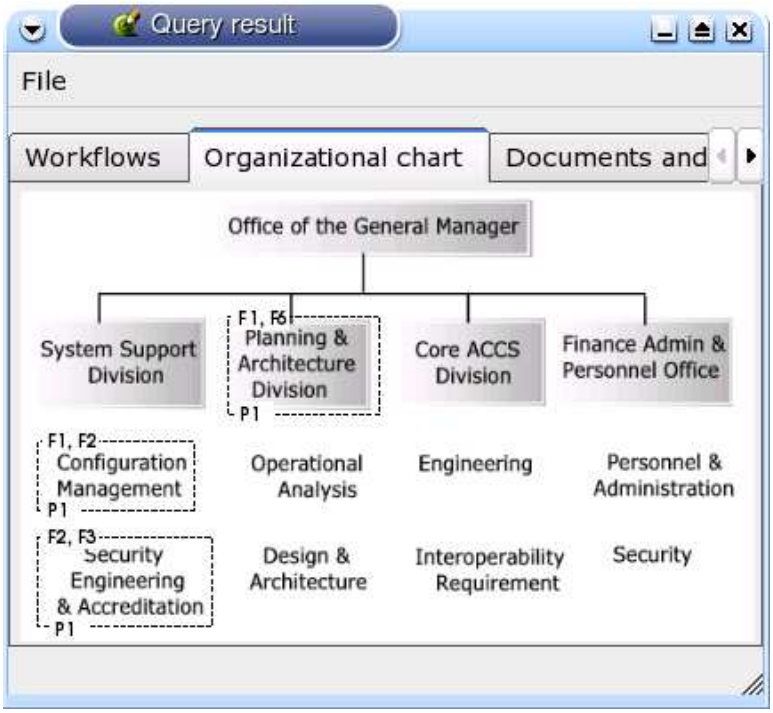

Figure 4.51: Example of a graphical view of the result of a query.

is shown in a tab. The selected tab corresponds to the representation based on the organizational chart. The area of three teamwork objects in the organizational chart is delimited with a dashed rectangle. It means that the three objects are involved in the result of the query.

Forces, perspectives, and breakdowns in the result of a query are represented by drawing and labeling the areas occupied by one or more related teamwork objects. To represent a force, the area of the teamwork objects it involves are drawn and labeled with a reference to the force. In the figure, the area occupied by the organizational unit "Configuration Management" is marked and labeled "F1, F2" to indicate the presence of forces "F1" and "F2" in the result. To represent a perspective, the areas of all objects involved in any of the forces contributed by the perspective are drawn and labeled with a reference perspective. The label "P1" under "Configuration Management" indicates the presence of perspective "P1" in the result. To represent a breakdown, the objects to be drawn and labeled after the breakdown are all those involved in a force that was found relevant for the diagnosis, or that are involved in a force that was affected by the solution. Other elements in the result of the query are not visually represented (i.e., objects of the classes Diagnosis and Solution). A teamwork object can be contained in more than one representation. Therefore, the graphical view can be composed of several images (i.e., several tabs).

The user can set the color to be used to draw areas in order to better match the colors already in use in the underlying image. All areas of all objects are drawn in the same color.

Users commonly draw teamwork representations so they can fit on a printed page or on a single screen. This results in drawings that do not leave much empty space for extra annotations such as labels. For this reason, the label that represents an element

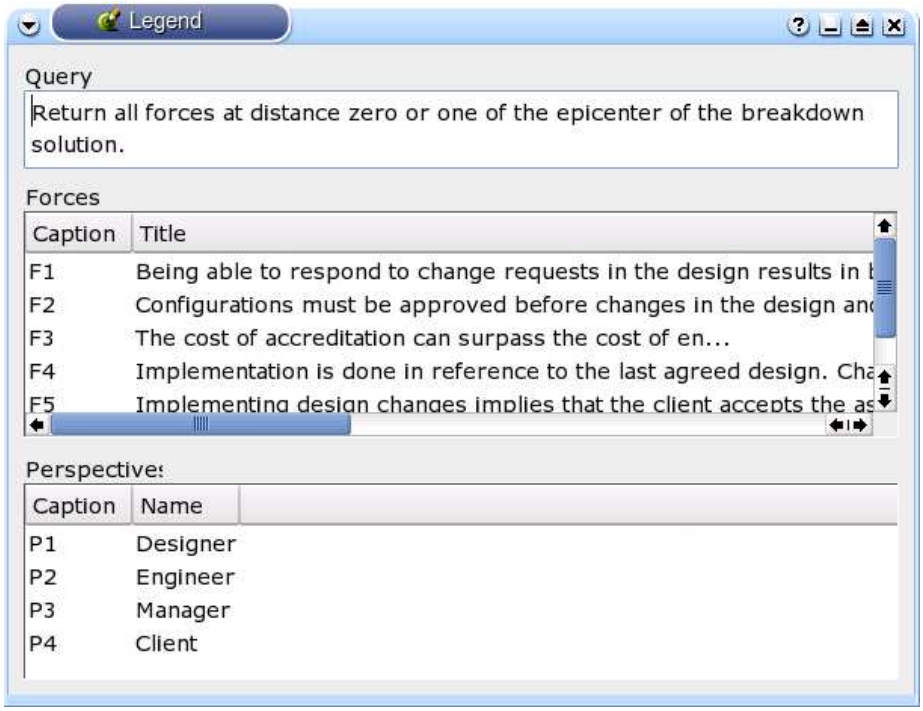

Figure 4.52: Legend window for the graphical view of Figure 4.51

in the result of a query is constructed by joining an upper case letter to indicate the type of element and a number automatically generated for the view. The letter "F" is used as the prefix for forces, the letter "P" is used for perspectives, and the letter "B" is used for breakdowns. The numbers that follow the prefix are in the range between 1 and the amount of elements of the given class in the result query. The legend window shown in Figure 4.52 explains the association between labels and elements. The numbers are newly generated for each view (instead of using some numeric value that is persistently associated to each element) to avoid introducing these cryptic labels into the vocabulary of the organization.

A teamwork object can be used to represent more than one element of the same type in the result of the query (this is the case of "Configuration Management" in the example in Figure 4.51). For example, an object may be involved in more than one force. In this case, the label of the object is constructed as a comma-separated list of the otherwise single-element labels. Moreover, a teamwork object can be used to represent elements from different types, for example, if a query results in forces and in the perspectives that documented these forces. In this case, two labels are generated. One label for one type of elements, located in the upper boundary of the corresponding areas and the other label for the other type of objects, located in the lower boundary of the corresponding areas. In case of a query that results in three types of objects (i.e., forces, perspectives, and breakdowns) to be represented by labels, the user configures the view to indicate what type is represented in the upper label, what type is represented in the lower label, and what type is left out of the view.

The legend window lets the user select an element. The label of the selected element is highlighted in the teamwork representation. Moreover, the legend window provides a mechanism to indicate single elements to be left out of the view, or to specify

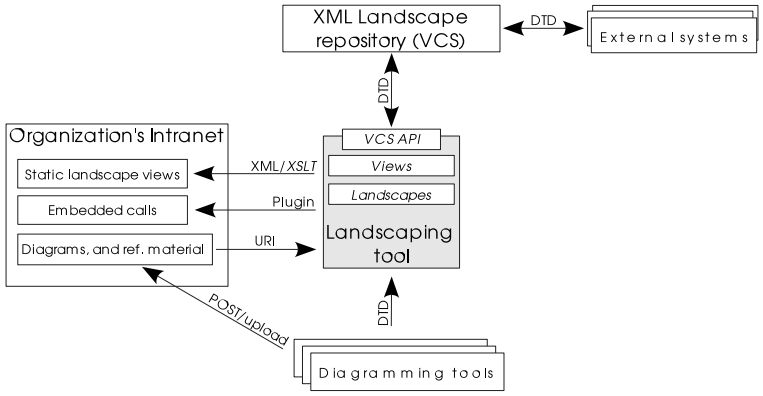

Figure 4.53: Architecture overview of the landscaping tool

colors for their labels. This functionality aims at helping the user find elements in the image, and further focuses communication. The graphical view provides functionality to export the image and the legend in HTML format so that the result of the query can be used in other documents or as the focus of on-line communication.

The Organizational Memory

The Breakdown Landscape pictures the organization as the interplay of forces, perspectives and representations of teamwork and the product (i.e., diagrams plus meta-data). The resulting picture is a dynamic one. It changes when the tool is used to document forces and when teamwork is changed as the result of implementing a solution to a breakdown. The picture that was used to create a breakdown report, is potentially different from the picture obtained when handling of the reported breakdown finishes.

As discussed in Section 2.3, artifacts that result from handling breakdowns are useful for handling future breakdowns. Some of these artifacts need to be interpreted in the context in which they were produced (e.g., effort logs). The Breakdown Landscape is an important part of such context. The Breakdown Landscape tool provides functionality to tag versions for later retrieval (method `tag` of the plug-in class in Figure 4.47) and to retrieve versions with given tags or revision numbers[9]. Method `checkoutRevisionReadOnly` retrieves a given revision of the Breakdown Landscape document. Method `checkoutTagReadOnly` retrieves the revision that corresponds to a given tag. The landscape tool can normally be used to explore and query revisions. However, changes can only be made (e.g., add new forces) when working with the latest version (i.e., after an update). Revision numbers are also used to label the results of queries. The history of all revisions of the Breakdown Landscape provides a form of organizational memory, where each revision captures the team's understanding of teamwork at a particular point in time.

Architecture

Figure 4.53 provides an overview of the architecture of the Breakdown Landscape tool. It shows the landscaping tool (with gray background) with its main components, and, around it, the systems that interface with the tool.

[9]Every time the Breakdown Landscape is modified and saved, a new revision number is automatically assigned

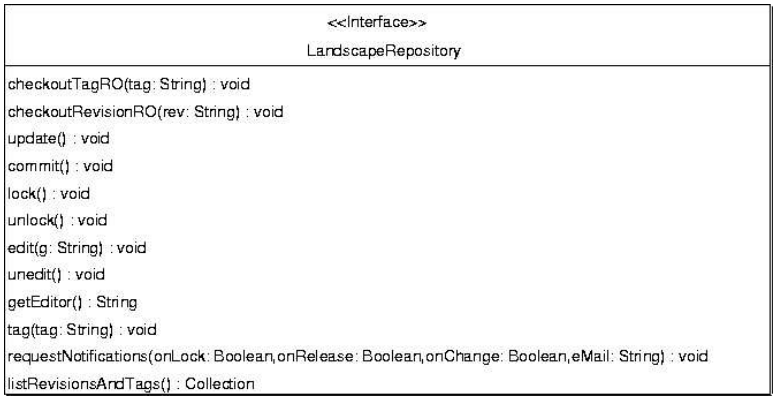

Figure 4.54: Summary of operations available on the landscape repository.

The tool is designed following the philosophy proposed in the introduction of the approach section, which is to provide small (core) units of data and functionality that can be used to achieve a particular teamwork objective. The landscape (as a document) is the data unit on which the tool is based. The tool itself provides functionality to edit and dynamically explore the landscape. There are other systems around the landscaping system (data plus tool) that provide or borrow services.

The XML landscape repository is used to store the landscape. The repository provides a subset of the basic functionality available in most version control systems (VCS). The UML diagram shown in Figure 4.54 presents a summary of the available operations. The method `update` is used to retrieve the last version of the landscape available on the server. The method `commit` uploads the changes made to the landscape in the client to the server. A new revision is automatically created. The methods `checkoutTagRO` and `checkoutRevisionRO` retrieve, in read-only mode, the revision in the server that corresponds to a given tag or revision number. Methods `lock` and `unlock` allow managing a lock on the landscape. The methods `edit` and `unedit` allow managing an edit lock on the landscape. Before modifying the landscape (i.e., adding new forces) users must acquire an edit lock. The method `getEditor` returns the name of the user that holds the edit lock on the landscape. The method `requestNotifications` registers a users preferences for lock, edit, and change notifications. The repository stores user preferences and sends e-mail notifications when events occur. Method `listRevisionsAndTags` returns the list of all available revisions numbers and tags available for the landscape.

The repository is additionally used as the mechanism to make the landscape data available to other applications. In order to allow interoperation with other applications, the landscape is stored in XML format, following the DTD shown in Figure 4.55.

The Breakdown Landscape tool is deployed via web-browser plug-in technology allowing a seamless integration (through embedded calls) with the organization's intranet. Graphical and textual views can be exported in conformance with the HTML 2.0 standard, which additionally allows integration of these views in other documents. HTML is generated by processing the XML data of the view with Extensible Stylesheet Language Transformations (XSLT).

The organization's intranet (or any http-server) is used to host reference material about representations, teamwork objects, breakdowns, and perspectives. Reference material is included in the landscape in the form of URIs. The Breakdown Landscape tool can retrieve these documents and present them to the user in an external web-

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT landscape (perspective*, object*, fact-or-argument*,
        system-of-forces*, teamwork-representation*, breakdown*)>
<!ATTLIST landscape
title CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
<!ELEMENT perspective EMPTY>
<!ATTLIST perspective
        id ID #REQUIRED
        name CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
<!ELEMENT object EMPTY>
<!ATTLIST object
        id ID #REQUIRED
        name CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
<!ELEMENT fact-or-argument EMPTY>
<!ATTLIST fact-or-argument
        id ID #REQUIRED
        type (fact | argument) #REQUIRED
        statement CDATA #REQUIRED
        identified-by IDREF #REQUIRED
        involved-objects IDREFS #REQUIRED>
<!ELEMENT system-of-forces EMPTY>
<!ATTLIST system-of-forces
        id ID #REQUIRED
        statement CDATA #REQUIRED
        identified-by IDREF #REQUIRED
        constituent-forces IDREFS #REQUIRED
        additionally-involved-objects IDREFS #REQUIRED>
<!ELEMENT teamwork-representation (object-representation*)>
<!ATTLIST teamwork-representation
        id ID #REQUIRED
        title CDATA #REQUIRED
        image-uri CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
<!ELEMENT object-representation (shape-path+)>
<!ATTLIST object-representation
        object IDREF #REQUIRED>
<!ELEMENT shape-path (point+)>
<!ELEMENT point EMPTY>
<!ATTLIST point
        x-coordinate CDATA #REQUIRED
        y-coordinate CDATA #REQUIRED>
<!ELEMENT breakdown (diagnosis, solution)>
<!ATTLIST breakdown
        title CDATA #REQUIRED
        reference-uri CDATA #IMPLIED>
<!ELEMENT diagnosis EMPTY>
<!ATTLIST diagnosis
        reference-uri CDATA #IMPLIED
        facts IDREFS #IMPLIED
        arguments IDREFS #IMPLIED
        systems-of-forces IDREFS #IMPLIED>
<!ELEMENT solution EMPTY>
<!ATTLIST solution
        reference-uri CDATA #IMPLIED
        facts IDREFS #IMPLIED
        arguments IDREFS #IMPLIED
        systems-of-forces IDREFS #IMPLIED>
```

Figure 4.55: DTD used by the landscaping tool

browser. Moreover, the same mechanism serves the purpose of storing and retrieving the diagrams corresponding to the teamwork representations in a landscape.

As explained in Section 4.3.7, external diagramming tools are used to create teamwork representations. The images corresponding to the representations are made available through the organization's intranet. The representation descriptors are directly imported by the landscaping tool.

Tailoring Hooks (continuation)

The functionality that the Breakdown Landscape tool needs from the repository, specified as an interface in Figure 4.54, can be found in existing VCS. As seen on Figure 4.56 the Breakdown Landscape tool server relies on an implementation of the LandscapeRepository interface that adequately provides the repository functionality. There can potentially be one implementation of the repository interface for each existing VCS (e.g., CVS, Starteam, and Source Safe). A configuration file provides the name of the class that should be used as an implementation. In this way, the organization can choose the VCS that best fits its needs. A repository implementation plays three roles. First, it acts as a proxy (Proxy pattern [26]) of the external object/server providing the version control functionality. Second, it adapts (Adapter pattern [26]) the protocol provided by the external object to the protocol expected by the landscape server. Third, it decorates (Decorator pattern [26]) the external object with functionality required by the interface that the external object does not provide. For example, CVS does not provide support for e-mail notifications on locks, therefore the CVSRepositoryImplementation needs to store notification preferences and manage notifications. This form of customization aims at simplifying the integration of the tool with the existing organization's infrastructure. Moreover, the version control system of an organization is a software likely to be subject to change. Being able to adapt the landscaping tool to work with other VCS additionally contributes to tailorability.

4.5.4 Artifact: Breakdown Diagnosis

Intent

The intent of this artifact is to document the forces that define the breakdown.

Instructions

The document presents information about the forces that team members find relevant for the breakdown in three separate sections. Figure 4.57 presents the class BreakdownDiagnosis that serves as the template for the artifact. The document is created and extended with the Breakdown Landscape tool, the Co-Scale tool and the Cause Finder tool.

The first section presents the results of the contribute forces activity (i.e., all documented forces). The content of the section is obtained with a Breakdown Landscape tool query that returns all forces in the diagnosis of the breakdown including contributing perspectives and involved objects. The query is opened as a textual view and directly copied to the diagnosis document.

The second section presents the result of the weight forces activity. For each perspective that identified forces, a subsection presents the identified forces in an ordered list of importance. Forces are ordered based on weights of relative importance provided by the members of each perspective. The content of this section is obtained with the export results menu option of the Co-Scale tool.

The third section presents the results of the identify causes activity, that is, the lists of all the forces that team members found unresolved. These forces are the cause

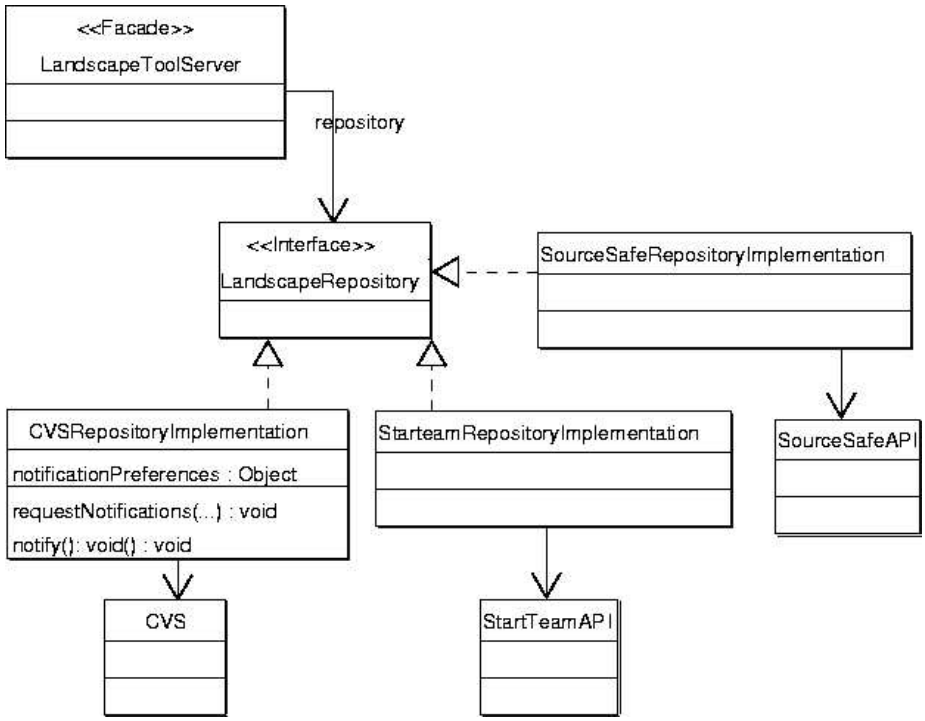

Figure 4.56: Implementing the landscape repository with existing VCS.

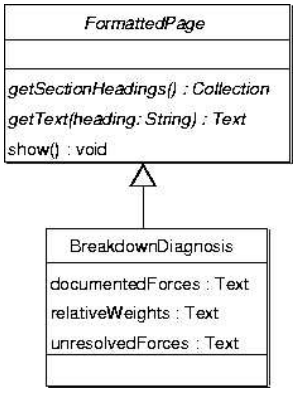

Figure 4.57: FormattedPage subclass for the breakdown diagnosis artifact

of the breakdown. The content of this section is obtained with the export results menu option of the Cause Finder tool.

The breakdown diagnosis document is important for the design and evaluation phases.

4.5.5 Activity: Call for Contributors

Intent

The intent of this activity is to identify further team members whose participation can be useful and to invite them to contribute.

Activation Conditions

This activity can be started when the activity decide (Activity in Section 4.4.12) of the definition phase is completed, and the decision is to proceed with breakdown handling.

Instructions

This aims at applying the insight about the reach of the breakdown gained during contribution of forces to maintain a list of contributors. It occurs in parallel to the contribution of forces. Newly invited team members are advised to take part in contributing forces.

As for the activity invite of the definition phase, the Breakdown Landscape (Tool in Section 4.3.7) can be used to query for perspectives that could make a contribution and whose members have not yet been invited. When a force is documented it may involve additional objects that are not on the key objects set of the breakdown report (these objects were already used to invited people during the definition phase). These additional objects can also be used to query the landscape for potential participants. The moderator (Participant in Section 4.3.4) must stay alert for changes in the set of involved objects to keep the contributors list up to date.

Completion Conditions

This activity finishes when the contribute forces activity (Activity in Section 4.5.2) is completed (as the Breakdown Landscape cannot be of further help).

4.5.6 Activity: Weight Forces

Intent

The intent of this activity is to augment the list of forces with an indication of the relative importance that each force has for the perspective that identified it.

Activation Conditions

This activity can be started when the activity contribute forces (Activity in Section 4.5.2) and the activity call for contributors forces (Activity in Section 4.5.5) are completed.

Instructions

Weighting is performed with the Co-Scale tool (Tool in Section 4.5.7). The tool allows each of the contributors (Participant in Section 4.4.3) to line up all the forces

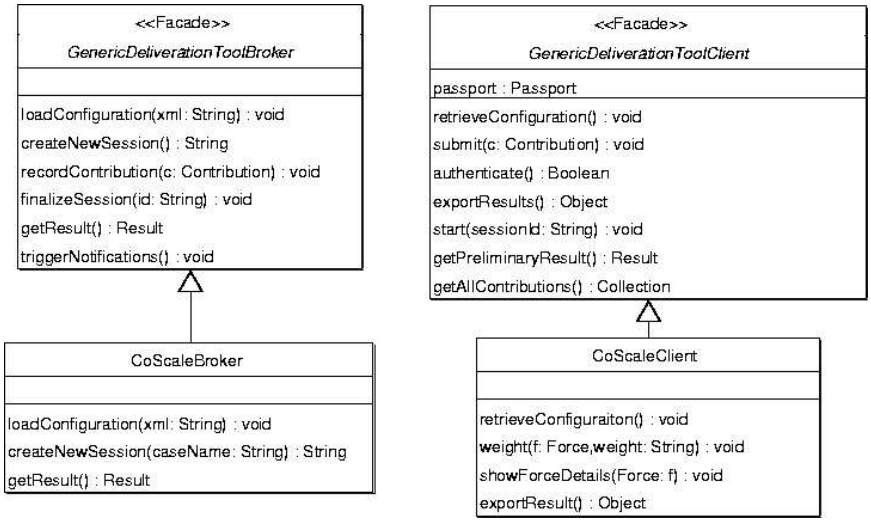

Figure 4.58: Class diagram for the CoScale broker and client objects

contributed by the perspective the participants belongs to according to a scale of importance. The contributions from all participants in a perspective are aggregated. The moderator sets a deadline for the duration of this activity. The deadline is based on records from past experiences (if available).

Completion Conditions

This activity is completed on the deadline set by the moderator. The moderator can decide to reschedule the deadline, for example, to finish earlier if it is clear that no more contributions can be obtained, or to finish later if having more time would contribute to obtaining a more representative result.

4.5.7 Tool: Co-Scale

Intent

The Co-Scale tool is used by team members of a perspective to create an ordering of the forces of their interest according to importance.

Instructions

The Co-Scale tool is a simple deliberation tool similar to the ones already described. It uses the generic architecture for loosely coupled deliberation tools documented in section 4.4.7. Figure 4.58 shows the concrete broker and client classes that specialize the abstract classes provided by the generic architecture.

Part of the tool's configuration is the list of forces documented for the breakdown being diagnosed and information about participation of team members in perspectives. Contributors weight forces and the tool aggregates the weights.

Forces are presented in lists. Contributors individually assign each force a weight in a range of values previously configured. A greater weight means that the force is of greater importance. When negotiating solutions, the team members' intention is to solve forces starting with those of higher importance. The collection of allowed

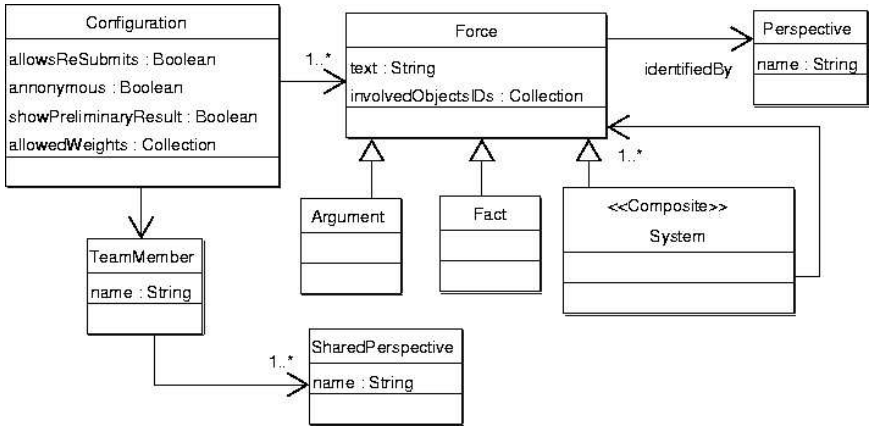

Figure 4.59: Configuration class for the CoScale tool

weight values, obtained from the configuration object shown in Figure 4.59, is sorted by importance. The first value means "most important" and the last value means "least important".

The list of forces to weight (the forces documented for the breakdown being treated) needs to be obtained from the Breakdown Landscape tool (see Section 4.5.3). This list needs to be loaded as part of the configuration of the CoScale tool together with the remaining configuration options such as allowing resubmits and keeping contributions anonymous. Forces are maintained as part of the configuration object. The method `loadConfiguration` inherited from GenericDeliverationToolBroker loads the configuration options form an XML file located in the broker. The CoScaleBroker class overrides the `loadConfiguration` method to additionally retrieve the last version of the landscape XML file from the landscape repository. Information about forces is obtained from the XML file.

The XML file obtained from the Breakdown Landscape tool contains information about all forces documented for all existing breakdowns (past or currently being handled). Each breakdown in the landscape is identified with a case name (attribute `caseName` of the Breakdown class). To build the configuration object, the tool's broker needs to filter out all forces from breakdowns other that the current one. When the session of the CoScale tool is created, for example, when the new instance of the scaffolding is created for the case corresponding to the current breakdown, the case name is passed as an argument to the `createSession` method of the broker. This method extends the `createSession` inherited from the superclass. The broker maintains an association between the session name and the case name, and uses it to correctly assemble the configuration.

Team members only weight the forces that were added to the breakdown diagnosis by a perspective which they belong to. The tool presents the forces contributed by each perspective to which the participant belongs in a separated tab pane as shown in Figure 4.60. The list of perspectives for the participant using the CoScale client needs to be loaded in the tool's configuration. This part of the configuration is specific for each user. The CoScale client class overrides the method `retrieveConfiguration` to additionally pass the username to the server. In this way, the server can assemble a specific configuration for this user. The name of the user is retrieved from the authentication passport in the inherited `authenticate` method.

The associations between team members and perspectives that the CoScale broker loads in the configuration object need to be retrieved from the scaffolding server (see

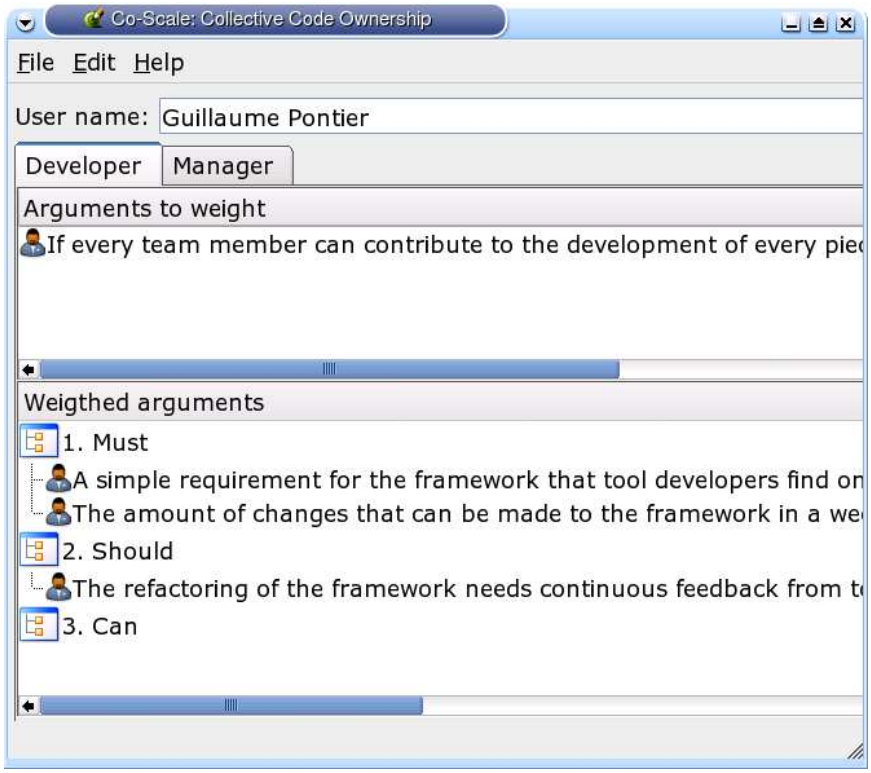

Figure 4.60: Simplified GUI design for Co-Scale

Section 4.2.2). The `shared` name-space of the scaffolding server contains the nodes that represent team members and shared perspectives. The scaffolding server is extended to serve HTTP GET requests to the URI *http://domain-name/shared/xmlDump*. The result is an XML file containing all team members, all shared perspectives, and the associations between them.

Each of the tabs presented to the user is divided into two sections. The upper section shows all forces contributed by the perspective that still need to be weighted. The lower section shows, in a first level, the allowed weights obtained from the configuration object (in the picture *Must*, *Should*, and *Can*). Under each weight value, in a second level of the list, the list shows all forces that have been given that weight. Users can drag elements from one section to the other to assign or reset weights (method `weight` of the CoScaleClient class). An icon (a person for arguments, gears for systems, and the square root of X for facts) indicate the type of force (in the figure there are only arguments).

Double clicking on a force opens the details dialog (method `showForceDetails)`. The details dialog shows all details of the force (i.e., full text, contributing perspective, and involved objects).

A menu option "Submit" submits the contribution of the user. The submit action is enabled only when all forces from all perspectives have been weighted. Figure 4.61 shows the CoScaleContribution class that specializes the class Contribution provided by the generic framework. The contributions groups the forces according to the weights that were assigned.

The broker keeps record of all contributions. The method `getResult` in class

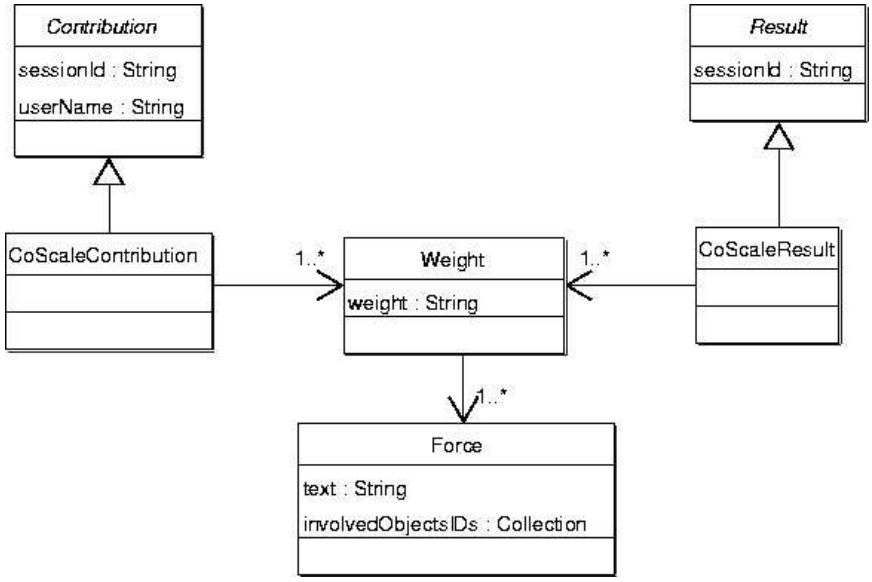

Figure 4.61: Configuration and result objects for the CoScale tool.

CoScaleBroker aggregates the contributions force by force in a CoScaleResult. The aggregated weight of a force is calculated by taking the most frequent weight given (i.e., the mode). If this value is not unique, the greater (most important) of the most frequent values is taken.

A menu option "View result" opens a separate window with a list similar to that in the lower section of the main window. The list is now built from the preliminary aggregated result. The weights shown in the list represent the aggregation of the contributions from all team members that already submitted. Double click on a force gives access to the dialog window that displays the details of the force and to a diagram where all contributions are shown as dots over an axis. The diagram can be used to get an idea of how dispersed the contributions were.

When the activity finishes, the tool can export the results for integration into a scaffolding artifact such as the breakdown diagnosis. A menu option "Export result" requests that the result of the tool be store in an HTML file.

Tailoring Hooks

The small subset of functionality provided by the Co-Scale tool can be tailored by changing the ordered set of values used for weighting. Some factors must be considered when exploring alternatives of value sets. The size or cardinality of the set directly impacts the experience of weighting and the results. If the set is too large, users may have problems choosing a value. If the set is too small, the resulting lists may be of no use for decision making. The vocabulary used to construct the values must be chosen with care. The difference between weights should be clear and leave no space for ambiguity (numeral values have this characteristic). Moreover, the vocabulary should make it easy to choose a value (domain terminology has this characteristic). Further guidelines can be found in the literature on survey design (e.g., [22]).

The tool can be configured to allow re-submits. If that is the case, a user can submit a new contribution that will replace any previously given one. The view results dialog can be disabled for users other than the moderator.

4.5.8 Activity: Identify Causes

Intent

The intent of this activity is to identify, from the forces that have been reported, those that are perceived by team members as the cause of the breakdown.

Activation Conditions

This activity can be started when the activity weight (Activity in Section 4.3.8) is completed.

Instructions

The final activity of the diagnosis phase is to identify the forces that team members perceive as the cause of the breakdown. The moderator sets a deadline for the duration of this activity. The deadline is based on records from past experiences (if available).

Identification of causes is done with the Cause Finder tool (Tool in Section 4.5.9). The tool allows contributors (Participant in Section 4.4.3) to indicate the forces that they see as the cause of the breakdown. Based on the individual contributions from team members, the tool builds an aggregated result.

To identify causes, forces need to be observed in relation to the information contained in the breakdown report (Artifact in Section 4.3.6). First, the team member must carefully read and analyze the breakdown report (symptoms, context and impact). Then the team member reads through the lists of facts and arguments. If a contextual fact is seen as a possible cause of the breakdown, it is marked as unresolved. Then, each force of type argument in the list is considered. As discussed in Section 2.3.4, an argument associates a choice for a tailoring hook with the impact with respect to a common value of selecting this choice instead of other possible choices. If the choice that is effective at the moment of the breakdown is seen as a possible cause, the force is marked as unresolved. The state of systems of forces is automatically derived from the state of constituent forces (it is resolved if and only if all its constituent forces are resolved).

A team member that evaluates a force as unresolved must also provide a short argument for this decision. During the design phase arguments are used as hints to explore the possible alternatives. During the evaluation, the argument is used as a reminder of the criteria used when the force was first tested.

The aggregated result is exported from the Cause Finder tool and included in the corresponding section of the breakdown diagnosis artifact.

Completion Conditions

This activity is completed on the deadline set by the moderator. The moderator can reschedule the deadline, for example to finish earlier if it is clear that no more contributions can be obtained, or to finish later if having more time would contribute to obtaining a more representative result.

4.5.9 Tool: Cause Finder

Intent

The intent of the Cause Finder tool is to obtain a list of the forces that team members consider to be the cause of the breakdown.

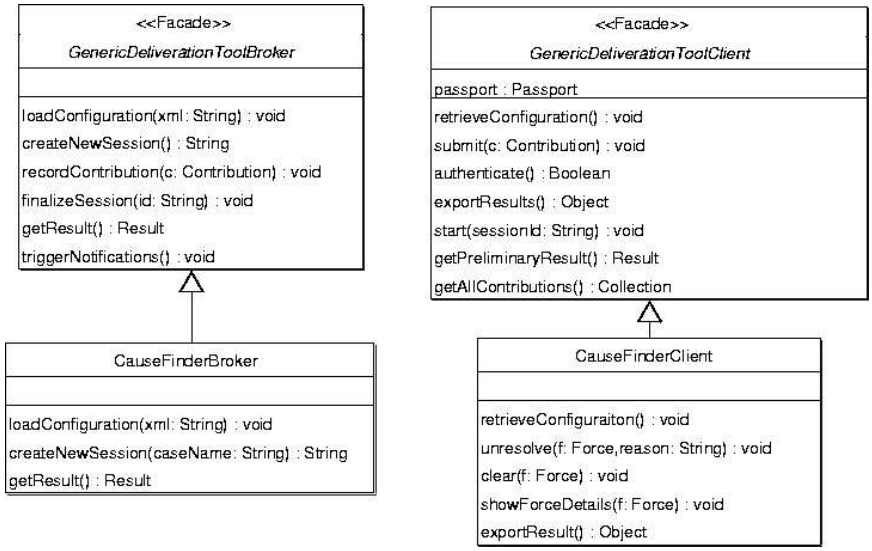

Figure 4.62: Class diagram for the Cause Finder broker and client objects

Instructions

The Cause Finder tool is a simple deliberation tool. Team members independently and asynchronously mark forces as causes of the breakdown (thus, as unresolved) and submit the list of causes. The contributions of all team members are aggregated in a result that is later taken as the team's evaluation of possible causes of the breakdown.

The Cause Finder uses the generic architecture for loosely coupled deliberation tools documented in section 4.4.7. Figure 4.62 shows the concrete broker and client classes that specialize the abstract classes provided by the generic architecture.

The tool takes as as part of its configuration the list of forces documented for the breakdown being diagnosed. As for the Co-Scale tool, the list of forces needs to be obtained from the Breakdown Landscape tool. This list needs to be loaded as part of the configuration. The CauseFinderBroker class overrides the `loadConfiguration` method to retrieve the last version of the landscape XML file form the landscape repository. Information about forces is obtained from the XML file. Figure 4.63 shows the configuration class for the Cause Finder.

Figure 4.64 depicts the user interface for the Cause Finder tool. Facts, arguments, and systems of forces are listed in separated tab panes. Forces that the user indicated as unresolved are marked with an X icon. Double clicking on a fact or argument opens the details dialog. The dialog shows all details of the force, and allows the user to change its state. If the user sets the state to unresolved, a text box to provide a reason is enabled. In this field, the user must explain why this force should be considered a cause of the breakdown. Providing a reason is mandatory. The method `unresolve` of the client class adds a force to the list of unresolved forces in the user's contribution (i.e., causes). The method `clear` removes a force form the list.

A menu option "Submit" submits the contribution of the user. To calculate the result, contributions are aggregated force by force . If any user has found a force to be a cause, then the force is included as a cause in the aggregated result. The notes provided by users as arguments are appended to a list (attribute `reasons` of class AggregatedCause). Figure 4.65 shows the contribution and result classes for the Cause Finder tool.

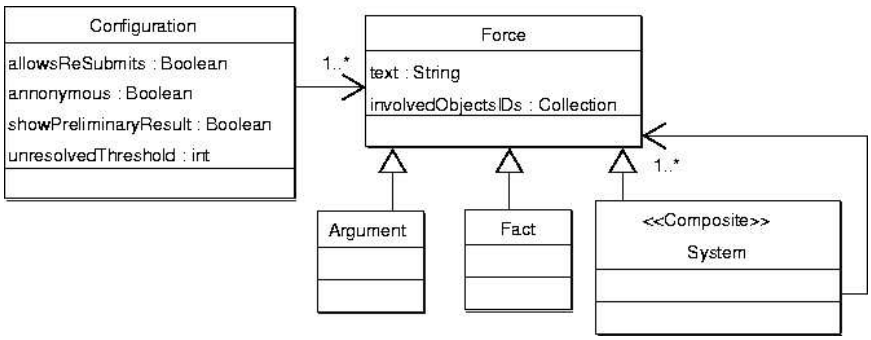

Figure 4.63: Configuration class for the Cause Finder tool

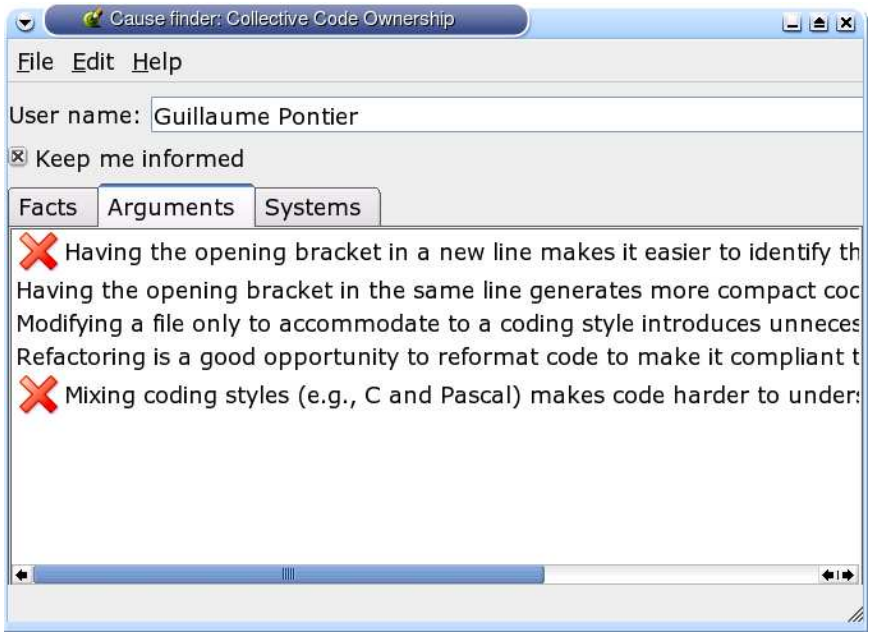

Figure 4.64: GUI design for the of Cause Finder tool

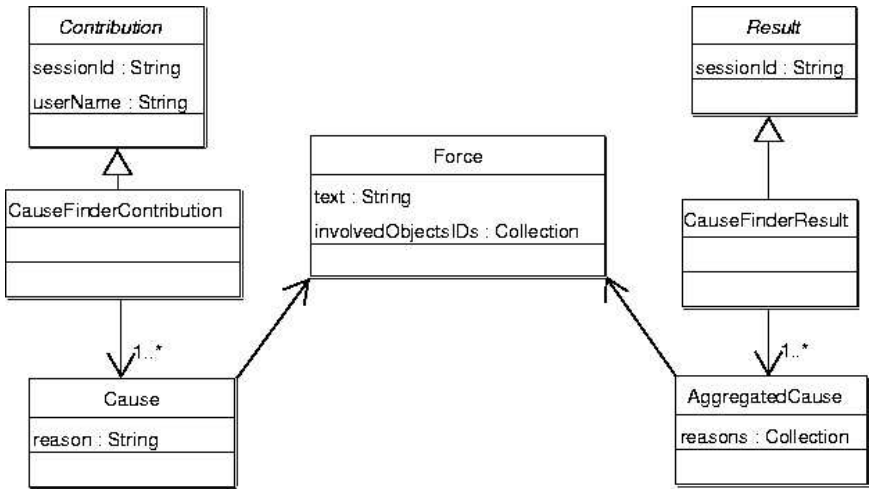

Figure 4.65: Configuration and result objects for the Cause Finder tool.

A menu option "View result" opens a separate window with the three tab panes. The states of forces are taken from the aggregated result. Double clicking on a force gives access to the dialog window that displays the details of the force, the number of users that see the force as unresolved, and the list of notes arguing for the unresolved states. Selecting a menu option "Export result" saves the result of the tool in an HTML file.

Tailoring Hooks

The tool can be configured to allow re-submits. If that is the case, a user can submit a new contribution that will replace any previously given one. In the default case, it is enough that one user sees the force as unresolved to get the force as a cause in the result. An integer threshold value can be set to configure the number of users that must find a force as unresolved to make it be unresolved in the result. Finally, the view results dialog can be disabled for users other than the moderator.

4.5.10 Activity: Review Diagnosis

Intent

A breakdown diagnosis has been completed. Allow the inclusion of newly discovered forces or weights, assuring that the resulting diagnosis still reflects the team's agreed understanding of the problem.

Activation Conditions

The review activity must be started if the request diagnosis review activity of the design phase (Activity in Section 4.6.9) was conducted and resulted in a decision to review the diagnosis.

Instructions

As previously said, the breakdown diagnosis is the team's agreed understanding of the problem. It is the basis for the work of teams developing solution alternatives. After the diagnosis phase is completed, no further changes should be made to the breakdown diagnosis. In the event that changes are required, all team members need to be given the opportunity to review their contributions and make further contributions.

When this activity starts, the moderator (Participant in Section 4.3.4) changes the state of activities contributing forces and calling for contributors to *InProgress*, and the state of activities weight forces, identify causes and log effort to *Inactive*. All activities are executed once again. Existing contributions (e.g., forces from the latest execution of the phase) are maintained.

Completion Conditions

This activity can be marked as completed when all other activities in the phase have been completed.

4.5.11 Activity: Log effort

Intent

The intent of this activity is to document the effort spent in a phase.

Activation Conditions

This activity can be started as soon as the activity identify causes (Activity in Section 4.5.8) is completed and the activity Review Diagnosis (Activity in Section 4.5.10) is completed or has not been started.

Instructions

In the definition phase, team members estimate the effort of doing diagnosis, design, treatment, and follow-up evaluation. Based on this estimate and on an estimate of the relevance of the breakdown, they decide whether handling a breakdown is worthwhile. To estimate effort, the moderator (Participant in Section 4.3.4) and the contributors (Participant in Section 4.4.3) benefit from any records of effort spent in previous cases. In order to provide that information it must first be collected. This is the goal of effort logging. During effort logging, each participant is asked to report in the effort log artifact (Artifact in Section 4.5.12) the time spent (in work hours) for each of the activities in the current phase. Participants log effort at the end of the phases of diagnosis, design, treatment, and follow-up evaluation.

The effort log is better interpreted in conjunction with the participants list and the document of the breakdown forces. For example, it can be used to infer how much effort in a phase relates to the aspects of teamwork that the case is dealing with and to the perspectives that are involved. However, this cannot be taken as a statistical proof. In the best case, one could try to infer trends in the complexity of cases that deal with a specific aspect of team work or that involve a given perspective (or a given number of perspectives).

Completion Conditions

The moderator should try to keep the effort log as complete as possible. The moderator decides when to mark this activity as completed. In the best case, this activity is complete when every member that participated has reported effort.

Phase	Activity	Total
Diagnosis	-	3
-	Contribute forces	2
-	Call for contributors	1
-	Weight forces	0
-	Identify causes	0
-	Review diagnosis	0
Design	-	0
-	Develop alternatives	0
-	Recruit	
-		

Figure 4.66: Part of the table showing effort totals

4.5.12 Artifact: Effort Log

Intent

The intent of this artifact is to document the effort spent by team members in diagnosis, design, treatment, and follow-up evaluation of the breakdown.

Instructions

The effort logs consists of two sections. The first section presents the total reported efforts, activity by activity and phase by phase. The second section provides links to all individual reports created by team members.

Total reported efforts are presented in a table like the one partly shown in Figure 4.66. The first column lists the phases, the second column lists the activities, and the third column presents the totals.

To create and maintain the effort log artifact, one could define a specific deliberation tool based on the generic architecture already presented. However, the logic underlying the effort log is very limited. Each team member works on a separated piece of data (the individual effort log). There is no need for locking or notifications of changes, and there is no complex aggregation logic. Moreover, the individual effort log data can be easily edited as an HTML form without the need of a specific application. The effort log artifact can be created as a specialization of the FormattedPage class of the scaffolding server. Individual effort reports are also specializations of the FormattedPage class. This approach is depicted in the class diagram in Figure 4.67.

The method `getTableOfTotals` of class EffortLog generates the HTML code that is used to build the table previously described. It relies on the methods `calculatePhaseEffort`, `calculateActivityEffort`, and `calculateTotalEffort`. These methods retrieve all individual effort reports linked from the effort log and calculate the totals. The edit button that is present in all scaffolding pages is disabled for the effort log artifact because its content is calculated dynamically from the linked individual reports.

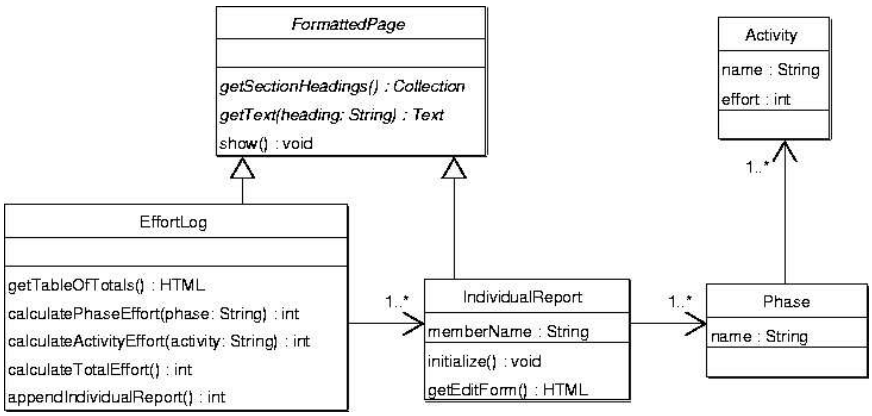

Figure 4.67: FormattedPage subclasses for the effort log artifact

The second section of the effort log presents links to all individual reports. Additionally, there is a button to append a new individual report. The method `appendIndividualReport` creates an instance of IndividualReport and links it. The method retrieves the name of the current user from the authentication passport (accessible, for example, through HTML scripting). The `initialize` method of the IndividualReport class creates and initializes the placeholders to report the effort of each activity. The edit button of the IndividualReport presents an HTML form (rendered in method `getEditForm`) with an entry field for each value to be provided.

4.5.13 Summary

Requirement 4 in Section 2.3.4 argued for teamwork support to document and evaluate the forces that shape the breakdown. The Breakdown Landscape tool allows team members from several perspectives to collaboratively identify and record the forces that shape the problem. The moderator uses the Breakdown Landscape to identify contributors that must be invited to contribute, and to assure that diagnosis considers the opinion of all important stakeholders. The Co-Scale tool allows team members to collaborate in the construction of a prioritization of forces. Finally, the Cause Finder tool allows team members to collaboratively find unresolved forces that seem to cause the breakdown. The breakdown diagnosis artifact records the team's understanding about the forces, priorities, and causes of the breakdown.

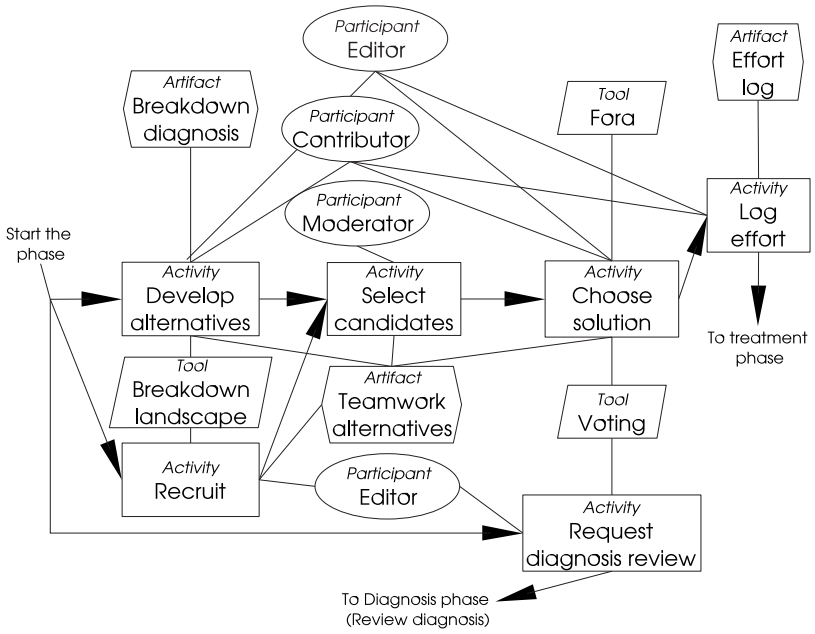

Figure 4.68: Design phase of breakdown handling

4.6 Designing Solutions

4.6.1 Overview of the Design Phase

The design phase aims at the generation of proposals of alternatives for teamwork and the selection of one of them as the best candidate to resolve the breakdown. Figure 4.68 provides an overview of the design phase. The six activities that define this phase are: develop alternatives, recruit, select candidates, choose solution, log effort, and request diagnosis review. The moderator sets deadlines for the completion of each of the activities.

4.6.2 Activity: Develop Alternatives

Intent

The intent of this activity is to produce several solution alternatives.

Activation Conditions

This activity can be started as soon as the activity Log Effort (Activity in Section 4.5.11) of the diagnosis phase is completed.

Instructions

In this activity, team members collaborate to design a solution for the breakdown. Collaborative design is itself an area of research within CSCW (see, for example, the work done by Arias and colleagues [9, 8, 10, 7]). For the scope of this thesis, the development of alternatives is seen as an atomic activity that results in several solution

alternatives. The focus of this section is on what needs to be done specifically to assemble a teamwork alternative document (Artifact in Section 4.6.3). Team members working on this activity can additionally benefit from a bibliography on team creativity techniques (e.g., [24, 6]) for the creation of alternatives.

The breakdown diagnosis (Artifact in Section 4.5.4) is used to inspire alternatives that have the potential of resolving the breakdown. There is no limit to the number of alternatives that can be proposed. However, team members should only propose a new alternative when there is no hope to contribute to any of the existing ones to make it follow the desired direction. A proposal that integrates contributions from many perspectives has more chances to correctly satisfy the forces.

A teamwork alternative is created by contributors (Participant in Section 4.4.3) in asynchronous collaboration. All members participating in the proposal can contribute to the teamwork alternative document. The editor (Participant in Section 4.6.5) distributes responsibility for the different parts of the document. When all contributions are ready, the editor edits the document and may request changes or corrections. On-line forums can be used to host design discussions. On-line discussions that require focusing on a common artifact (e.g., a representation of teamwork) can be supported with chat tools that support referencing drawings from chat messages (e.g., [20]). Argumentation tools such as gIBIS [21] can be used to organize design conversations and capture early design deliberations.

The exploration of the space of possible alternatives of solutions is guided by the forces that need to be solved. A solution is built from many interdependent decisions. A decision (e.g., a choice of a tool) that aims at solving a group of forces (among all the forces that need to be solved) may limit the the possible decisions to solve another group of forces. The explored part of the solution space shaped by the decisions taken at each point by team members can be described with a decision tree [55] like the one shown in Figure 4.69. The root of the tree represents the starting point of the exploration. Team members may decide to first tackle forces X, Y, and Z by taking a decision A (e.g., adopt tool N). As a result of this decision, they have a new (hopefully smaller) set of forces to solve and, therefore, a new set of alternative paths to follow. They are represented by the node labeled 1 in the figure. As an alternative to decision A, they could have taken decision C. Decision C could be a different way to tackle the same groups of forces as decision A, or could tackle a different set of forces. The decisions that lead from the root to a leaf (where all forces have been solved) form a solution. How to collaborate to explore the solution space (i.e., how to do collaborative/cooperative problem solving) is outside the scope of this thesis (see [77] for a discussion on cooperative problem solving from the point of view of artificial intelligence). However, it can be expected that the team's decisions are based on some utility function that lets them choose among alternative paths and eventually among alternative solutions. Such a utility function likely involves minimizing the cost, maximizing the number of high priority forces that are solved, and minimizing the side effects.

A decision mainly aims at solving a force or group of forces from the set of forces identified as causes in the breakdown diagnosis. Additionally, the decision can consider new forces not stated in the breakdown diagnosis. For example, Force 11 in the breakdown presented on page 16 was found during design. It states "Refactoring is a good opportunity to reformat code to make it compliant to a particular style". These forces must be documented as design forces in the teamwork alternative document (Artifact in Section 4.6.3). They are documented with the Breakdown Landscape tool and copied to the document from a textual view.

During the design, team members may need to see the impact that a decision may have using the Breakdown Landscape (Tool in Section 4.3.7). As the Breakdown Landscape describes teamwork in its current form, it cannot be modified for exploratory purposes. The moderator creates a branch in the breakdown repository

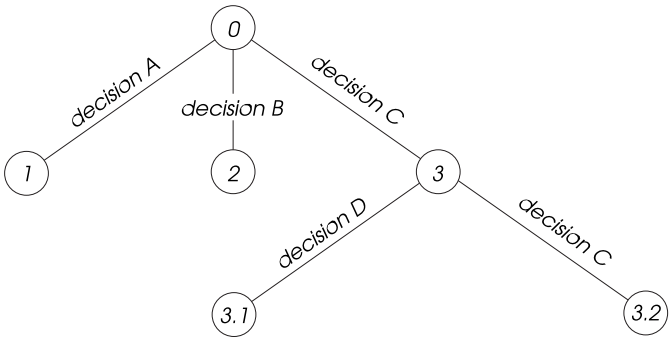

Figure 4.69: Decision tree

of the Breakdown Landscape and names it after the solution being designed. Team members can launch the Breakdown Landscape tool and retrieve the last revision of the branch that the moderator prepared for them. They can modify the branched landscape to represent the solution being designed. They can remove and contribute forces, create and import new representations, and perform queries.

Solutions take time to show impact. The pace in which the results of the suggested changes will be perceived depends on the extent of the changes and the nature of the breakdown. During the preparation of an alternative, the proposing group agrees on a period to wait after changes have been made and before a follow-up evaluation can be done in order to examine the impact of the changes. A follow-up plan is created and documented with the teamwork alternative document. The follow-up plan consists of a collection of evaluation activities and the date when they should take place. The choice of evaluation activities depends on the problem being solved and the proposed solution. These activities aim at confirming the effect promised by the solution and assuring that no unexpected side effects appear. In cases where deciding is a subjective issue that depends on the perception of team members, it may be enough to open a dedicated forum for a limited period of time to give team members the opportunity to report their observations. During this period, team members read the breakdown report and the teamwork alternative artifact. If they consider that something is wrong with the solution, they can contribute to the forum. A ballot (with some voting tool) can serve to formally decide on the effect of the solution (either solved, failed, or more time is needed). It is possible that for some problems, specific objective evaluations activities need to be designed such as a formal performance evaluation of some area of teamwork.

Completion Conditions

This activity must be completed by the deadline set by the moderator. The moderator can decide to finish it early if all proposals are ready, or to extend it.

4.6.3 Artifact: Teamwork Alternatives

Intent

The intent of this artifact is to capture a solution alternative.

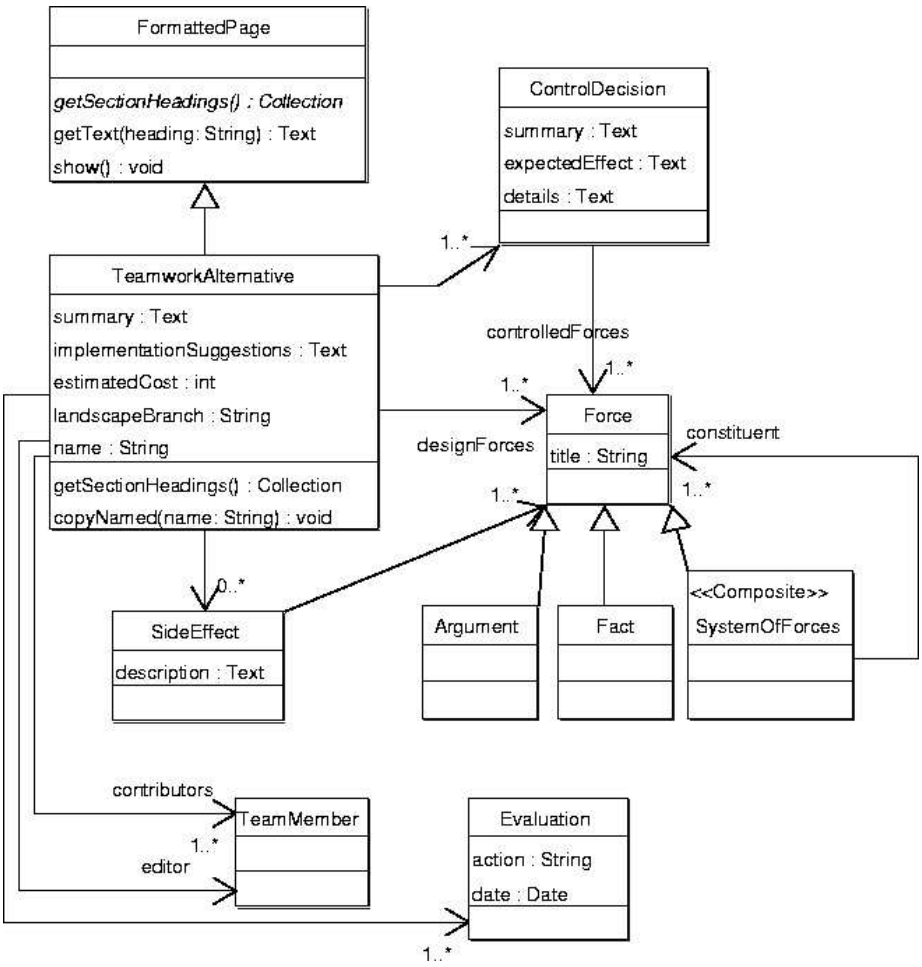

Figure 4.70: FormattedPage subclass for the teamwork alternative artifact

Instructions

A solution alternative artifact captures a proposal for a solution to the breakdown created by a group of team members.

As explained in Section 4.2.2 an artifact node in the scaffolding server is a placeholder for meta-data about the artifact (e.g., for usage instructions). Additionally, the artifact node has a hyperlink to the real resource that holds the concrete artifact's data. The class diagram in Figure 4.17 on page 63 shows that a real resource can be one of ExternalResource, FormattedPage, and UntypedPage. Figure 4.70 shows how the class FormattedPage of the scaffolding server is specialized to model one concrete teamwork alternative resource.

During the design phase, several teamwork alternatives can be prepared in parallel by team members. Each of these alternatives requires an independent real resource for the teamwork alternatives artifact. When a group of team members decides to start working on a teamwork alternative, the moderator creates a copy of the initial real resource (through the method `copyNamed` of the TeamworkAlternative class), and adds a link to it from the artifact's node. As a result, the teamwork alternatives node in the

scaffolding will have links to several real resources. Each alternative is identified by a name. Moreover, each alternative has hyperlinks to the nodes of the team members who contribute to the alternative and to the team member who works as editor.

A solution alternative is expected to resolve the forces that were identified as causes in the breakdown diagnosis. Moreover, a solution may require that new forces are introduced which were not considered during design. A section of the teamwork alternative lists all design forces. A solution alternative resolves the forces identified as causes. Ideally, it should not turn unresolved any other force that was previously solved (e.g., by previous breakdown handling) in the organization's Breakdown Landscape. In reality, team members often reach compromise solutions that solve important breakdowns in detriment of less important breakdowns previously solved. During design, team members must try to identify side effects of solutions and document them in a section of the document.

During design, team members may need to temporarily document design forces using the Breakdown Landscape. These forces can be persistently added to the organization's Breakdown Landscape if the alternative is selected and implemented. To let team members document design forces, a branch of the Breakdown Landscape is created. A branch of the landscape is a copy that can be used independently from the main version of the XML file that holds the data of the Breakdown Landscape tool in the landscape repository. The branch name needed to retrieve the landscape data is stored as part of the alternative document.

The summary section of a breakdown alternative shortly describes the proposed solution. The core of the alternative are the decisions (e.g., actions or changes to teamwork) that the contributors suggest. Each decision aims at solving a particular force or system of forces. A section of the document explains each decision, the forces it aims to solve, and its expected effect on these forces. The text describing each decision and the forces it solves are copied from a textual view (result of a query) of the Breakdown Landscape tool.

For each evaluation activity suggested by the contributing team members, the teamwork alternative artifact includes a description and a suggested date when the action should be carried out. To complete the description of a solution alternative, team members must provide an estimate of the cost of implementing it (in a monetary unit). As an option, the designers of the alternative can provide guidelines and suggestions for its implementation.

4.6.4 Tool: Breakdown Landscape (continuation)

During design, team members conduct exploratory modifications of the Breakdown Landscape. These modifications should not affect the main organization's Breakdown Landscape but should be done on an isolated copy. The LandscapeRepository class presented in Figure 4.54 on page 118 needs to be extended with a method `branch`. The arguments of this method are the name of the new branch (the main branch is named HEAD following the convention used by the CVS version control system) and the existing branch and revision number that should be taken as the base for the copy. Figure 4.71 provides an overview of the relation between branches and revisions. It shows the main (HEAD) branch with its four available revisions. Revision 4 is the latest (i.e., current) revision. Revision 2 was used as the base for a new branch called BR1. Branch BR1 was modified and saved twice, first as revision 2 and then as revision 3. Moreover, the figure shows that revision 2 of branch BR1 was used as the base for a new branch BR2.

Besides the forces that were introduced in the landscape during the diagnosis phase, there are also the design forces. During exploration, design forces need to be added to the landscape branch used by the team. The object model of the Breakdown Landscape tool is extended as shown in Figure 4.72 to consider design forces with an

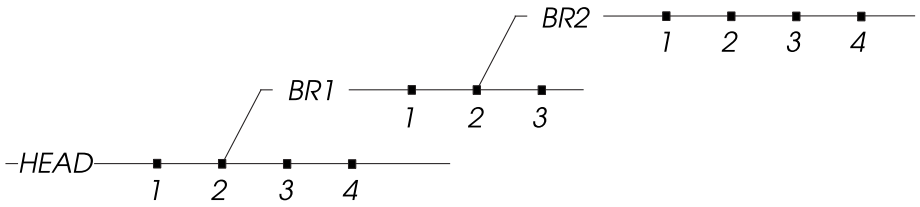

Figure 4.71: Relation between branches and revisions

additional relation (`introduces`) between class Solution and class Force. Section 4.5.3 presented the force editor window of the Breakdown Landscape. It allows forces to be added. The force editor is extended with two radio buttons (labeled "Diagnosis force" and "Solution force") as shown in Figure 4.73 to indicate the type of force being added.

Each decision the team makes is included in the Breakdown Landscape. This is done with the solution editor window shown in Figure 4.74. The solution editor also provides an input field to indicate the name of the solution being documented. This name can later be used to find the teamwork alternative artifact in the scaffolding. Decisions can be added. For each decision, the contributors provide the summary (same text as in the teamwork alternative artifact) and the list of controlled forces. Forces are selected from all forces documented in the Breakdown Landscape. The details of each decision are only documented in the teamwork alternative artifact.

4.6.5 Participant: Editor

Contribution

An alternative is developed by a group of participants. One of them is designated as the editor. The editor is responsible for coordinating the effort of the proposing group and for editing the final version of the teamwork alternative document.

Requirements

In cases where the designation of the editor is not straight forward, it can be resolved by voting (simple majority) among the volunteering candidates or the project manager can appoint an editor.

4.6.6 Activity: Recruit

Intent

The intent of this activity is to recruit team members that do not yet contribute but their contribution may be of help.

Activation Conditions

This activity can be started as soon as the activity identify causes (Activity in Section 4.5.8) of the diagnosis phase is completed.

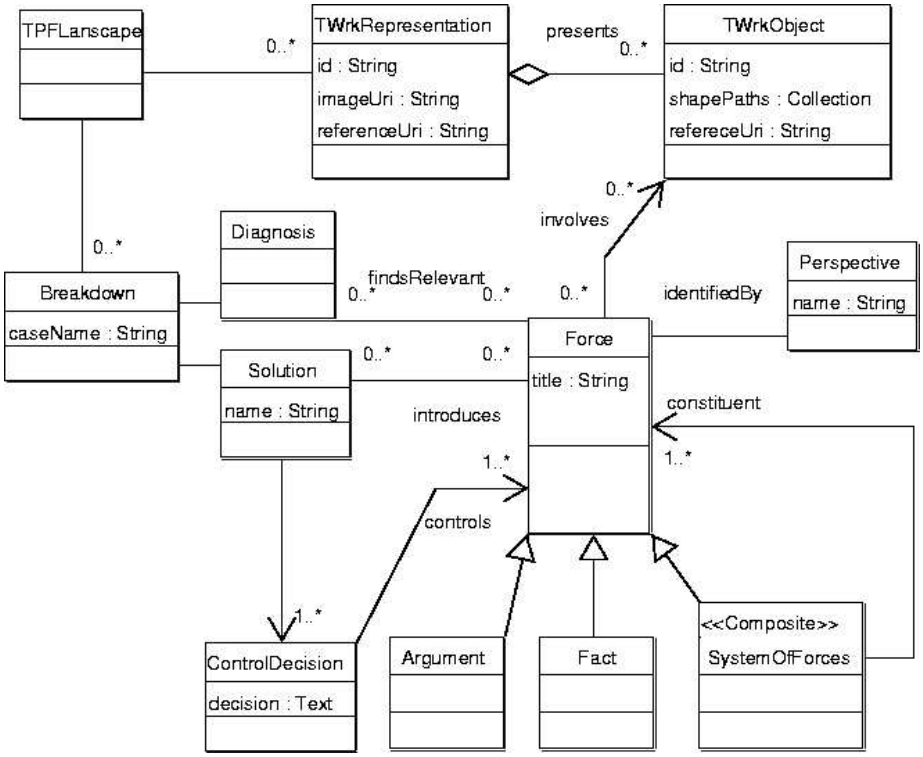

Figure 4.72: UML diagram for the object model of the Breakdown Landscape

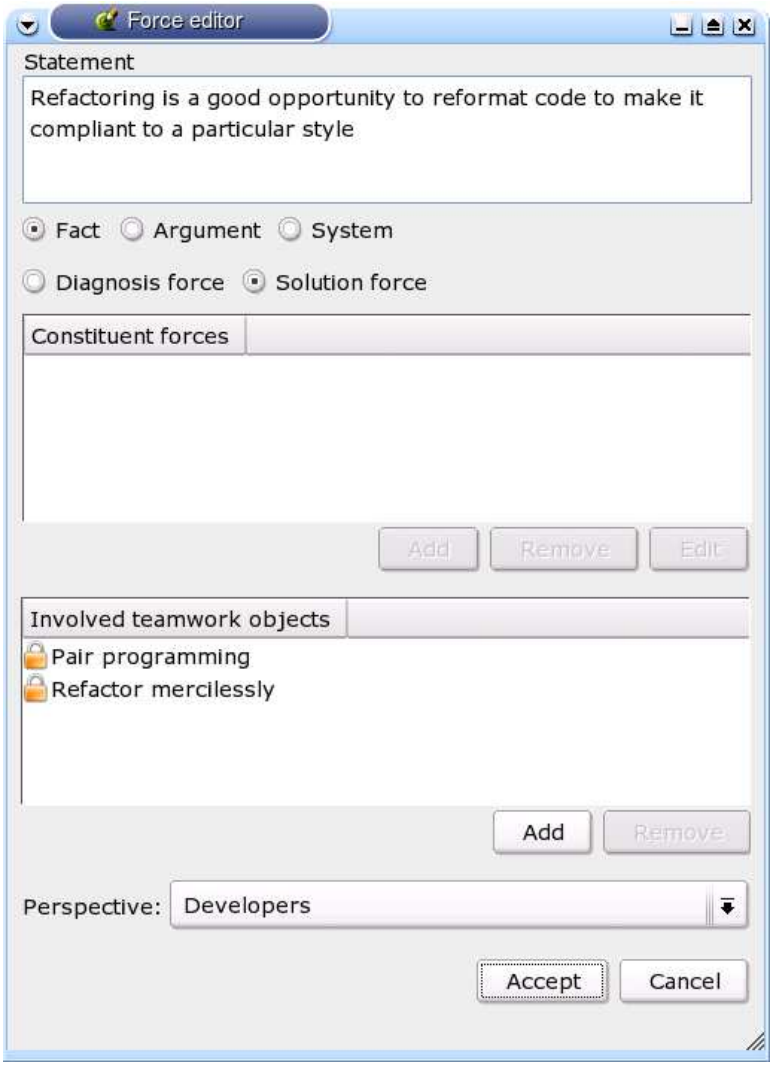

Figure 4.73: GUI design for the "Force editor"

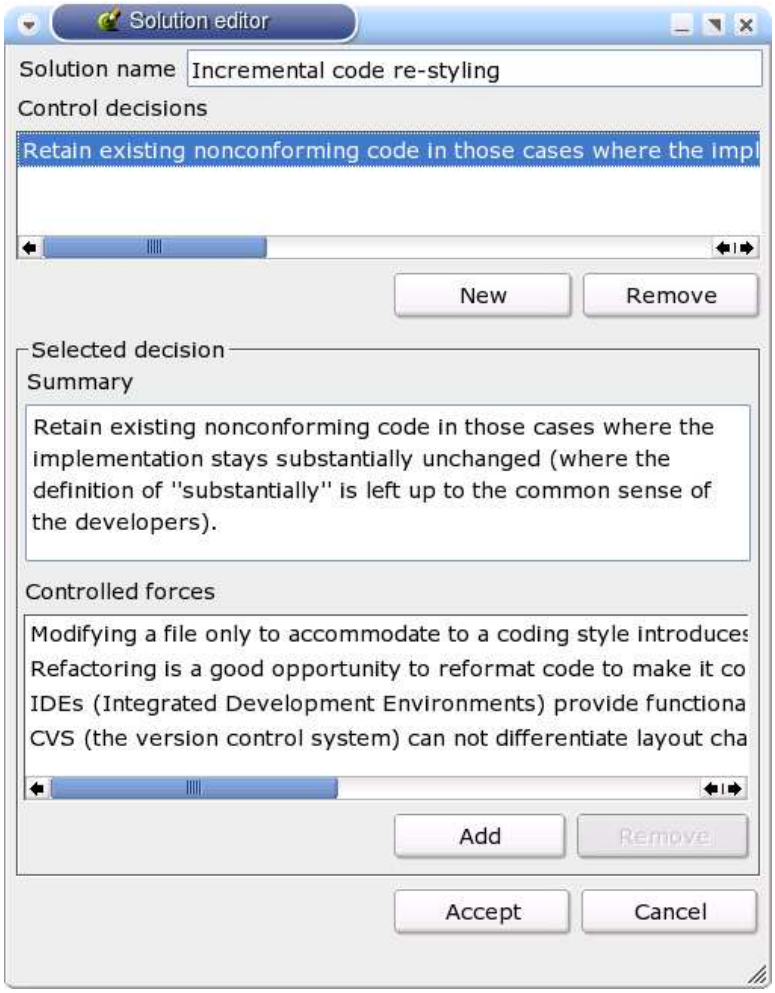

Figure 4.74: GUI design for the "Solution editor"

Instructions

Any team member can decide to start the creation of a solution alternative. It is recommended that at least one team member from each perspective that contributed forces to the breakdown diagnosis participates in the design of the alternative.

When a solution is designed, it may involve objects of teamwork that were not involved in the breakdown report or in the forces that were identified in the diagnosis. These objects can result in identifying team members whose participation may be of help and who are not yet contributing. The editor (Participant in Section 4.6.5) of a teamwork alternative (Artifact in Section 4.6.3) must periodically check with the Breakdown Landscape (Tool in Section 4.3.7) if the objects that are affected by the solution lead to perspectives that are not yet involved. If this is the case, at least one team member of these perspectives should be convinced to contribute.

Completion Conditions

This activity is completed when the activity develop alternatives is completed.

4.6.7 Activity: Select Candidates

Intent

The intent of this activity is to select those alternatives that clearly outperform the other available alternatives.

Activation Conditions

This activity can be started when the activities develop alternatives (Activity in Section 4.6.2) and recruit (Activity in Section 4.6.6) are completed.

Instructions

As a result of the develop alternatives activity, several teamwork alternatives (Artifact in Section 4.6.3) may be produced. The mechanism presented in section 2.3.5 to find non-dominated alternatives is used to select the best candidates. The moderator (Participant in Section 4.3.4) compares all alternatives according to the set of resolved forces, the cost, and side effects. Let $fResolved()$ be a function that returns the set of forces from the breakdown diagnosis that an alternative resolves. An alternative "A" is better than an alternative "B" with respect to fResolved(), if fResolved(B) is strictly included in fResolved(A). "A" is as good as "B", if fResolved(A) equals fResolved(B). Let $estimatedCost()$ be a function that returns the cost estimated by the team for the implementation of a given alternative. An alternative "A" is better than an alternative "B" with respect to estimatedCost(), if estimatedCost(A) is strictly less than estimatedCost(B) minus the "largest insignificant difference" (a threshold value agreed by the moderator and the manager to indicate that differences in cost below that threshold are not significant enough to qualify one alternative as cheaper or more expensive than the other). Let $knownSideEffects()$ be a function that returns 0 if there are no known side effects for the alternative, 1 otherwise. An alternative "A" is better than an alternative "B" with respect to knownSideEffects(), if knownSideEffects(A) is strictly less than knownSideEffects(B). An alternative "A" is a non-dominated alternative (therefore a candidate) if *there is no alternative "B"* such that "B" is as good as "A" regarding knownSideEffects(), fResolved(), and estimatedCost(), and it is better than "A" regarding at least one of them.

Completion Conditions

This activity is completed when the moderator has obtained the set of candidates.

4.6.8 Activity: Choose Solution

Intent

The intent of this activity is to select one alternative for its implementation.

Activation Conditions

This activity can be started when the activity select candidates (Activity in Section 4.6.2) is completed.

Instructions

If the select candidates activity yields only one candidate, that candidate is chosen for implementation. If it yields several candidates, contributors (Participant in Section 4.4.3) select one by voting. The alternative (Artifact in Section 4.6.3) with more votes is chosen. All team members can cast one vote. Team members can change their votes anytime until the deadline set by the moderator. Voting is done with a Voting Tool (Tool in Section 4.4.13).

A discussion forum is provided for each alternative. The forum allows team members who did not participate in the development of the alternative to ask questions to those who did, and make comments regarding their opinions on the impact and cost of implementing the alternative. Team members can change their votes in order to give these discussions the potential to influence the results of the voting.

Completion Conditions

This activity is completed on the deadline set by the moderator.

4.6.9 Activity: Request Diagnosis Review

Intent

The intent of this activity is to decide about conducting a review of the diagnosis of the breakdown.

Activation Conditions

This activity is started if one of the editors requests the moderator to start a review of the diagnosis of the breakdown.

Instructions

The moderator and the editors discuss the request and agree how to proceed. If necessary, the moderator can configure a voting tool to decide on the issue.

If the decision is to conduct a review of the diagnosis, the moderator changes the state of all activities in the diagnosis phase to Inactive.

Completion Conditions

This activity is completed when a decision is made.

4.6.10 Summary

Requirement 5 in Section 2.3.5 argued for teamwork support to design alternatives, plan follow-up evaluation, and select an alternative for its implementation. The Breakdown Landscape tool and the breakdown diagnosis document are used by teams to explore alternatives. As alternatives evolve, the Breakdown Landscape can be used to find other relevant contributors. When proposals are published, all team members can discuss them in on-line forums and make-up their minds as they vote for one. Solution alternatives are created and selected in collaboration.

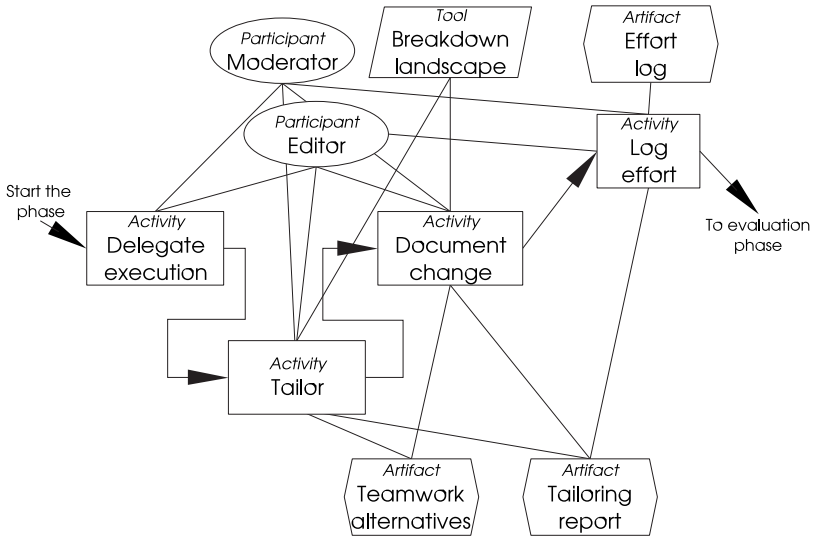

Figure 4.75: Treatment phase of breakdown handling

4.7 Treating the Breakdown

4.7.1 Overview of the Treatment Phase

The phase of treatment has the goal of implementing the changes to teamwork that are required to make it conform to the solution alternative selected during evaluation. Figure 4.75 depicts the phase of treatment. It consists of four activities, namely delegate execution, tailor, document change, and log effort. The treatment itself (i.e., implementing the selected alternative) is done in the tailor activity.

Support for executing the planned changes to teamwork is outside the scope of this thesis. It involves many other technical issues such as tailorability of the affected systems and artifacts, and non technical issues such as planning and executing changes without disrupting the functioning of the system. Moreover, executing the changes may require the participation of external experts (e.g., software experts). The focus of this thesis is on supporting the tasks that can be performed (and may be better performed) by the team members.

The approach taken here to support collaborative breakdown handling is to view application of changes as an externally executed task. A detailed request for changes is delegated to an external group that has the capability to perform the changes. When changes are in place, control returns to the team.

4.7.2 Activity: Delegate Execution

Intent

The intent of this activity is to delegate the execution of the selected alternative to a tailoring team.

Activation Conditions

This activity can be started as soon as the activity Log Effort of the design phase is completed.

Instructions

To delegate execution, the moderator (Participant in Section 4.3.4), or someone designated by the moderator negotiates with a group of implementors (e.g., developers, advance users, handymen) the implementation of the designated changes. The teamwork alternative artifact (Artifact in Section 4.6.3) is passed on to the team of implementors as the requirement specification. During the phase of treatment, the editor (Participant in Section 4.6.5) of the chosen alternative acts as the domain expert.

Completion Conditions

This activity is completed when the team of implementers agrees to implement the requested changes.

4.7.3 Activity: Tailor

Intent

The intent of this activity is to implement the selected alternative.

Activation Conditions

This activity can be started as soon as the delegate execution (Activity in Section 4.7.2) is completed.

Instructions

The team of implementers implements the requested changes as good as possible. In case of doubt, they can ask the editor for clarifications. When the changes are finished, the implementers return to the moderator (Participant in Section 4.3.4) with the tailoring report (Artifact in Section 4.7.4). Additionally, the moderator discusses with the implementers and the editor (Participant in Section 4.6.5) of the implemented alternative if changes to the follow-up evaluation schedule are needed. If that is the case, the moderator updates the plan for evaluation.

During implementation, the team of implementers can introduce additional forces. For example, to implement the use of CVS as the revision control system, a force is introduced: "source code changes are no longer anonymous, but the author can be identified." These forces are documented as solution forces (that relate to control decisions found during implementation) together with any design forces that could have been introduced during design. These forces are documented in the branch of the Breakdown Landscape (Tool in Section 4.3.7) used by the design team.

Completion Conditions

This activity is completed when the moderator accepts the changes implemented by the implementers and has confirmed/updated the schedule the follow-up evaluation.

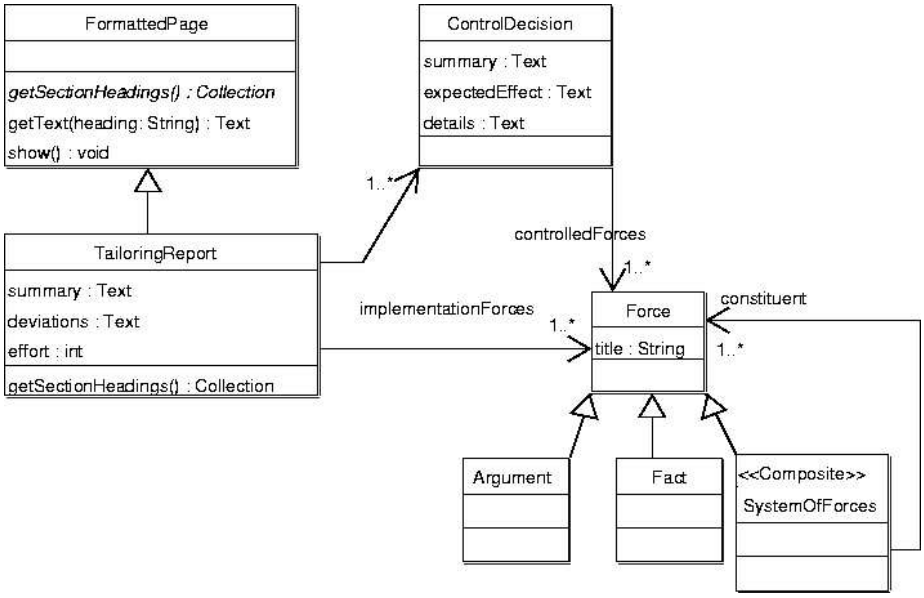

Figure 4.76: FormattedPage subclass for the tailoring report artifact

4.7.4 Artifact: Tailoring Report

Intent

The intent of the tailoring report is to document the results of the work done by the implementers when trying to implement the chosen solution.

Instructions

The tailoring report artifact provides a summary of the results of the implementation of the solution. Any deviations from the requested changes and what the implementers were able to implement are also documented. As in the teamwork alternative document, any newly introduced forces are documented. Figure 4.76 shows how the class FormattedPage of the scaffolding server is specialized to model one concrete tailoring report resource.

4.7.5 Activity: Document Change

Intent

The intent of this activity is to update the representations of teamwork and of the product to reflect the implemented changes.

Activation Conditions

This activity can be started as soon as the tailor (Activity in Section 4.7.3) activity is completed.

Instructions

The moderator (Participant in Section 4.3.4), assisted by the editor (Participant in Section 4.6.5), has the task of updating the representations of teamwork and the Breakdown Landscape tool (Tool in Section 4.3.7) to document the changes documented in the tailoring report (Artifact in Section 4.7.4) and to reflect the new situation. If all design and implementation changes in the representations and the forces were correctly documented in a branch of the Breakdown Landscape, all that needs to be done is to merge the alternative's branch with the HEAD branch.

If the diagrams included in the proposal were created with teamwork editors, and if the changes were implemented in total compliance with the request, updating the representations of teamwork can be as easy as replacing the existing diagrams by those included in the proposal.

During the diagnosis and evaluation phase, new systems of forces and factual forces were identified. The evaluation phase resulted in a solution proposal that considered these forces and possibly introduced control decisions for some of the uncontrolled systems. These forces are documented in the Breakdown Landscape tool.

Completion Conditions

This activity is completed when all changes have been documented in the teamwork representations and the Breakdown Landscape.

4.7.6 Tool: Breakdown Landscape (Continuation)

Updating Teamwork

When the organization implements changes in the forms of teamwork, these changes need to be reflected in the representations using the corresponding editors. A change in teamwork may need to be reflected in many representations. Given that these changes occur outside the landscaping tool, they must be imported. Importing a representation that already exists results in an update. The update is based on the unique identifiers of representations and objects.

Changes in teamwork may result in the introduction of new objects or in the removal of existing ones. References from forces to any removed object (i.e., dangling references) are consequently deleted. Moreover, after each import, a report is generated. The report (a text file) lists all objects that were imported without problems, all objects that were removed without problems, all objects that were removed and caused forces to have dangling references, all forces with dangling references, and all forces that reference an object that changed (e.g., an existing object that now appears in a new presentation). The report can be reviewed to see if any forces have become obsolete or need to be corrected. The landscaping tool provides no other support to deal with conflicts generated by imports.

During design and treatment, new forces are introduced (solution forces). Changes in representations and forces are first performed on a branch of the Breakdown Landscape. The team members that modify the landscape are responsible for keeping it consistent (e.g., update forces with dangling references). In section 4.3.7, importing representations is described as a server-side operation. However, as it needs to work on a landscape branch, it must be implemented as a client (plug-in) side operation. After the solution has been implemented, the moderator takes the XML file of the Breakdown Landscape data that corresponds to the last revision of the branch and commits it to the HEAD (main) branch. In this way, the changes become visible in the organization's Breakdown Landscape.

4.7.7 Activity: Log effort

Intent

The intent of this activity is to document the effort spent in this phase.

Activation Conditions

This activity can be started as soon as the document change (Activity in Section 4.7.5) activity is completed.

Instructions

The editor (Participant in Section 4.6.5) and the moderator (Participant in Section 4.3.4) indicate in the effort log (Artifact in Section 4.5.12) the time they spent during this phase. Moreover, the moderator also indicate in the effort log the effort that the implementors reported in the tailoring report.

Completion Conditions

This activity is complete when the effort log is updated.

4.7.8 Summary

The activities proposed in this section tackle requirement 6 that deals with the provision of support to spawn the tailoring project and resume breakdown handling when the tailoring project is completed. The teamwork editors and the Breakdown Landscape are used to document the changes that were implemented.

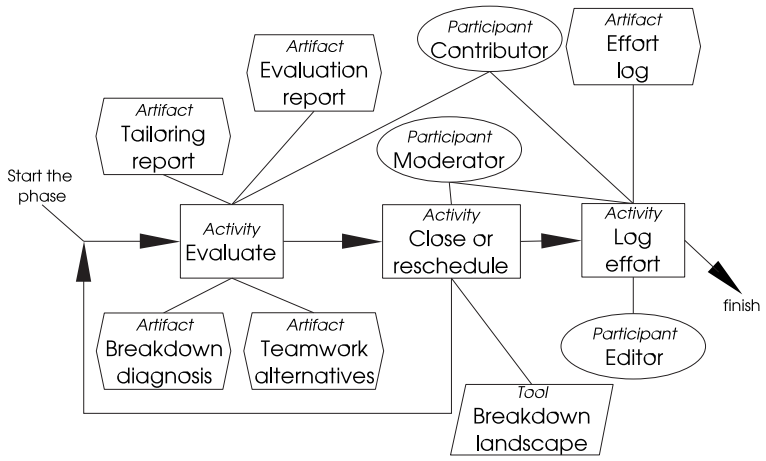

Figure 4.77: Evaluation phase of breakdown handling

4.8 Evaluating the Solution

4.8.1 Overview of the Evaluation Phase

The evaluation phase has the purpose of evaluating whether the changes had the desired effect on the forces. Figure 4.77 depicts this phase.

4.8.2 Activity: Evaluate

Intent

The intent of this activity is to evaluate the state of resolution of the forces involved in the breakdown.

Activation Conditions

This activity is started on the dates scheduled in the teamwork alternative artifact.

Instructions

On the dates indicated for each of the evaluation activities indicated in the teamwork alternative artifact of the implemented solution, the moderator sets the state of this activity to *InProgress*. Contributors (Participant in Section 4.4.3) collaborate to perform the evaluation activity as indicated in the teamwork alternative artifact (Artifact in Section 4.6.3). The results of the evaluations are documented in the evaluation report artifact (Artifact in Section 4.8.3).

After the last planned evaluation activity is completed, the moderator and the editor infer the overall result of the evaluation. In principle, the evaluation succeeds only if all individual evaluation activities succeeded. However, how the results of the individual activities add up to the overall results depends on the planned activities.

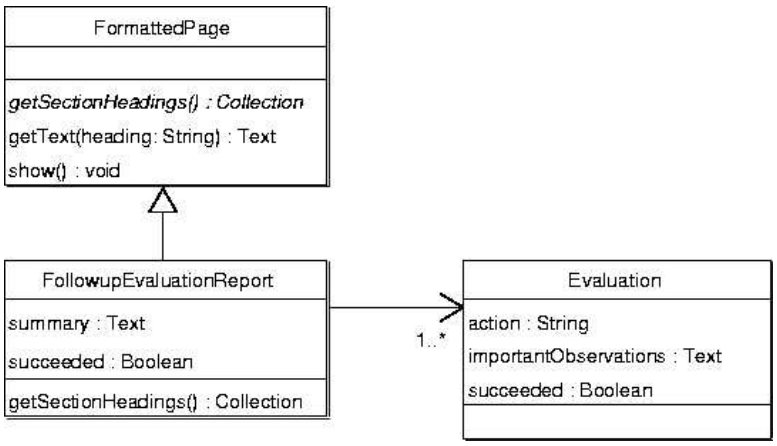

Figure 4.78: FormattedPage subclass for the evaluation report artifact

4.8.3 Artifact: Evaluation Report

Intent

The evaluation report documents the results of the evaluation activities.

Instructions

The content of the evaluation report tightly depends on the evaluation activities that are performed. In all cases, the first section of the report indicates whether the team concluded that the breakdown was solved (i.e., success) or that the breakdown was not solved (i.e., failure). Additionally, one section for each evaluation activity indicates the result and provides observations that team members consider important to record. Figure 4.78 shows how the class FormattedPage of the scaffolding server is specialized to model one concrete evaluation report resource.

4.8.4 Activity: Close or Reschedule

Intent

The intent of this activity is to decide how to react in response to the results of evaluation.

Activation Conditions

This activity takes place when the evaluate activity (Activity in Section 4.8.2) is completed.

Instructions

In the case of a successful evaluation, the moderator (Participant in Section 4.3.4) closes the case. In the case of a failed evaluation, the moderator (in discussion with the editor of the chosen proposal) can decide to give the solution more time to take effect and consequently re-schedule some or all of the evaluation activities. Alternatively, they can decide to close the case as it is. If the forces that stay unresolved are of

importance, closing the case will eventually cause a breakdown and a new breakdown handling case would start.

A solution introduces changes to teamwork and, potentially, new forces. If the solution fails to solve the breakdown, keeping all these changes and forces represents unnecessary cost. However, it may not be possible to undo them (e.g., the team has appropriated a newly introduced tool and refuses to abandon it). This unnecessary change and forces will eventually cause a breakdown and a new breakdown handling case would start, under new circumstances. To handle the new breakdown, a solution may be to undo all changes and go back to a previous, problematic situation. Other solutions may try to work from taking the changes as part of the new situation.

When the case is closed, the effort of the last phase is documented and all documents are frozen. The result of the case is also documented in the Breakdown Landscape (a boolean flag in the Breakdown object). Documenting the result of breakdown handling in the landscape makes it possible to query, for example, for all forces that were introduced and are still in effect due to a failed solution.

Completion Conditions

This activity is completed when the case is closed. If evaluation is rescheduled, the state of this activity is set to Inactive.

4.8.5 Summary

This section discusses the provision of support for collaboratively evaluating the result of implementing the chosen solution. Thus, requirement 7 is covered.

Chapter 5

Implementation

The prototypes of the tools were implemented as the need for them arose during usage experiences. Prototypes were implemented as specified in the approach chapter. This chapter discusses some aspects of the implementation that deviate from the specification (for practical reasons) or that may be of special interest to the reader.

5.1 Implementation Architecture

Figure 5.1 summarizes the architecture of the implemented prototype. The scaffolding server (Scaki Server) stores the hypermedia content (e.g., specification and state of the scaffoldings) in an XML-based repository. A file repository is used to store files such as scaffolding artifacts and the plug-ins of tools. The content of the scaffolding is served to regular web-browsers via HTTP. The Scaffolding Overview component is implemented as a Squeaklet (Applet like application implemented with Squeak Smalltalk). It retrieves XML data from the Scaki Server via HTTP. The clients of the deliberation tools (Breakdown In-box, Co-Scale tool, Co-Estimation Tool, and Cause Finder tool) are implemented as Visualworks Smalltalk plug-ins. They exchange data with their corresponding brokers in the form of serialized objects over TCP/IP connections. The Breakdown In-box and the Co-Scale tool additionally retrieve data, in the form of serialized objects over a TCP/IP connection, from the Scaki Server. The same method is used by the Co-Scale tool to retrieve data from the Breakdown Landscape Server. The Breakdown Landscape Server uses an external CVS executable to store and retrieve XML files from its file repository. Serialized objects over a TCP/IP connection is also the communication mechanism between the Breakdown Landscape Server and its Client.

5.2 Scaki: the Scaffolding Server

Swiki [65], an implementation of WikiWikiWeb written in Squeak Smalltalk, has been chosen for the implementation of the prototypical scaffolding server (called Scaki). Swiki provides a simple way of publishing on-line, it provides full text search, has an extensible object model, uses XML storage, provides basic support for asynchronous collaborative document editing (e.g., locks, revisions, change tracking, notifications, security), and provides basic document storage (useful to deliver applications and attach external documents).

Swiki includes markup elements to create headings, tables, and HTML links that point to anchors inside pages. Additionally, Swiki allows the definition of page templates. Each template defines how pages are seen in edit and in presentation mode.

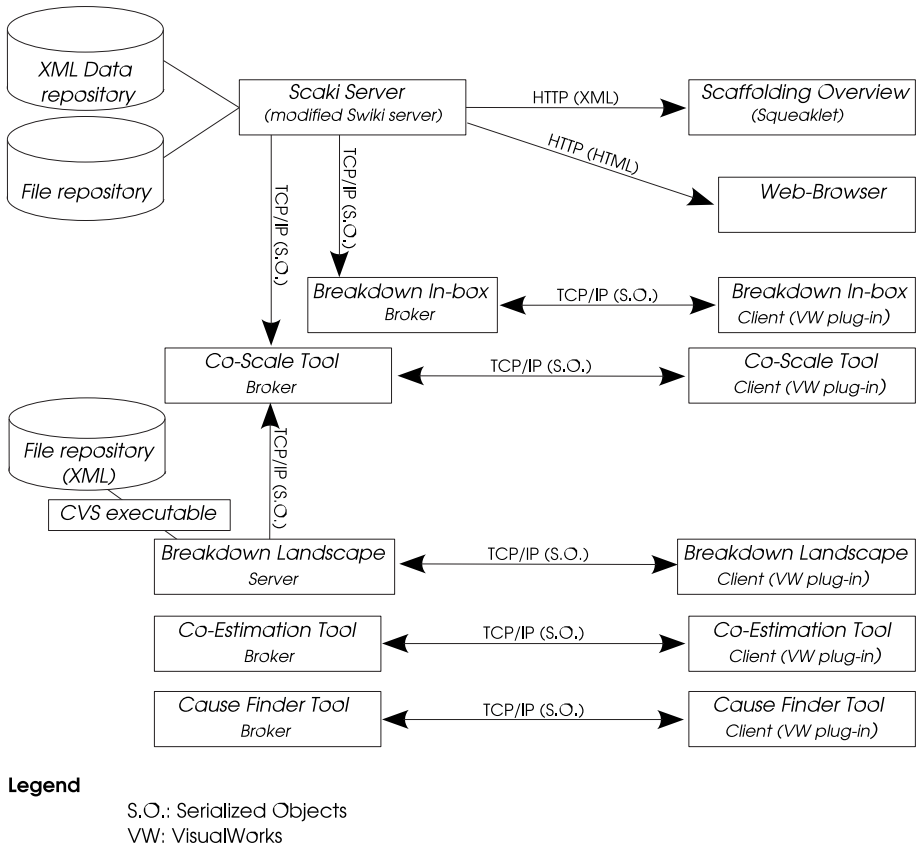

Figure 5.1: Overview of the implemented architecture

The richness of the editing and presentation facilities of Swiki simplified the creation of the edit and presentation pages of the different types of scaffolding elements (especially artifacts). Each of the types of elements in the scaffolding is defined as a template in the Swiki.

Swiki has functionality to calculate, for a page, all incoming references. Originally, the incoming references are listed at the bottom of each page. This functionality was extended, based on a filter that considers the type of the referencing page, to create the lists that appear at the bottom of every element's page. Figure 5.2 presents, as an example, the activity weight of the diagnosis phase in show mode. The first icon in the list of icons to the upper-right (i.e., cross with arrow heads) is a hyperlink to activate the overview window. The other icons are the regular Swiki actions, namely, show, edit, attachments, versions, top, changes, search, and help. After each hyperlink (e.g., Contribute forces), there is an image label indicating the type of the target page (AR means artifact, AC means activity, TO means tool, and PA means participant).

Extensions to the Swiki server can be programmed in a tool provided for this purpose. Extensions are stored in text files that are read by the server on startup. This approach makes extending the Swiki with ad-hoc functionality possible. This programming facility was used to implement the clone operation, the extensions to the calculation of incoming references, displaying anchors with an icon indicating the type of the target node, and the automatic creation of entries in the event logs.

Squeak Smalltalk provides a graph layout package based on its Morphic [1] framework [29]. The graphical overview was implemented as a Squeak applet (Squeaklet) running in an independent browser window. The Swiki server was extended (also using the programming facilities of Swiki) to dynamically generate the web-page containing the applet. An HTTP request is sent from the applet to the server to retrieve an XML representation of the scaffolding. Figure 5.3 shows the graphical overview. A button labeled refresh, located in a movable inner window, causes a reload of the scaffolding data from the server. The graph layout framework used to create the 2D image provides functionality for automatic layout. A check box located in the movable window activates or deactivates the automatic layout. Graph nodes can be moved with the mouse pointer. The prototype shows each node as a circle. Different types of elements are drawn using different icons inside circles. The label of activities is extended with an uppercase letter between parentheses indicating the state of the activity.

5.3 Breakdown Landscape

The prototype of the Breakdown Landscape tool specified in sections 4.3.7, 4.4.8, 4.5.3, 4.6.4, and 4.7.6 was incrementally implemented in Visualworks Smalltalk. Smalltalk enables fast prototyping and graphical construction of user interfaces. Moreover, applications can be run stand-alone or within the Visualworks web-browser plug-in. The object model was implemented as specified by the UML class diagram presented in Figure 4.72 on page 140. XML marshalling and unmarshalling was done with the XMLObjectMarshallers Framework delivered with Visualworks.

Smalltalk provides a flexible set of collection querying methods. The methods collect, select, reject, detect, that are applicable to most types of collections, can be used to query the object model of the Breakdown Landscape. Visualworks implementation of closures [27] supports the addition of new queries at runtime using Smalltalk syntax. A query is specified as a two-argument block. The first argument is the landscape object. The second argument is a dictionary of parameter-name/parameter-value pairs. Advanced users have access to a query editor for defining new queries. Figure 5.4

[1] Morphic is a direct-manipulation User Interface (UI) construction kit. It replaces the original Model View Controller graphics toolkit of Smalltalk-80.

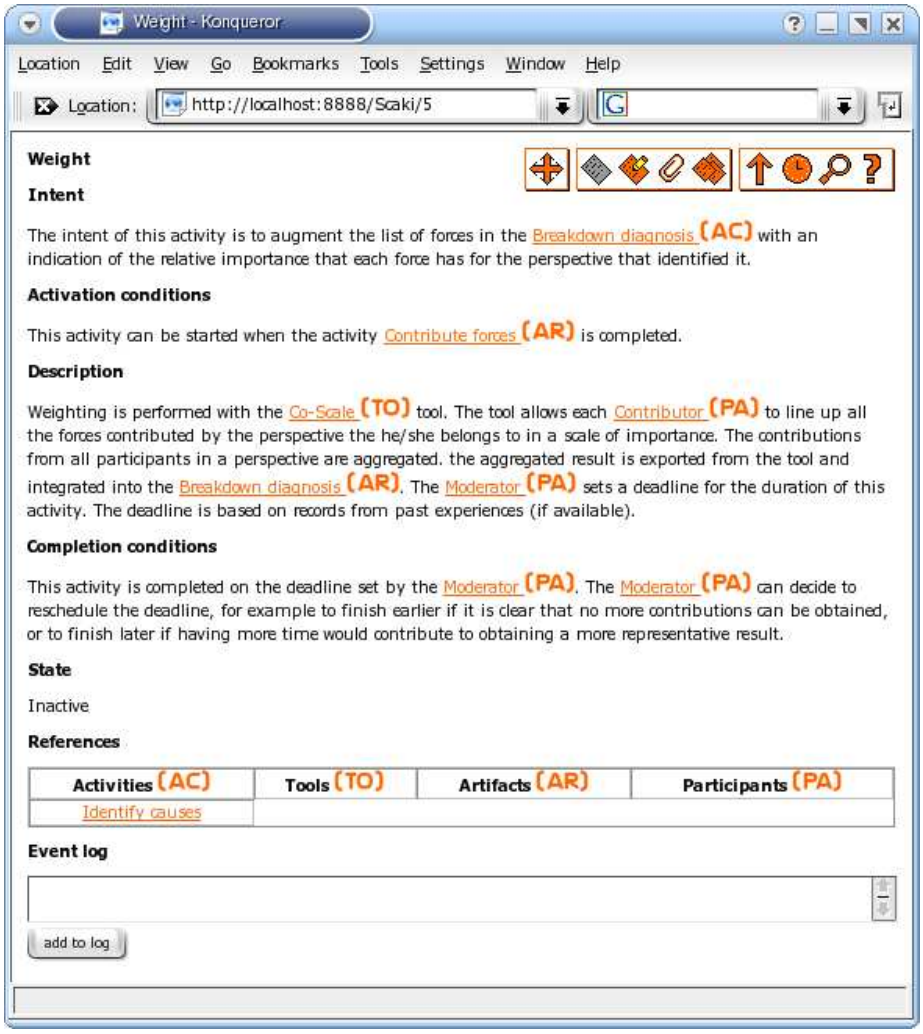

Figure 5.2: Example of an activity in show mode

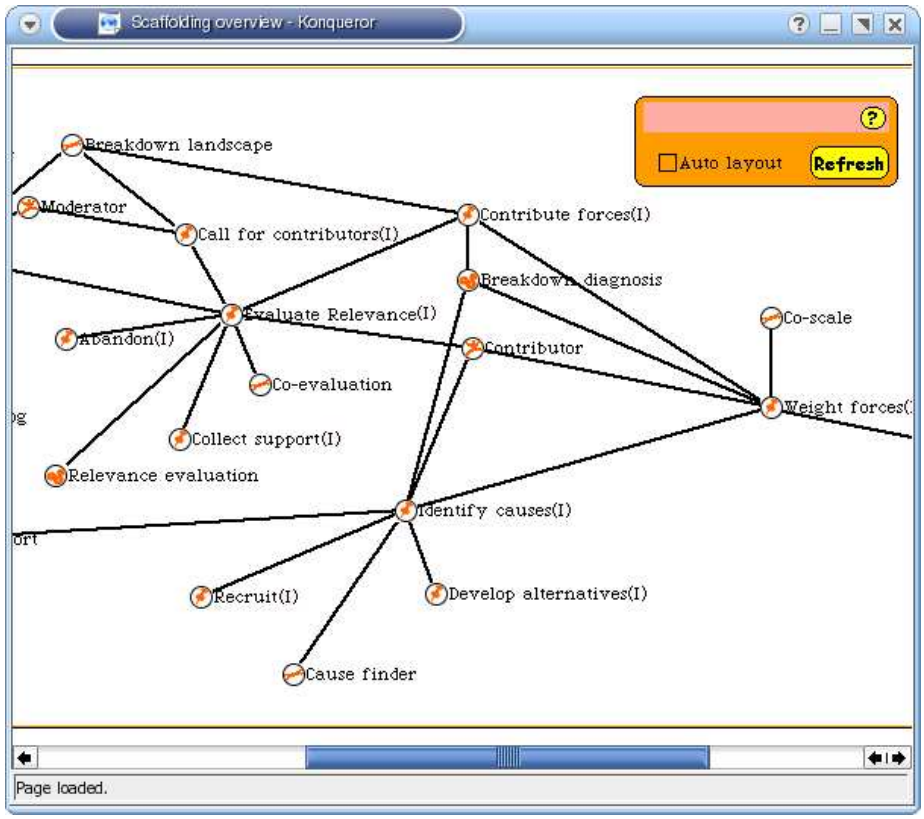

Figure 5.3: Scaffolding overview running as a Squeaklet in a web-browser window.

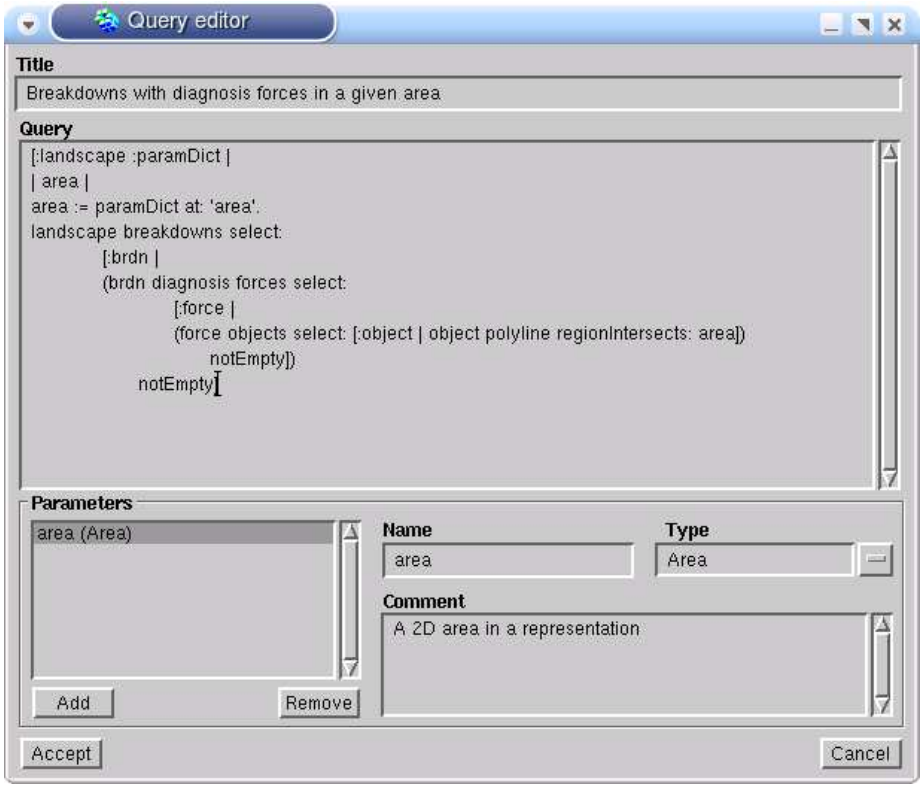

Figure 5.4: Breakdown Landscape - Query editor window

show the query editor window. There are four available types of parameters: String, Number, Objects, and Area.

To execute a query, the user must provide values for all required parameters in the query runner window (see Figure 5.5). There are two ways to assign values to parameters: immediately and from buffers. Users normally give immediate values for parameters of type String and Number. Immediate values are directly entered in the query runner window (e.g., the keywords parameter in Figure 5.5). Numbered buffers are used to hold selections of objects in representations, areas in representations, or results of queries. If a query requires an area or object parameter, the value is passed through one of the available buffers (in the figure, the parameter `problemArea` takes its value from buffer two).

The Breakdown Landscape server was also implemented in Visualworks Smalltalk, using a CVS back-end for the repository. The CVSRepositoryImplementation (see class diagram in Figure 4.54 on page 118) was built as a proxy and wrapper for the external CVS executable. As both the server and the client are implemented in Smalltalk, communication among them is done by means of serialized objects using the Visualworks Binary Object Storage Framework (BOSS). The client connects via TCP/IP to a known port of the server. Requests are encapsulated as serialized objects (Command pattern [26]) and sent to the server. If the client requests the Breakdown Landscape data, the server loads the XML from the repository, parses it to Smalltalk objects, and sends the serialized breakdown landscape back to the client.

The server maintains the list of all users that requested notification on changes to

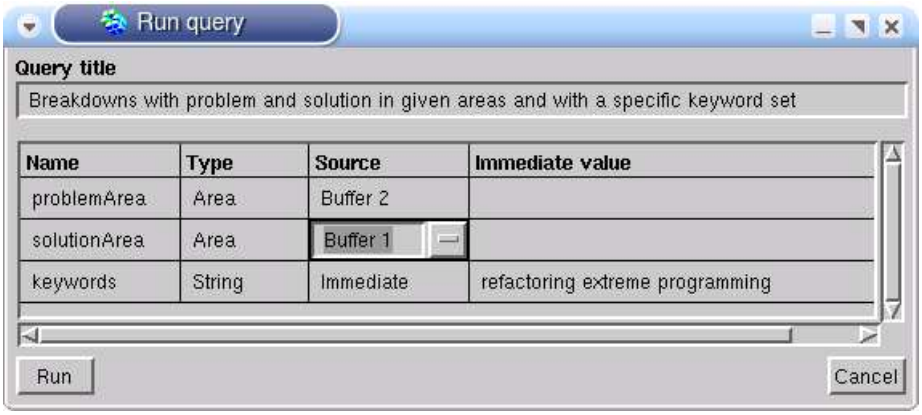

Figure 5.5: Breakdown Landscape - Query runner window

the landscape. After serving a commit request (delegating the corresponding commit command to the external CVS executable), the server iterates the collection of interested users and sends e-mail notifications (via SMTP). This approach, although fast to implement, has the drawback of not being aware of changes introduced in the XML files in the CVS repository by other applications.

5.4 Lightweight Framework for Loosely Coupled Deliberation Tool

Visualworks Smalltalk was also chosen for the implementation of the prototype of the generic architecture for loosely coupled deliberation tools (see Section 4.4.7) and the prototype tools. The architecture was implemented as an object-oriented framework. To build an application, developers must subclass GenericDeliberationToolClient and GenericDeliberationToolBroker. An inherited method `submit` on the client class encapsulates connecting to the tool broker and delivering a contribution as a serialized object. Similarly, the inherited method `getResult` encapsulated connecting to the server and retrieving the aggregated result as a serialized object. Communication between the client and the broker occurs as sending and receiving serialized objects through TCP/IP.

On the server, a single Visualworks virtual machine and image can serve several tools. Each instance of a subclass of GenericDeliberationToolClient (e.g., CoEvaluationToolClient) is configured to listen on a different port. The prototype implementation of the framework cannot share brokers among different sessions (i.e., different instances of the same tool cannot serve different sessions). Each instance (session) of a tool must run on a different port. Clients are delivered via the Visualworks plug-in architecture (currently only available on Windows). Information about session id and the Internet address and port where the broker is listening is passed in an argument to the HTML EMBED tag.

5.4.1 Breakdown In-box

The prototype of the Breakdown In-box specified in Section 4.3.13 was implemented as a special case of a deliberation tool. The In-box tool requires information stored as part of the scaffolding server, namely, the list of all breakdowns and their state.

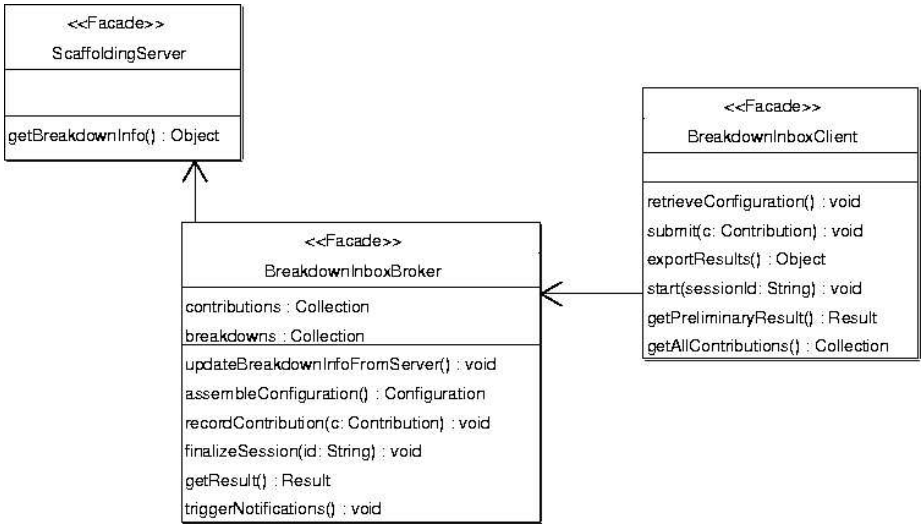

Figure 5.6: Breakdown in-box - a special case of a deliberation tool

Additionally, the in-box maintains a prioritized list of breakdowns that wait for handling. This list is constructed from the contributions of all team members. Figure 5.6 provides an overview of the implemented architecture. When users start the client, it connects to the server to retrieve the configuration (method `retrieveConfiguration`). In response, the BreakdownInboxBroker connects to the scaffolding server to retrieve the list of breakdowns and their state. The scaffolding server was extended to listen on a TCP/IP port for this requests. Communication takes place via serialized objects [2]. Once the BreakdownInboxBroker received the list of breakdowns from the scaffolding server, it builds (method `assembleConfiguration`) the object model required by the client (breakdowns, plus contributions, plus aggregated result) and returns it to the client.

5.4.2 CoScale Tool

The CoScale tool, similar to the Breakdown In-box, requires data from other tools. The CoScale tool presents the list of forces documented for a breakdown. Team members contribute weights for these forces. In consequence, the CoScale tool needs data from the Breakdown Landscape and information about team members in the organization and the shared perspectives to which they belong. To obtain the required initial data, the CoScaleBroker connects over TCP/IP to the scaffolding server (to the same TCP/IP port used by the BreakdownInboxBroker) and retrieves serialized objects for team members and shared perspectives. To obtain the list of forces for the breakdown, the CoScaleBroker connects to the Breakdown Landscape server over TCP/IP (in the same way that the Breakdown Landscape client did) to retrieve the breakdown landscape.

[2]The scaffolding server, implemented in Squeak Smalltalk is not compatible with the Visualworks serialization mechanism used for the Breakdown Landscape (BOSS). Therefore, serialization was done with XML

Chapter 6

Usage Experience

The method and tools described in the approach are the result of an iterative exploration. The supporting effect of many of the principles (e.g., rank breakdowns, state forces first) and technology (e.g., cooperative hypermedia, lightweight architecture for deliberation tools) was evaluated as the approach was taking shape. Section 6.1 describes early experiences that serve as indicators of the usefulness of parts of the approach. Section 6.2 describes how the complete approach was applied to handle a work breakdown (breakdown on level A). Section 6.3 describes how the approach was applied to handle a tailoring breakdown (breakdown on level B). Section 6.4 provides a summary of observations.

Conclusions from usage experiences were collected in interviews with the team members who participated from the use cases. Interviewed team members were requested to comment, at least, on three aspects of the experiences: a) perceived differences between tailoring teamwork with the approach proposed here and tailoring teamwork as it was done previously; b) capabilities introduced by the approach which were not available before; c) problems introduced by the approach; d) sources of complexity in the approach.

6.1 Early Experiences

In 2002-2003, a group of team members of the Concert division of the Fraunhofer IPSI initiated a breakdown handling initiative (at that time it was seen as an improvement group) called M42. This group would meet periodically to discuss about teamwork problems and potential solutions. The key principles driving the work of the team were: identify problems, explicitly state expectations, identify key variables, evaluate alternatives, recommend course of action, plan assessment, be prepared to change again, and record decisions.

The Concert division is a multidisciplinary team of researchers. In Concert, psychologists, pedagogues, and computer scientists do research and development on CSCW and CSCL. Concert heavily relies on groupware systems to support the work of their team members. The M42 group established a mechanism to report teamwork breakdowns and to handle them. A cooperative hypermedia (a Swiki) was used to publish the principles of the group and to store breakdown reports and breakdown handling data.

Breakdown reporting was done with a breakdown report form (implemented as a Swiki form). The form had the following sections. Each section provided a set of questions and fill-in-the-blanks templates to help team members provide helpful information.

Time frame: Indicates when the breakdown was first observed and documents any deadline for its resolution.

Motivation: Describes a scenario that helped team members identify the problem.

Impact: Provides an initial assessment of the impact of the problem.

Participants: Identifies the participants (stakeholders) in the problem and their interests/requirements/expectations (i.e., forces)

Team members contributed to the problem reports (and later to solutions) when they had time. The group met periodically to discuss problems, prioritize them, and initiate handling. When handling of a problem started, an invitation to contribute was sent to all team members. The invitation included a link to the problem report. Team members who had observed the problem contributed from their perspective. The most commonly found perspectives were developer, manager, tester, evaluator (researcher in charge of evaluating the result of products with real users), and user. Each meeting was moderated following a predefined agenda.

A Swiki page acted as the meeting agenda and location for documenting results. After a time of proposing and discussing solution alternatives, the group issued a recommendation for action. The recommendation (created form a template form) had hyperlinks to the problems report and associated meetings. It had a detailed description of the action to be taken. The details clearly stated how each of the stakeholders were affected. The recommendation had additional implementation suggestions and a date when the results should be evaluated.

As the handling of the problems progressed, participation changed. Psychologists and pedagogues were very active during the definition phase, and less active during design and implementation of solutions (as this activity was more the arena of computer scientists).

In the time the M42 group was active, a handful of problems were reported and successfully solved. The lessons learned from this experience are: the problem report form, the meeting moderation form, and the recommendation form are valuable resources to focus collaboration and make it productive. An explicit mechanism to handle breakdown (e.g., how meetings are organized) minimizes the effort needed to collaborate in breakdown handling. Keeping a record of past actions and the rationale behind them helps avoiding that, while solving new problems, the solutions created for past problems stop working.

In the E-Qualification Framework project [2, 34] of Fraunhofer-Gesellschaft (2000-2002), a portal for the development and delivery of training material was built. An electronic process guide (EPG) was a critical component of the portal. The EPG delivers all instructions for activities in hypermedia format. Moreover, it also provides details for the selection of team members to fulfill roles and provides access to artifact templates. The XCHIPS [70] system was used to build an overview of the process, to maintain the state of running projects, and to provide access to artifacts. The lesson learned from the E-Qualification Framework project is that the tight integration of detailed instructions in the form of hypermedia with information about the state and artifacts of running projects successfully enables collaboration. The integration of the XCHIPS system and the EPG was inflexible (as these two systems were developed independently with sometimes diverging goals). This motivated the development of the scaffolding hypermedia server specified in Section 4.2.2.

In the ConcertStudeo project [73], a framework was developed to embed small interactions (e.g., voting, ranking, brainstorming) within learning material. Each interaction had a server side component and a client side component (running on lightweight devices). Interactions were integrated as applets or plug-ins in the learning material delivered as HTML. The architecture was used to build a set of simple interactions which were evaluated with the purpose of evaluating its potential. The ConcertStudeo project demonstrated the feasibility of building a generic architecture

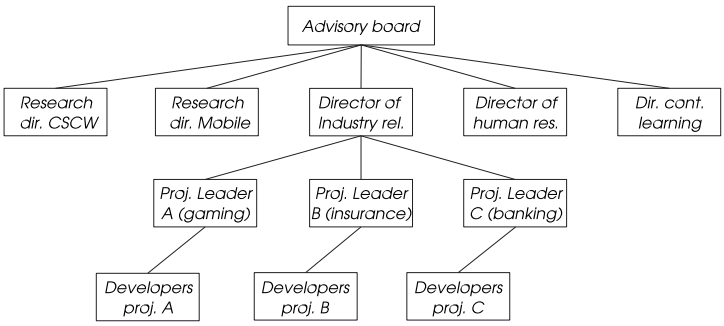

Figure 6.1: Organizational chart

for deliberation tools and of using such tools to support planned collaboration. The architecture presented in section 4.4.7 extends the resulting architecture built in the ConcertStudeo project to provide further tailoring possibilities and to better implement the setup-initiation-contribution-finalization model.

6.2 A Communication Breakdown: Counterproductive Pride

LIFIA is a research and development lab in the National University of La Plata, in Argentina. The lab is organized in two departments, research and technology transfer. The later aims at transferring research results to industry. The lab has approximately 130 team members. Most researchers have offices at the University campus (two different buildings). Technology transfer members are distributed among several locations (three locations in La Plata City, two locations in Buenos Aires city, one location in Portugal, and one location in Chile). Most transference members started as researchers or research assistants. Knowledge is LIFIA's key asset.

In September 2004, the lab's directors and the human resources manager expressed interest in applying the breakdown handling approach to a peer learning (peer help) breakdown identified by several team members. Previous attempts to solve the breakdown delivered apparently promising solutions that nonetheless failed. This section reports on the experience of handling the breakdown.

To handle the breakdown, the prototype of the scaffolding server with the method description was deployed on one of the labs server. The shared name space was initialized with the perspectives that were found critical for the resolution of the breakdown, namely developers, project managers, lab directors, continuous training managers, and human resources. The breakdown involved team members from two projects in two different locations (this section refers to them as Project A and Project B). To keep the experience controllable, only project managers and developers from these two projects were initially given team member pages in the shared name-space. In case of need, other projects would join.

The Breakdown Landscape was initialized with representations of an organizational chart with enough detail to serve as the focus of the present breakdown (Figure 6.1), and with a description of the most common way to handle technical difficulties (Figure 6.2). This way of handling difficulties was not formulated as an explicit practice but implicitly embedded in the organization's culture.

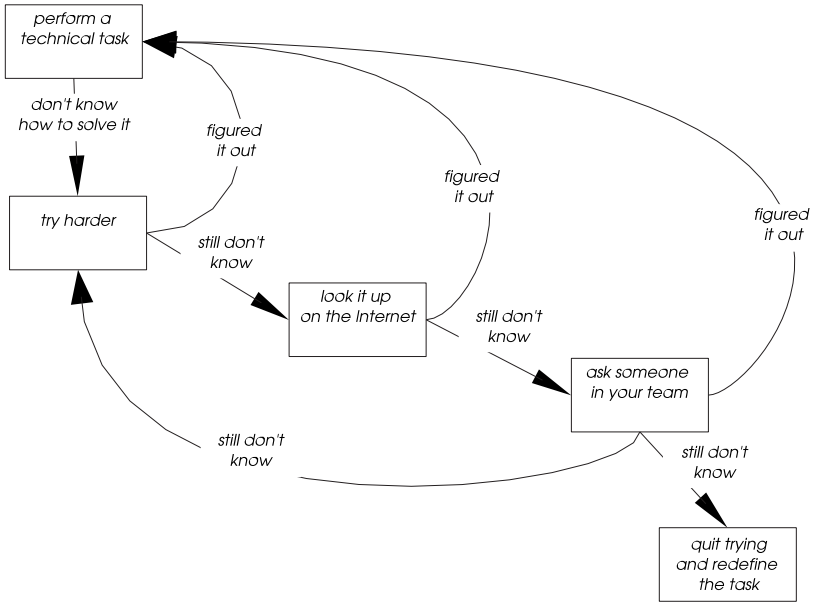

Figure 6.2: Common way to handle technical difficulties

6.2.1 Triggering

The team members that initiated the discussion about the problem (three developers, two from project A and one from project B) participated as reporters. The author of this thesis acted as the moderator. Moreover, a number of e-mails that team members exchanged in the past regarding the problem were scrutinized to find further information to include in the report.

The breakdown was labeled "Counterproductive Pride". The context section of the breakdown report is reproduced in the next paragraph. It uses the \ref{id} notation to introduce references to elements in the diagrams in Figures 6.1 and Figure 6.2.

> *We developers (\ref{Developers-Proj-A}, \ref{Developers-Proj-B}, \ref{Developers-Proj-C}) are often confronted with technical challenges (\ref{Perform-Technical-Task}) we still don't have the answer for. The latest example was when they had to implement a persistence back-end for our Object-Oriented C++ insurance application. In such a situation we first try to solve it without help (\ref{Try-harder}). You can call this pride or simply trying to avoid bothering others. If that does not work, we search the Internet (\ref{Look-it-up}). If that does not work, we ask our team mates (\ref{Ask-team-mate}). If nobody has the answer, we give it another try and think harder. Working this way is not bad per se. However, it turns out to be frustrating (for you, your boss, and your colleagues) if you learn after a while that another team in the lab had a similar problem and had already found a solution. That is in fact what happened when we learned, after a week of worthless effort, that the team working on Project A (\ref{Developers-Proj-A}) had already implemented a persistence back-end.*

The symptoms section of the breakdown diagnosis states:

- *The clearest symptom of this problem is realizing that you have spent time looking*

for an answer that someone else already had.

- *Lone rangers (e.g., lonely developers that like working without asking anyone for help) are a good indicator for an early diagnosis of this problem.*

Reporters initially had difficulties in referencing elements in the representations. The chosen representations and the text of the report were not compatible. With the help of the moderator, they tuned the report to focus on the available representations. Moreover, they suggested the introduction of "developers" as a part of the organizational diagrams that was not present in the initial version.

There were no other breakdowns in sight (and no plans to extends the experiment). Therefore, the moderator decided to skip weighting. The definition phase started.

6.2.2 Definition

As this is the first breakdown handled with the approach, no forces had been documented in the Breakdown Landscape as yet. Therefore, the Breakdown Landscape was of no help to find perspectives that contributed forces involving any of the objects referenced in the report. The moderator added a hyperlink from the "Contributor" node to the team member node of each team members sharing the "Developers" perspective. All developers agreed to contribute. Moreover, the project managers from both projects asked to be invited. The moderator added links from the "Contributor" node to their team member pages.

During the definition phase, team members simultaneously estimated the effort of handling the breakdown and assessed the cost (consequences) of not handling the breakdown. Based on this information, they could decide whether or not to handle the breakdown. There were no records in the system of past breakdown handling efforts. However, the moderator asked those in the lab who were involved in previous attempts to solve the problem for an estimation of the effort spent. Based on their estimations, the moderator set the effort to 8 work days.

Developers had already decided to handle the breakdown and voted in favor of handling it. Managers also had interest in handling the breakdown. As there was no need to provide arguments to convince a simple majority of voters to vote in favor of handling the breakdown, there were no contributions to the relevance evaluation. The decision to handle the breakdown was unanimous. The diagnosis phase started.

In later discussion about the diagnosis phase, the managers realized that developers outnumbered them and that it would have been futile to vote against handling (assuming they were against handling). Deciding on breakdown handling (see Section 4.4.12) is done through simple majority voting. They considered this a problem and reported it as a level B breakdown called "Unfair Voting" (see Section 6.3).

This breakdown was an exceptional case. Firstly, because there was no record from past breakdowns in the Breakdown Landscape. Secondly, because team members were clear about their interest in handling the breakdown using the approach. The flexibility of the hypermedia scaffolding approach allowed the moderator to slightly deviate from the suggested method (e.g., skip weighting) to proceed much faster.

6.2.3 Diagnosis

Developers and managers contributed forces. Developers found it easier than managers because developers had experience writing design patterns. Several iterations were required for the first group of forces to be usefully stated. Then this initial set served as a parameter and example for further forces to be documented.

First, developers and managers documented forces. Although the Breakdown Landscape had no data to support identifying other important stakeholders, the reported forces suggested that top management perspective, and human resources perspective could provide important insight about the problem. These perspectives were

added to the scaffolding server and their members invited. These newly invited members also contributed forces.

After a deadline of three days set by the moderator to contribute forces, team members proceeded to weight the forces contributed by their perspectives. The following list corresponds to the content of the second section of the breakdown diagnosis artifact (weighted list of forces). The **(F)** in front of a force indicates the force was documented as a fact. The **(A)** indicates that the force was documented as an argument. The list that follows each force contains the teamwork objects objects involved in the force. Forces marked with a star (*) were identified as causes during the identify causes activities. Causes are listed in a section of the diagnosis artifact. For space reasons and to avoid redundancy, they are marked a in weighted list of forces.

Forces contributed by developers

Must forces

(*) **(F)** In everyday work, it is common to find technical problems that require skills we haven't yet acquired. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C.

(A) Asking a colleague reduces the effort of searching for a solution, understanding it, and evaluating its effectiveness (trust). Objects: Ask-team-mate.

(*) **(F)** When you have nobody to ask and need to learn something yourself, it is hard to find the right material to learn from. Objects: Try-harder.

(F) There are common questions that come up over and over again. Specially for newcomers. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C.

(F) When you are on fire[1], you need help immediately. There is no time to wait for others to have time to help you. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C; Ask-team-mate.

Should forces

(F) There may be nobody in our project team to ask for help. Objects: Ask-team-mate.

(F) We do not have time for reading or learning about other stuff than what we use. We do not have time to stay up to date with technology. Objects: Try-harder.

(A) When you are on fire you have no time for helping others. Unless, you recognize that helping has higher priority (e.g., relates to a high priority problem). Objects: Ask-team-mate.

Can forces

(F) Even if we find the one who knows, we still need to ask for help. We can look as dummies. Our inquiry can bother our colleagues. Objects: Ask-team-mate.

Forces contributed by the human resources perspective

Must forces

(*) **(F)** There is a small core of skills that all workers have. Besides that, each worker has a particular area of expertise and interest beyond his/her formation in the team. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C.

(F) It is hard to find workers qualified beyond the core knowledge. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C.

(F) Workers are commonly young, with no previous experience in development projects. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C.

[1] Developers used this expression to describe days when they are overloaded with work that must be resolved immediately, e.g., when the system being developed goes into production.

Forces contributed by the top management perspective

Must forces

(F) Skill acquisition has a cost that translates to the price of the service/product. Objects: Try-harder; Ask-team-mate; Look-it-up.

(F) Duplicated skill acquisition effort represents an unnecessary cost. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C; Try-harder; Ask-team-mate; Look-it-up.

Forces contributed by the project management perspective

Must forces

(A) Hiring only core-qualified workers increases the risk that a problem cannot be solved due to lack of specific (non core) skills. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C; Perform-technical-task.

(A) Hiring multi-talent (clearly overqualified) workers is hard and costly. Objects: Developers-Proj-A; Developers-Proj-B; Developers-Proj-C; Perform-technical-task.

(*) **(S)** A system of forces involving the first two forces in this list.

Most contributors were new to the idea of using forces to diagnose problems. Even those who had pattern writing experience were not sure about classifying forces as facts and arguments, and about identifying causes. However, as participants practiced, they progressed and felt more comfortable with these concepts.

6.2.4 Design

The team decided to collaborate to design one solution. A branch of the Breakdown Landscape was created in case they needed to add forces or modify representations. The moderator also took the role of editor. He suggested that they invite experts from the lab's CSCW research area. A shared perspective CSCW experts was created. Two researchers agreed to participate.

While designing, the following design forces were documented.

Design forces

(F) After asking a colleague, one becomes knowledgeable in the area. Maybe not an expert, but knowledgeable enough to be able to answer the question if it comes up again. Objects: Ask-team-mate.

(F) It is hard to find the one who knows. Objects: Ask-team-mate.

(A) If a question is directed to a specific person, this person will usually react either with help or indicating that he/she cannot help with the issue. As a result, the requester knows whether to turn to someone else, to wait for promised help, or to quit hope. Objects: Ask-team-mate.

(A) If a question is directed to a group, it is possible that those who can help do so, or wait to see if someone else (with more spare time) helps first. As a result, the requester may wait indefinitely for help or may quit waiting even if there was someone who had the answer. Objects: Ask-team-mate.

(A) We have access to professionals with state of the art knowledge (possibly without practical experience). Objects: Research-director-CSCW, Research-director-CSCW.

(A) Researchers and academics have time and will to explore, learn and teach. Objects: Research-director-CSCW, Research-director-CSCW.

The proposed solution started as "First Ask the Community then Learn-it-yourself". It consisted in trying to change how people went around technical problems. Instead of trying to solve a complicated technical problem, developers would first post a question to an e-mail list (internal to the organization). This should be seen as the "default" attitude and should not be criticized. Those who know the answer should feel compelled to answer or to say they should be able to help soon. Therefore, the designed solution had a process factor (the new way of approaching technical challenges) and a tool factor (implementing the distribution list). Follow-up evaluation was scheduled for three months after the solution was implemented.

6.2.5 Treatment

The implementation of the tool (mainly technical) part of the design was straight forward. Additional implementation forces were found that had to do with going through firewalls and reducing spam. For the process part (mainly sociological), it was necessary to conduct a workshop with the developers. The workshop consisted of several team building exercises that aimed at helping team members understand, discuss, and eventually adopt the new way of working.

6.2.6 Evaluation

When the breakdown described in the previous section was handled, follow-up evaluation was done differently from the way specified in Section 4.8. Team members would conduct a collaborative testing of forces. Each one of them would subjectively indicate the state (resolved or unresolved) of each force in the breakdown diagnosis document. Three months after the solution was implemented, team members tried to do the testing. The results were not consistent. It was not clear how forces should be evaluated. After several attempts to make a sensible evaluation, the moderator decided to report it as a level B breakdown titled "Nonsense evaluation" (see Section 6.3). Moreover, the moderator proposed to ask all those contributors to give their opinion regarding the effect of the solution. There had been no recurrences of the breakdown in the last three months. However, it was not clear that the solution effectively changes the culture that caused the breakdown. They decided to re-schedule evaluation for a later date, thus giving more time for the solution to work, waiting for the level B breakdown to be handled, and to deliver a better way to do evaluation.

6.3 Handling Level B Breakdowns

When the "Nonsense evaluation" level B breakdown was reported, the "Unfair Voting" level B breakdown was still unhandled. With both breakdowns still in the triggering phase, team members needed to weight them according to frequency, severity and focus (see Section 4.3.13), and to decide which one should pass to the definition phase. The weight of breakdown "Unfair Voting" had "Could recur" for frequency, "Conditions" for severity, and "Critical path activity" for focus. The weight of breakdown "Nonsense evaluation" had "Seldom" for frequency, "Introduces risk" for severity, and "Critical path activity" for focus. The top priority breakdown calculation showed breakdown "Nonsense evaluation" as the only top priority breakdown, therefore it passed to the definition phase.

As a result of handling the "Nonsense evaluation" breakdown, a solution was proposed and implemented that suggested that the form to evaluate each breakdown should be determined when the solution is designed. To implement the solution, the specification of the develop alternatives activity (see Section 4.6.2) and the specification of the evaluate activity (see Section 4.8.2) were re-written, in the master scaffolding, to the form they currently have in this document. New cases (i.e., new instances

of the scaffolding for later breakdowns) used new specification. The instance of the scaffolding that corresponded to the case of the "Counterproductive Pride" breakdown was created from the old master of the scaffolding. Therefore, it was necessary to manually update it to reflect the change in the process. Moreover, the develop alternative activity of the design phase was shortly reactivated to complete the evaluation section of the design alternative artifact. To evaluate the solution, team members decided on a week where prepared questions (for which there were team members who had the answers) would be posted to see the reaction of team members. The expected results was that they, at least, answered that they would be able to help as soon as they had a spare minute.

6.4 Observations

Usage experience, although limited, led to the following observations of key contributions and problems that require further work. Explicitly documenting forces contributed to higher acceptance of proposed solutions. There were cases where team members agreed to adopt a tool that they were rejecting in the past because they now could see that the arguments for its adoption were convincing. Before the introduction of the process of collaborative tailoring, team members would propose new groupware tools simply because they were appealing. Similarly, they would oppose other tools. Explicitly documenting forces made these attitudes hard to sustain.

The introduction of the approach changed the distribution of participation and effort. In the past, team member participation and effort spent was aimed mainly at the design of solutions. Moreover, design discussions were often complex and unproductive. As a result of applying the approach, effort concentrated in documenting forces. Team members tried hard to provide all arguments they could think of, sometimes to contribute to a good decision, sometimes to bias the potential solutions to their preference. Participation declined during design to the minimum required (one team member for each invited perspective). Only in forced situations team members developed parallel solutions (because it would contribute to a better solutions). Participation (motivation) increased during the follow-up evaluation phase. However, this can be explained by an interest in evaluating the results of tailoring with a method instead of random tailoring as it was done in the past.

The introduction of the approach changed the tailoring culture of the groups that evaluated it. In the past, tailoring was done either as the result of major reorganizations in response to external changes and major problems, or as the result of random proposals for change (e.g., adopt this modern agile methodology). After having used the approach proposed in this thesis, even though tools and support were no longer available, team members continued documenting work breakdowns, prioritizing problems, explicitly documenting interests and requirements, and maintaining the list of solutions whose impact needed to be confirmed.

The breakdown handling approach itself (i.e., the process, the tools, the artifacts, and communication) generated much interest and discussion. Many changes and improvements were suggested soon after introduction. Due to a lack of time and of good reasons (e.g., the lack of usage experience directly translated to poor diagnosis) only a few of them were handled as breakdowns on the level B. The introduction of the breakdown handling approach brought to the attention of the involved teams the need of an explicitly documented, well thought change policy (in this case, based on the reaction to breakdowns) and that such a policy must be itself changeable. In this way, the evaluation of the approach served as a means to introduce the concept of organizational improvement as proposed by Engelbart [23].

To be able to talk about breakdown handling, the approach presented here serializes the work to be done in a sequential process. This is done by many disciplines

(e.g., software engineering) to help tackle complexity. People are used to work following such processes. However, when it comes to solving work breakdowns, they tend to act chaotically unless the process is driven by an expert, for example, by a consultant, or a computer system with built-in rules and tools that support the processes. Separating diagnosis from design did not feel natural for them. In most cases, they were tempted to propose solutions even though the problems had not been clearly analyzed.

6.4.1 The Groupware

The groupware (i.e., the Scaffolding Server, deliberation tools, and shared artifacts) effectively supported collaborative breakdown handling among distributed team members. In the past, only team members who had the possibility to attend the tailoring meetings participated in tailoring discussions. These meetings were usually organized and dominated by small groups of team members (e.g., the infrastructure group). Deliberation tools allowed distributed team members to contribute on-line from their offices whenever they had time, which was vital to involve several shared perspectives.

The artifacts (e.g., the breakdown diagnosis artifact) helped to make collaboration more productive. Team members concentrated work on completing the required parts of the document. The structure of the artifacts and the meta-information that the scaffolding provides for each artifact helped team members stay focused.

Use case participants indicated that the Breakdown In-box tool, the Co-Evaluation Tool, and the Co-Scale tool help to get more out of the energy invested in tailoring. They provide an overview of the problems the organization faces, and they help team members agree on which problems to tackle depending on their importance and potential cost.

Experiences with the use of the breakdown handling method and related artifacts started before most tool prototypes were operational. In early experiments, the operation of tools was emulated by a combination of existing tools, such as e-mail to submit contributions and Excel sheets to automate aggregation and publication results. The moderator acted as a coordinator, performed all required computation (e.g., aggregation of contributions), and published results (e.g., in a web page). Prototypes were developed based on how critical they were to support the method. The scaffolding server and the Breakdown Landscape were the first tools to be prototyped. The scaffolding server was critical as it provides instructions, records the status of the process, and provides access to artifacts. The Breakdown Landscape is required in most phases of breakdown handling. It reflects the team's understanding about teamwork, perspectives, and breakdowns forces.

The most promising tool, the Breakdown Landscape, also proved to be the most difficult to use. Participants found it difficult to state useful forces. It only worked smoothly when the team members involved had pattern writing experience. In other, cases the author of this thesis had to help restate forces so as to build a useful landscape. Producing useful representations of teamwork (i.e., diagrams) requires training and practice. Organizational charts and process descriptions were the simplest to create because they are based on well known notations. Diagrams for specific aspects of teamwork not commonly diagrammed in literature (e.g., how communication tools and their features were used to support collaboration in the organization) required iterative improvement until they became useful for anchoring forces (i.e., relating forces to teamwork objects).

Data captured in the Breakdown Landscape during the use cases was too limited to draw final conclusions about its impact. However, interviewed team members shared the opinion that the ability to identify elements of teamwork that are key for different breakdowns is a means to reduce the chance that solutions to new breakdowns break solutions of past breakdowns. The Breakdown Landscape effectively helped identify team members whose participation was necessary. Managers indicated that identifying

areas of teamwork where many breakdowns meet has strategic value, for example, to identify the need of organizational changes that would be too costly and take too long if only solved as the result of incremental change through breakdown handling.

Chapter 7

Conclusions

This thesis tackles the problem of the lack of computer support for distributed team members that need to perform tailoring in the context of teamwork. The challenge of tailoring in the context of teamwork is to understand and support the needs of the group members, from the moment they encounter a breakdown during work until they have enacted the changes they deem necessary. Any contribution in this area will act as a multiplier for the value of existing tailoring capabilities.

The approach followed in this thesis is based on the premise of participation as a means to achieve acceptance of change. It consists of a method for collaborative breakdown handling, a selection of specific groupware tools to be used for the deliberation activities defined by the method, and guidance in the form of scaffoldings for the application of the method. The method, the specific tools that support the method, and the scaffolding for the application of the method, have been designed with a balance of flexibility and structure. Moreover, the method, the tools, and the scaffolding can be tailored, individually and and collaboratively. The proposed support for collaborative tailoring of teamwork is delivered as a stand-alone groupware system for collaborative tailoring. The system can be deployed along existing groupware systems, thus extending them with support for collaborative tailoring. However, the extent to which they can be tailored dependens on the tailoring facilities that they provide.

7.1 Summary of Contributions

This thesis asks the question of how to design our systems to adapt to change, not automatically, but rather through the team's negotiated tailoring of their work processes, communication, resource management, and production of work results? To answer this question, the thesis delivers the following original contributions:

- It proposes collaborative breakdown handling as conceptual view of collaborative tailoring of teamwork, centered on the existence of multiple perspectives and the occurrence of breakdowns (see Chapter 2).
- It presents an approach to support collaborative tailoring of teamwork that comprises an on-line delivered collaborative breakdown handling scaffolding, a system to support the coordinated application of the scaffolding, templates and a repository for the artifacts that are created during breakdown handling, and a selection of groupware tools to support collaboration, communication, and negotiation for breakdown handling (see Chapter 4).
- A prototype of the scaffolding system and tools provides a proof of concept of the approach. In addition, the system supports collaborative tailoring of the team's tailoring practices and is itself tailorable (see Chapter 5).

- Results from two case studies suggest that the approach indeed facilitates collaborative breakdown handling (see Chapter 6).

7.2 Comparison to Related Work

Chapter 3 presented work in three related areas: tailorable groupware, understanding tailoring, and collaborative tailoring. The chapter pointed out the various limitations of past work. Tailoring is only covered as a technical issue (looking at the technical system) without considering the team processes that form the context of tailoring (the social system). There is no integrated approach (covering all aspects of teamwork) that describes teamwork for tailoring in terms of processes, tools, artifacts, and communication. Related work fails to recognize that tailoring is teamwork and as such it is subject to change and should itself be tailorable. Existing approaches providing groupware solutions for tailoring are limited to a particular system and support a limited set of the required capabilities (communication, collaboration, co-operation and coordination, and negotiation).

This thesis exceeds related work by approaching tailoring of teamwork as a social system with a model that explains tailoring as the result of collaborative breakdown handling. The requirements of communication, collaboration, co-operation and coordination, and negotiation observed in the social system are supported by the corresponding technical system. Such a system contributes to all aspects of teamwork, namely, processes, tools, artifacts and communication. The approach in this thesis is not limited to its application in a particular scenario or groupware system. The only requirement is that the target system/scenario provides tailoring hooks (that is, it can be tailored). Finally, the approach has been conceived to enable and support its own evolution as the result of its tailoring.

7.3 Outlook

Section 6.4 pointed out some of the limitations and problems found during use. These limitations and problems are the focus of future work.

The most promising tool, the Breakdown Landscape, also proved to be the most difficult to use. Participants found it difficult to state useful forces. The Breakdown Landscape stands on three pillars: (a) team members belonging to multiple perspectives, (b) documenting expectations/considerations regarding problems and their resolution, (c) and a common reference model to anchor expectations/considerations and to allow querying and inference. Forces are one approach to (b). Although they seemed simple in the beginning, their lack of formalism may have been the cause of the difficulty in using forces. Further work can benefit from exploring alternative approaches. Promising alternatives can be found in the area of software engineering, specifically from goal-oriented requirements elicitation methods (e.g., [19]). Representations based on diagramming conventions are one approach to (c). Although team members can use existing representations, it is difficult to create useful ones (that allow meaningful anchoring). Future work can explore innovative forms of knowledge modeling. Ontologies may be a good alternative, as there is research in building tools that simplify the creation of ontologies and that support team members in collaboratively creating them (for example, Protégé [56]).

To be able to talk about breakdown handling, the approach presented here serializes the work to be done in a sequential process. However, when it comes to solving work breakdowns, team members tend to act chaotically (unless the process is driven by an expert, for example, by a consultant). In software engineering, agile methodologies have recently emerged as an alternative to traditional engineering processes. Agile methodologies (e.g., extreme programming [12]) propose other mechanisms to

handle the complexity of software development than sequentializing work. It is possible that something analogous to agile methodologies can be conceived for collaborative breakdown handling. Seeking such a solution is another direction for further work.

To reduce complexity, breakdown handling was limited to handling one breakdown after the other. However, breakdowns may occur at the same time and it may be necessary (or desired) to handle them in parallel. This presents new challenges for further work, for example, how to identify and manage co-dependencies among breakdowns, how to consistently extend the Breakdown Landscape to document the diagnostic forces from independent breakdowns, and how to consistently design, implement and evaluate solution alternatives.

The approach presented in this thesis focuses on changing the operational level of an organization (understand that teamwork is a part of an organization's operational level). Is the approach presented here (tailoring as the consequence of breakdown handling) also valid for other organizational levels such as the strategical level (where breakdowns may be related, for example, to the vision and goals)?

Effective groupware emerges with time, as the result of co-adaptation. The usage experience conducted in this thesis was limited in time and richness of cases. The real potential of this approach can be observed only after extensive, long time usage. Such long term studies need to be the focus of further work.

Bibliography

[1] *Code conventions for the java programming language*, http://java.sun.com/docs/codeconv/, Last accessed: 20/10/2003.

[2] *e-qualification framework*, http://www.e-qf.de/, Last accessed: Dec/14/2004.

[3] *Yahoo groups*, http://www.yahoogroups.com/, Last accessed: Dec/20/2004.

[4] Christopher Alexander, *The timeless way of building*, Oxford University Press, New York. USA, 1979.

[5] Christopher Alexander, Sara Ishikawa, and Murray Silverstein, *A pattern language : Towns, buildings, construction*, Oxford University Press, New York. USA, 1977.

[6] Teresa M. Amabile, *The social psychology of creativity*, Springer-Verlag, 1983.

[7] Ernesto Arias, Hal Eden, Gerhard Fischer, Andrew Gorman, and Eric Scharff, *Transcending the individual human mind: creating shared understanding through collaborative design*, ACM Trans. Comput.-Hum. Interact. **7** (2000), no. 1, 84–113.

[8] Ernesto Arias, Hal Eden, and Gerhard Fisher, *Enhancing communication, facilitating shared understanding, and creating better artifacts by integrating physical and computational media for design*, Proceedings of the conference on Designing interactive systems, ACM Press, 1997, pp. 1–12.

[9] Ernesto G. Arias, *Designing in a design community: insights and challenges*, Proceedings of the conference on Designing interactive systems, ACM Press, 1995, pp. 259–263.

[10] Ernesto G. Arias and Gerhard Fisher, *Boundary objects: Their role in articulating the task at hand and making information relevant to it*, International ICSC Symposium on Interactive & Collaborative Computing (ICC'2000) (University of Wollongong, Australia), ICSC Academic Press, 2000, pp. 567–574.

[11] Roswitha Bardohl, Karsten Ehrig, Claudia Ermel, Anilda Qemali, and Ingo Weinhold, *Specifying visual languages with genged*, APPLIGRAPH Workshop on Applied Graph Transformation (AGT'02), Satellite Event of ETAPS 2002 (Grenoble, France), 2002, pp. 71–82.

[12] Kent Beck, *Extreme programming explained: Embrace change*, Addison-Wesley, 1999.

[13] Richard Bentley and Paul Dourish, *Medium versus mechanism: Supporting collaboration through customisation*, Proc. Fourth European Conference on Computer-Supported Cooperative Work (ECSCW'95), Sept. 1995, pp. 133–148.

[14] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon, *Xquery 1.0: An xml query language*, http://www.w3.org/TR/xquery/, Last accessed: Nov/2003.

[15] Barry Boehm, Paul Grünbacher, and Robert O. Briggs, *Developing groupware for requirements negotiation: Lessons learned*, IEEE Software **18** (2001), no. 3, 46–55.

[16] John D. Bransford and Barry S. Stein, *The ideal problem solver*, 2nd ed., W H Freeman & Co, 1993.

[17] Robert O. Briggs and Paul Gruenbacher, *Easywinwin: Managing complexity in requirements negotiation with gss*, 35th Annual Hawaii International Conference on System Sciences (HICSS'02), vol. 1, 2002.

[18] Stephen Casner, *Building customized diagramming languages*, Visual Languages and Visual Programming (Shi-Kuo Chang, ed.), Plenum Press, New York, 1990, pp. 71–95.

[19] L. Chung, B. A. Nixon, E. Yu, , and J. Mylopoulos, *Non-functional requirements in software engineering.*, Kluwer Academic Publishers, 2000.

[20] Elizabeth F. Churchill, Jonathan Trevor, Sara Bly, Les Nelson, and Davor Cubranic, *Anchored conversations: chatting in the context of a document*, Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press, 2000, pp. 454–461.

[21] Jeff Conklin and Michael L. Begeman, *gibis: a hypertext tool for exploratory policy discussion*, Proceedings of the 1988 ACM conference on Computer-supported cooperative work, ACM Press, 1988, pp. 140–152.

[22] Jean M. Converse and Stanley Presser, *Survey questions : Handcrafting the standardized questionnaire (quantitative applications in the social sciences)*, Sage Publications, 1986.

[23] Douglas C. Engelbart, *Toward high-performance organizations: A strategic role for groupware*, Proceedings of the GroupWare '92 Conference, San Jose, CA, August 3-5, Morgan Kaufmann Publishers, 1992.

[24] J. W. Fellers and R. P. Bostrom, *Application of group support systems to promote creativity in information systems organizations*, Proceedings of the Twenty-Seventh Hawaii International Conference on Systems Sciences, 1993.

[25] Gerhard Fisher, *Symmetry of ignorance, social creativity, and meta-design*, International Journal Knowledge-Based Systems **13** (2000), no. 7, 527–537.

[26] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design patterns: Elements of reusable object-oriented software*, Addisson-Wesley, Toronto, Ontario. Canada, 1995.

[27] Adele Goldberg and David Robson, *Smalltalk-80: The language and its implementation.*, Addison-Wesley, 1983.

[28] Jonathan Grudin, *Why cscw applications fail: Problems in the design and evaluation of organizational interfaces*, CSCW '88, Proceedings of the conference on Computer-supported cooperative work (New York), ACM Press, 1988, pp. 85–93.

[29] Mark J. Guzdial and Kimberly M. Rose, *Squeak: Open personal computing and multimedia*, Prentice Hall, 2001.

[30] Austin Henderson and Morten Kyng, *Design at work: cooperative design of computer systems*, ch. There's no place like home: continuing design in use, pp. 219–240, Lawrence Erlbaum Associates, Inc., 1992.

[31] Thomas Hermann and Katharina Just-Hahn, *Organizational learning with flexible workflow management systems*, SIGOIS Bull. **17** (1996), no. 3, 54–57.

[32] Marike Hettinga, *Understanding evolutionary use of groupware*, Telematica Instituut Fundamental Research Series, Universal Press, Veenedaal, The Netherlands, Enschede, The Netherlands, 2002.

[33] David Hollingsworth, *The workflow reference model.*, Tech. Report TC00-1003, Workflow Management Coalition, Hampshire, UK, January 1995.

[34] Ines Ines Grützner, Jürgen Münch, Alejandro Fernandez, and Badie Garzaldeen, *Guided support for collaborative modelling, enactment and simulation of software development processes.*, Proceedings of ProSim'03, 2003.

[35] Knowledge Media Institute and The Open University, *Meet-o-matic: The world's simplest cross-platform meeting arranger*, http://www.meetomatic.com/, Last accessed: 08/01/2004.

[36] ISO, *Product data representation and exchange: Integrated generic resource. part 41: Fundamentals of product description and support*, second ed., 2000.

[37] P. Johnson-Lenz and T. Johnson-Lenz, *Post-mechanistic groupware primitives: rhythms, boundaries and containers*, (1991), 271–294.

[38] Peter Johnson-Lenz and Trudy Johnson-Lenz, *Computer-mediated communication systems*, ch. Consider the Groupware: Design and Group Process Impacts on Communication in the Electronic Medium, Academic Press, Newark, New Jersey, 1981.

[39] ———, *Rhythms, boundaries, and containers: Creative dynamics of asynchronous group life*, The International Journal of Man Machine Studies (1991), no. 34, 395–417.

[40] Katharina Just, *Step-by-step: a concept for describing co-operation within workflow management systems*, SIGOIS Bull. **17** (1996), no. 1, 15–17.

[41] Helge Kahler, *From taylorism to tailorability. suporting organizations with tailorable software and object orientation*, Symbiosis of Human and Artifact, proceedings of the HCI'95 (Amsterdam) (K. Ogawa Y. Anzai and H. Mori, eds.), Elsevier, 1995, pp. 995–1000.

[42] Anders I. Mørch. M. Kyng and L. Mathiassen (eds), *Computers and design in context*, ch. Three Levels of End-User Tailoring: Customization, Integration, and Extension, pp. 51–76, MIT Press, 1997.

[43] Jeff Langr, *Essential java style*, Prentice Hall, 1999.

[44] Nancy H. Leonard, Richard W. Scholl, and Laura L. Beauvais, *An empirical study of group cognitive style and strategic decision making.*, Presented at the Annual Meeting of the Academy of Management in August, 1998, 1998.

[45] Bo Leuf and Ward Cunningham, *The wiki way*, Addison-Wesley Professional, 2001.

[46] Wendy E. Mackay, *Patterns of sharing customizable software*, Proceedings of the conference on Computer-supported cooperative work (CSCW'90), ACM Press, October 1990, pp. 209–221.

[47] ———, *Users and customizable software: A co-adaptive phenomenon*, Ph.D. thesis, Massachusetts Instititute of Technology, 1990.

[48] Allan MacLean, Kathleen Carter, Lennart Lövstrand, and Thomas Moran, *User-tailorable systems: pressing the issues with buttons*, Conference proceedings on Empowering people: Human factors in computing system: special issue of the SIGCHI Bulletin (New York), ACM Press, 1990, pp. 175–182.

[49] Thomas W. Malone, Kenneth R. Grant, Kum-Yew Lai, Romana Rao, , and David A. Rosenblitt, *The information lens: An intelligent system for information sharing and coordination*, Technological Support for Work Group Collaboration (M. H. Olson, ed.), Lawrence Erlbaum Associates, 1989, pp. 65–88.

[50] Thomas W. Malone, Kum-Yew Lai, and Christopher Fry, *Experiments with oval: A radically tailorable tool for cooperative work*, Proceedings of the ACM CSCW'92, Toronto, Canada (New York) (J. Turner and R. Kraut, eds.), Conference on Computer Supported Cooperative Work, ACM Press, 1992, pp. 289–297.

[51] Gerard Meszaros and Jim Doble, *A pattern language for pattern writing*, vol. 3, ch. Patterns on Patterns, Addison Wesley Professional, 1998.

[52] Anders Mørch, Oliver Stiemerling, and Volker Wulf, *Tailorable groupware: issues, methods, and architectures (workshop report)*, The SIGCHI Bulletin **30** (1998), no. 2.

[53] Anders I. Mørch and Nikolay D. Mehandjiev, *Tailoring as collaboration: The mediating role of multiple representations and application units.*, Computer Supported Cooperative Work (CSCW). The Journal of Collaborative Computing. Kluwer Academic Publishers. Netherlands. **Volume 9, Issue 1** (2000), 75–100.

[54] Tony Mobily, Paul Weinstein, and Mark Wilcox, *Professional apache security*, Wrox Press, 2003.

[55] Bernard M. E. Moret, *Decision trees and diagrams*, ACM Comput. Surv. **14** (1982), no. 4, 593–623.

[56] N. F. Noy, R. W Fergerson, and M. A. Musen, *The knowledge model of protege-2000: Combining interoperability and flexibility*, Proceedings of EKAW 2000, 2000.

[57] Jonathan W. Palmer, *Supporting the virtual organization through information technology in a new venture: the retex experience*, Proceedings of the 1996 ACM SIGCPR/SIGMIS conference on Computer personnel research, ACM Press, 1996, pp. 223–233.

[58] George Polya, *How to solve it*, Princeton University Press, Princeton, NY, 1971.

[59] Dave Raggett, Arnaud Le Hors, and Ian Jacobs, *Html 4.01 specification*, http://www.w3.org/TR/REC-html40/, Last accessed: December/2003.

[60] Daniel Riesco, Edgardo Acosta, and German Montejano, *An extension to a uml activity graph from workflow*, (2003), 294–314.

[61] Horst Rittel and Melvin Webber, *Dilemmas in a general theory of planning*, Policy Sciences 4, Elsevier Scientific Publishing, Amsterdam, 1973, pp. 155–159.

[62] Robert W. Root, *Design of a multi-media vehicle for social browsing*, Proceedings of the 1988 ACM conference on Computer-supported cooperative work, ACM Press, 1988, pp. 25–38.

[63] Donald A. Schön, *The reflective practitioner. how professionals think in action*, Basic Books Inc., USA, 1983.

[64] Robert Slagter, *Dynamic groupware services - modular design of tailorable groupware*, Ph.D. thesis, University of Twente, 2004.

[65] *Swiki swiki*, http://minnow.cc.gatech.edu/swiki, Last accessed: March/2003.

[66] Gunnar Teege, *Implementing tailorability in groupware. wacc'99 workshop report*, SIGGROUP Bulletin **20** (1999), no. 2, 57–59.

[67] Wil van der Aalst and Kees van Hee, *Workflow management: Models, methods, and systems (cooperative information systems)*, 1st ed., MIT Press, 2002.

[68] W.M.P. van der Aalst., *Three good reasons for using a petri-net-based workflow management system.*, The Kluwer International Series in Engineering and Computer Science, vol. 428, ch. 10, pp. 161–182, Kluwer Academic Publishers, Boston, Massachusetts, 1998.

[69] Lev S. Vygotsky, *Thought and language*, MIT Press, Cambridge, MA, 1986.

[70] Weigang Wang and Jörg M. Haake, *Flexible coordination with cooperative hypermedia*, Proceedings of ACM Hypertext'98 (HT98), June 1998, pp. 245–255.

[71] ———, *Supporting workflow using the open hypermedia approach*, Proceedings of the First Workshop on Structural Computing (Peter J. Nrnberg, ed.), Aalborg University Esbjerg, 1999, Technical Report AUE-CS-99-04, pp. 12–17.

[72] Don Wells, *Extreme programming: A gentle introduction*, http://www.extremeprogramming.org/, Last accessed: Jan/01/2004.

[73] Martin Wessner, Peter Dawabi, and Alejandro Fernandez, *Supporting face-to-face learning with handheld devices.*, Proceedings of the International Conference on Computer Support for Collaborative Learning 2003. (B. Wasson, S. Ludvigsen, and U. Hoppe, eds.), Kluwer, 2003, pp. 487–491.

[74] Steve Whittaker, Susan E. Brennan, and Herbert H. Clark, *Co-ordinating activity: an analysis of computer supported cooperative work*, CHI'91: Conference on Human Factors and Computing Systems (New Orleans, USA) (Scott P. Robertson, Gary M. Olson, and Judith S. Olson, eds.), ACM Press, 1991, pp. 360–367.

[75] *Wikiwikiweb*, http://c2.com/cgi/wiki?WikiWikiWeb, Last accessed: March/15/2004.

[76] Terry Winograd and Fernando Flores, *Understanding computers and cognition*, Ablex, Norwood, 1986.

[77] M. Wooldridge and N. R. Jennings, *Formalizing the cooperative problem solving process*, 13th International Workshop on Distributed Artificial Intelligence (IWDAI-94) (Lake Quinhalt, WA, USA), 1994, pp. 403–417.

[78] Volker Wulf, *Negotiability: a metafunction to tailor access to data in groupware.*, Behaviour & Information Technology **14** (1995), no. 3, 143–151.

Appendix A

Selected Publications

1. Alejandro Fernandez, Badie Garzaldeen, Ines Grützner and Jürgen Münch. Guided Support for Collaborative Modelling, Enactment and Simulation of Software Development Processes. Journal on Software Process: Improvement and Practice. Willey Interscience. In Press.
2. Alejandro Fernandez. Scaki - The Scaffolding Wiki. In Proceedings of DEXA 2004. pp., 271-275. IEEE Presss. ISBN 0-7695-2195-9.
3. Martin Wessner, Peter Dawabi, and Alejandro Fernandez. Supporting Face-To-Face Learning With Handheld Devices. In B. Wasson, S. Ludvigsen, and U. Hoppe (Ed.): Designing for Change in Networked Learning Environments, Proceedings of the International Conference on Computer Support for Collaborative Learning 2003., pp. 487-491, Dordrecht, Kluwer, ISBN: 1402013833.
4. Alejandro Fernandez, Torsten Holmer, Jessica Rubart, and Till Schmmer. Three Groupware Patterns from the Activity Awareness Family. In Jutta Eckstein, Alan O'Callaghan and Christa Schwanninger (Ed.): Proceedings of the 7th European Conference on Pattern of Languages of Programs, 2002, p. 375-394, Konstanz, Universittsverlag Konstanz GmbH, 2003, ISBN 3-87940-784-3.
5. Alejandro Fernandez, Jörg M. Haake, and Adele Goldberg. Tailoring Group Work. In J. M. Haake and J. A. Pino (Ed.): Groupware: Design, Implementation, and Use. Proceedings of the 8th International Workshop , CRIWG, La Serena 2002, pp. 232-242, Springer, ISBN 3-540-44112-3.
6. Weigang Wang and Alejandro Fernandez. A Graphical User Interface Integrating Features from Different Hypertext Domains. Proceedings of the Third Structural Computing Workshop. Lecture Notes in Computer Science , Vol. 2266. Reich, Siegfried; Tzagarakis, Manolis M.; De Bra, Paul M.E. (Eds.) 2002.ISBN: 3-540-43293-0.
7. Alicia Diaz and Alejandro Fernandez. A Pattern Language for Virtual Environments. Journal of Network and Computer Applications. Vol. 23, No. 3, July 2000pp. 291-309. ISSN: 1084-8045. Academic Press.
8. Tomek, A. Diaz, A. Rito da Silva, R. Melster, M. Haahr, A. Fernandez, Z. Choukair, M. Antunes, A. Ohnishi, W. Wieczerzycki. Multi-User Object Oriented Environments. LNCS volume 1743. Springer Verlag.