Scandinavian Journal of Information Systems

Volume 11 | Issue 1 Article 7

1999

Cooperative work across cultural boundaries in systems design

Hilda Tellioğlu
Vienna University of Technology, Austria, hilda.tellioglu@tuwien.ac.at

Ina Wagner
Vienna University of Technology, Austria, ina.wagner@tuwien.ac.at

Cooperative work across cultural boundaries in systems design

Hilda Tellioglu & Ina Wagner
hilda.tellioglu@tuwien.ac.at[a] and ina.wagner@tuwien.ac.at[b]
Vienna University of Technology
Institute of Design and Assessment of Technology

Abstract

This paper is concerned with heterogeneity and the notion of boundaries within systems design. It uses case study material to identify sources of heterogeneity and to understand how these are oriented to within the practicalities of design work. A key concept for analysing work practices is the notion of place. The paper analyses the qualities of the places in which design work takes place (regionalisation, neutrality, specificity) and explores in how far these places allow membership in multiple worlds.

Keywords: CSCW, work practice research, boundaries, systems design

1. Introduction

Building software is about making linkages between a multiplicity of object worlds, practices, and commitments. One way of bringing these multiple worlds together is through shared tools and procedures, another way is through imagery, metaphors, and descriptions which can translate between different views of the product to be developed.

It is the necessity of working with and making productive use of heterogeneity and multiplicity which motivates our analysis of work practice in systems development. With this we address a central issue of studies of cooperative work: How do people engaged in systems design account for multiple perspectives - of designers with different knowledges, of management, of the multiple future users of their product, etc. - while at the same time ensuring cooperation across boundaries?

Our key concept for analysing the dynamics of heterogeneity in software development teams is the notion of *place* which is increasingly used within CSCW research (e.g. Fitzpatrick et al. 1996, Benford et al. 1996, Harrison and Dourish 1996). Benford et al. argue that spatial approaches to CSCW help understand phenomena such as the persistency of human cooperation over long periods of time, peripheral awareness, navigation and orientation. We build on an integrated sociological and architectural view on place (Lainer and Wagner 1998a). A place is a specific context of people, (built) environment, history, tools, events, life styles, etc. Analysing the characteristics of place is crucial in understanding "the multiple visible and invisible closures of interaction spaces and the dense, complex and multi-layered connections between people who are not necessarily co-present in space and time" (Clement and Wagner 1995).

We will elaborate this notion of place in our analysis of case-studies of design practice. The cases have been researched into and discussed as part of the *Software Cultures*[1] Project . All of them are cases of small software development teams (varying between 2 and 12 people). Each of them offers a different view on heterogeneity as a feature of design

1.'Software Cultures' was a nationally funded research project within the COST A4 framework (for the Final Report see Tellioglu and Wagner 1995). It combined case-study work in Austrian software companies with cross-cultural comparisons. This was done through a series of international workshops as well as cross-visits to Roskilde University, Risø Research Laboratory, and Lancaster University.

practice. Our own case-study work has been analysed in a previous paper with a focus on issues of cooperative work (Tellioglu and Wagner 1999), and, more specifically, on configuration management as a practice and tool (Tellioglu and Wagner 1997). In this paper we will widen our framework for a cooperative case analysis, including fieldwork done by our partners within the Software Cultures framework.

The cases differ markedly with respect to time, size and profile of the development team, and task. Also methods of data collection varied considerably. For the *Document Case* (1986-1993) we chose to reconstruct some of the crucial phases of the design work through in-depth interviews with some of the key participants and, in addition, analysed project documents such as the team's paper-based reports and memoranda. One of the projects described as part of the *TV Case* was well enough bounded in time and complexity to make a combination of participant observation with in-depth interviewing (in 1994, over the course of several months) possible. Regarding the *Danish Radio Case* (Kensing et al. 1998) and the *Foss Electric Case* (Carstensen et al. 1995, Carstensen and Sørensen 1995) we not only rely on in-depth discussions as well as interviews with their researchers but had the opportunity of doing some additional fieldwork. *The Dispatch Case* (Clement and Halonen 1998) was presented and discussed in one of the *Software Cultures* workshops.

Our analysis is structured as follows:

- Section 2 develops a theoretical framework for studying heterogeneity, based on the notion of place and boundaries;
- Section 3 presents the case-study material, discussing the managing of heterogeneity within software development teams;
- Section 4 discusses different aspects of heterogeneity in design work, substantiates our notion of boundary management and looks at how this is reflected in place;
- Section 5 draws some conclusions regarding the need for boundaries in design work and practices of boundary management.

2. Looking at Design Spaces and Boundaries

Talking about space and place requires an interdisciplinary approach. Some sociological work has explicitly looked at building(s) as "a domain of knowledge in so far as it embodies a spatial ordering of categories and a domain of control in so far as it involves an ordering of boundaries" (Prior 1990, p.92). From this work we conclude that heterogeneity is reflected in and supported by the spatial organisation of work.

Bucciarelli's studies of engineering design (1988) come closest to a spatial view of work practice. He makes a distinction between *actor worlds*, as the world of ongoing social interaction and evolving work practice, and *object worlds,* as the space of tools and representations or 'inscriptions' - as Latour points out:

"We are so used to this world of print and images, that we can hardly think of what it is to know something without indexes, bibliographies, dictionaries, papers with references, tables, columns, photographs, peaks, spots, bands" (Latour 1986, p.14).

The actor world of software development is heterogeneous, the design space shaped and occupied by different communities of practice (Jordan 1995), from software developers (with their different backgrounds and knowledges) to management, vendors, customers, special experts within customer organisations, and finally, end-users. In very large projects (such as building standard software packages) work is done by multiple (often distributed) groups each with their own standards and practices.

The physical places in which design work takes place are furnished with material artefacts of all kinds (among them computers, office equipment, manuals, books), and with all the knowledges of people and their object world of software tools, written code, specifi-

cations, flow charts, block diagrams, task lists and other inscriptions. These arrangements provide a place for both, people's ongoing interactions, and the objects they use and produce in their work. As Neumann and Star, in their study of a large digital library project, observe:

"Each team in the project has different objects that they are focusing on; these many different foci and building projects revolve around one generally defined goal. Rather than an 'object world' we are dealing with an 'object universe'" (Neumann and Star 1996, p.235).

Several key notions are used for describing the heterogeneity of these arrangements. On the one hand multiplicity is reflected in the *regionalisation* of the design space - its subdivision into areas that have been designated to specific persons and activities, including the distinction between more private and more public spaces. Creating (spatial) regions allows to accentuate, solidify, and create differences of work practices and knowledges, of culture and identity. Regionalisation often reflects existing power structures and dependencies. It may protect a special location and the vision it provides from powerful, potentially overriding views and interests (Clement and Wagner 1995).

Regionalisation is often expressive of the internal complexity of the product to be developed, with the needs for specialists or other relevant people located in different units of a company or outside to cooperate. One example are users who are often located at a distance from where the development work takes place. Developers and users may meet at particular places for communicating, negotiating and eventually aligning their multiple perspectives. These places can be a local or national union, a meeting room at the designers' place, the work place of a user, a consultant firm, an in-house development unit, or simply a coffee house. Each of these places is cultured in its own way which in turn shapes what can be made explicit, visible and shared.

A crucial question here is in how far a place can accommodate different perspectives and practices and if it allows membership in multiple worlds. We address this question through the notion of *boundary.* It focuses on the specific qualities of connections and transitions between regions. A boundary can be physical, social or organisational. So can a building's façade be seen as a boundary which mediates between interior and exterior spaces. While an envelope of rocky stones hermetically encloses the inner space of a building, a translucent glass façade supports the communication of its contents to the outside world (Lainer and Wagner 1998b).

Leigh Star (1991) uses the term *boundary object* for capturing the qualities of objects that are suited for maintaining or crossing the boundaries between regions and (disciplinary) communities of practice. It is a notion in two parts, stressing the fundamental ambiguity of objects (whose meaning is given in use) and the durability of arrangements. A boundary object is both, flexible and malleable (for actors to fill in their particular vision), and sufficiently durable (allowing actors to develop conventional or routine ways of coping with ambiguity, fuzziness and multiple meanings). A place is a boundary object (Star 1991), as it is both, flexible, shapeable and open for a variety of activities and forms of expression, and sufficiently defined for allowing the development of particular forms of perception and use.

Shared images, visions, metaphors can serve as boundary objects, helping to relate different levels of knowledge and expertise. On the technical-procedural level, formats, standards and procedures (including configuration management tools) play an important part in providing transitions between different regions of a design space. We will use the term *standardisation* here in a broad sense, referring to practices, images, notations, norms, tools, etc. which help actors to align their work so that it becomes readable and useable by relevant others.

3. The Cases

The five cases exemplify different sources of heterogeneity in design work as well as different ways of responding to them. Table 1 provides an overview of the cases and some of the key arguments we develop.

Table 1: Overview of the cases.

CASES AND ACTORS	PRODUCT	DESIGN SPACES: WORK PRACTICES AND OBJECT WORLDS	PLACES AND INTERACTIONS WITH OTHER RELEVANT GROUPS
TV Two loosely coupled one-person-teams	Specific application on Silicon Graphics platform with high performance requirements	Fragmented object world based on strict modularisation Cultivated differences of code and life styles	Strong interaction with omniscient end-user based on power and mutual trust Highly interactive testing and integration work Multiple places (home, office, TV studio, coffee house)
FOSS ELECTRIC Small group of in-house software developers	Software to configure and operate instruments for measuring the quality parameters of milk	Disciplining work practices through integration period (a temporal grid)	High status mechanical engineers as competitors Establishing protective walls
DOCUMENT Small team of developers Chief developer as vision provider	Standard software package with high internal complexity	Object world shared and developed from scratch Shared electronic space Disciplined work practices	Management, customer support people, vendors and end-users Chief developer as mediator between team and outside Strong knowledge-based boundaries The kitchen versus meeting spaces for controlled interactions
DISPATCH Dispatch Dept. IT Dept.	Dispatch facility for one particular customer organisation	Interpretive flexibility of the artefact: working tool versus stable system with well-structured code	Regional dispatch workers producing unauthorised versions
DANISH RADIO Project Group (researchers, IT Dept., users)	Introducing a Participatory Design approach into the organisation Prototype supporting radio production	Populating the object world with new kinds of artefacts designed for boundary crossings	Journalists as end-users PD practices Conflicting visions of product: PD process versus prototype

3.1 Maintaining Separate Worlds: the TV Case

The community of practice in the TV Case consists of two loosely coupled one-person teams. They have some good reasons for practicing such a fragmented approach to a project some of which date back to university times when the two first met. One is specialisation - developer A's strengths are project acquisition and the hunting of new products, development and graphical presentation tools. Developer B is a competent designer and developer of complicated graphics algorithms and knows to handle difficult tasks such as converting different types of files. Another good reason are the differences in life style. Both developers live in differently timed worlds. While developer A is a polychronic person who loves doing several tasks in parallel and finds it difficult to set time apart for non-work activities, developer B effectively plans and reduces his working time to regular periods of intense and directed activity. Finally, the two have completely different coding styles and their priorities for developing applications are not easily compatible. While developer B practices a highly structured approach to programming, sometimes using paper and flow or structure charts for designing a complicated function, developer A loves writing very long source files with global variables and non-structured procedures.

"His coding style, we would only fight with each other. Therefore I don't touch his code. If we have a problem, when it is not clear how to program, then we talk about it, for example how to structure the database, in a friendly manner. Usually he has a huge C-file with 2000 lines or something like this. And he hasn't even tried to write structured makefiles. He has just taken one makefile from somewhere and put the names of his program files into it. ... I am a fan of modular programming, everything separated into modules, no global variables, while he loves using global variables. ... I simply do not touch global data. For instance, I could not read his programs and he not mine, since I put the opening brackets into the same column as the closing one, and he puts them at the beginning of the row (not into the same column). I am used to building the code structure with tabulators, and he with spaces. ... My modules are hierarchical, at the bottom the in/out functions, on top of them those functions which call the others, a nice pyramid." (Developer B, TV Case)

The two developers' approach to software design-development is one of subdividing a task into functions which are developed separately and integrated in the very last minute. This is only possible, because both of them are willing to step over the boundaries of their own worlds at some point and to engage in intense cooperation. A shared space is created for meetings with their client, for testing, and for last minute adjustments.

An interesting feature of the TV Case is the intense interaction with an omniscient customer (a prominent TV journalist) who has some expertise in programming and a very strong notion of the functionality of the application and of many details of the graphical presentation. Interactions with this customer are dense and of a high level of immediacy. Meetings are used for ad-hoc demonstrations and for communicating change requests which require developers' immediate (and positive) response. There is also a strong moment of trust in this relationship with the customer, visible in the informality of the contract, the habitus of last minute changes, the customer venturing into using the system in front of the camera after only a few hours of training (with their support in the background) which shows both, his enormous self-confidence and his trust in their ability to deliver a good system.

3.2 Managing Integration Internally: the Foss Electric Case

This is the case of a software development team under threat and therefore in crisis. At one point it came close to top management closing down the project, as they did not trust the software designers to successfully complete their task[1]. Top management fired the software project manager and hired a consultant whose task it was to assess whether producing the software with the team at hand was a realistic enterprise. The consultant found the project problematic but manageable and realistic. While top management was still hesitant about

what to do, the software designers realised how critical the situation was and that there was a chance that management would outsource all software development tasks in the future. They decided to prove their ability to finish this project successfully. They worked on their own for more than a year without any outside intervention or control.

They had started out with a completely underspecified project and the constantly changing specifications of the mechanical, chemical and electronic parts of the complex measuring equipment continued to create major problems. As the team had no experiences with new methods and there were no methods and standards in the company for how to produce software for a specific product, it set out to develop their own platform (including a suitable programming language) and some common standards such as how to document the software, how to specify the functionality and how to structure the code. They also made some important decisions with regard to the organisation of their work. A spec-team of three was set up which was responsible for diagnosing the bugs and for making sure that these were corrected by the right designer within a reasonable time limit (the bug reporting system has been extensively documented by Carstensen and Sørensen 1995). A plan manager (one of the spec-team) is creating and updating the work plans for each designer. Depending on the activities and on changes in the program modules, the plan manager updates the work plan through re-aligning modules, tasks, actors, and times.

This is done during the so-called *integration period* which marks the time when all designers meet physically to align their work. The team has divided its work into two periods - four weeks for developing and one week for integrating the program modules. This week is also called *platform period*. The software designer who is responsible for creating a new version of the entire program in this predefined platform period is called platform master. This is a rotating assignment. Defining the structure of the program modules is done on the central computer where all files of one module are saved in one directory. Each designer is responsible for putting files, like source code, compiled code, resource files, etc., onto the server needed for the module at least at 12 o'clock Monday morning before the integration period starts. In this phase all meet in one room and explain to each other what they have done over the last weeks. The meeting is not terminated before they have agreed on how to handle the problems that have come up. Either they can decide on a solution during the meeting or they put it on the agenda for the next one. In these integration periods no programming takes place so that all files can be reviewed together and parallel developments and the resulting inconsistencies can be avoided.

In comparison to more elaborate configuration management practices, this was a minimalist, but for the time very effective strategy of aligning work practices (Tellioglu 1996). Setting time apart for integration work, and this in a highly disciplined way, is one of its main aspects.

3.3 Disciplining a shared design space: the Document Case

Work practices in the Document Case were shaped by a strong research orientation (and this at a time when there were few examples of best practice in software engineering on which to build), and a shared vision of the product to be developed - a document management system with a graphical user interface, including subprograms for scanning documents, for creating and managing document folders, for document retrieval and other server-based management functions such as database and jukebox management applications which should run on standardised platforms (MS Windows as client and OS/2 version 1.2 as server). As there was no suitable development environment for such a project at the time of

1.In this company, mechanical engineers are looked upon as the real designers in the project while software development is thought of as peripheral.

the project, since MS Windows (version 1.3) was still quite primitive and not supportive of multiple networked users, the team developed its technical platform from scratch.

Within a commercial environment a strong research orientation was outstanding. It was mainly encouraged by the chief programmer who incited a high level of interest in developing method-based programming styles, in systematising the program both in its structure and performance, and in acquiring and applying new programming tools and languages. Although the company had serious financial problems, there was always enough money to buy books, new software or to attend an important programmers' meeting or conference.

Most of the eight developers (and this was not unusual at that time) were students of computer science or self-trained. Each developer had specialised in one part of the whole program and took the responsibility for designing and developing modules, for building a subprogram, for testing its functionality, for integrating it with the other subprograms, and for finally maintaining it. Owning a module became an important identity-marker.

While these multiple voices were heard and their resources tapped in the many ad-hoc design sessions that took place, often at the request of one team member, the chief developer established a tight regime of practices for the implementation work. Driven by the idea of method, the developers artfully constructed their object world from textbooks, papers, and products. It was made transparent and readable to all inside the community of practice.

A lot of effort went into building a shared development environment with identical development tools being installed on each computer. This grew into a shared electronic space, with configuration management as a supporting discipline. Charts provided an overview of the program structure and the project status. The technical documentation contained detailed and regularly updated information on the software architecture, the dependencies between components, and application programming interface definitions, and this on both, a graphical and textual level. Among the highly valued and enforced best practices were the disciplined use of robust naming conventions and filling the module headers with detailed descriptions so as to facilitate the mutual understanding of each others' modules. Several types of comments were added to the code. On the one hand developers placed global descriptions of procedures onto the procedure header (including parameters, data types and the function of the procedure), on the other hand they wrote up explanations of each step (including the underlying rationale) of complicated code parts such as long loops, algorithms or dynamic data exchanges between the modules and subprograms. Testers created graphical representations of each testing step and of the behaviour of program modules they were testing. These documents helped developers to follow closely each step in the testing process and to locate the bugs and corresponding source code.

While building their internal design space, the team had almost securely sealed itself off. There were strong boundaries against the outside world which was shielded off as potentially hostile to the nature of the task which was seen as internally driven.

"... and when management or whoever, the sales people, say we have to make money, then it is sort of unavoidable to hand over the pressure. What do you do then? ... This is a real issue, how to protect the development team from the outside world. This is a must if you want to develop something. So, a product is ready for delivery only when it is finished and that's it." (Chief Developer, Document Case)

One of these boundaries was directed against the company's management. Management did not find it easy to understand and share the vision of the developer team which was oriented towards shaping itself into perfect programmers and developing the best standard software in the field. Several times management promised customers new versions without seeking to clarify the status of the product. The developers in turn simply ignored the deadlines set by management. Even urgent jobs were approached systematically. Equally restricted were interactions with the outside world of vendors and customers, with only completed versions crossing the boundaries.

3.4 The Flexible Artefact: the Dispatch Case

This case study looks at the in-house design of a system in support of dispatch work within a large utility company. It describes an evolving design space whose changing shape is closely related to the social construction of the personal computer. The system reflects an effort at gradually stabilising the PC while building bridges to the mainframe world (for which some rather strange intermediate solutions were worked out, Clement and Halonen 1998).

The system's biography was furthermore influenced by the changing constellations of ownership and responsibility for its design within the company. The dispatch system traveled between different departments and with each boundary crossing adapted its face to the perspectives and knowledges present in the new design space. Different actors clearly had a different view on the system. For the Dispatch Department it represented a tool intended to overcome past inefficiencies of dispatch work and a new way of working. People within the IT department treated it as

"an application that did not conform to proper standards of programming and for data security" (Clement and Halonen 1998, p.12[1]).

While for one set of actors the dispatch system evolved into a highly reliable and easily understandable support of their everyday work, the others perceived it as unstable (in its code).

Switching the prime responsibility for the construction and maintenance of the artefact from Dispatch to IT Department also meant that different features of the system came into focus. When the IT Department took over the system, it concentrated on

"clean up the code; design new module and prepare documentation; establish one uniform version of the system throughout the organisation; develop systematic procedures for rolling out new versions" (ibid, p.19).

This happened as a reaction to the many informal changes dispatch workers in the regions had made which actually made their work run much more smoothly. The move towards collapsing these different versions into one standardised product was clearly in conflict with local differences of work practice. Also, different actors' notions of accessibility and transparency of the system differed. Before this cleaning up, although "the artefact's 'insides' of program code were still a mystery and not directly accessible to most dispatch workers, they had indirect but timely access to the inside of the black box through their 'social' relations with such people as DM and D11 (members of Dispatch)" (ibid 1998, p.25). IT Department's notion of stabilising the system (e.g. by cleaning up 'spaghetti code') made it much more difficult to use for them.

Clement and Halonen summarise the salient events of the dispatch system's history, mentioning that "at each of these points the two groups disagreed on what constituted improvement of the system, and even disagreed on what the system was" (ibid 1998, p.15).

3.5 Designing Artefacts for Boundary Crossings: the Danish Radio Case

The *Danish Radio* Case started with the intent of testing the MUST approach (a participatory design approach developed by the Design Group at Roskilde University) and of establishing it within *Danish Radio's* IT Department. A project group was established, consisting of the research team (RUC), members of IT, two journalists (as user representatives), and some administrative staff (whose job it is to give secretarial support to radio production). All of them engaged in some field work (doing interviews, participant observations) aimed at understanding how radio production works, as well as in joint meetings.

1.Quotations have been taken from a draft version.

During these meetings an object world was built which from the start was designed as a boundary space, a space that represents and supports an evolving common vision. This space was populated by documents such as interview transcripts, loud thinking protocols, the posters produced during design sessions, etc. At some point, the IT people introduced data modeling (which is a quite common technique within software engineering but rarely used within a PD framework) as a tool for making the information flow during radio production (in particular temporal dependencies) transparent. This turned out to be a particularly relevant artefact since it enabled the research team to gain a clearer understanding of the work processes involved and informed the design of the prototype.

There was the double notion of producing these kinds of artefacts and making their construction transparent to the IT people so that they could adopt the approach in their own work. On principle, end-users should also be engaged in this process, but this idea was abandoned at a point where the pressure to produce a product took over.

Also in this case, different visions of the product to be developed existed, and their relevancy changed in the course of the project. While RUC was strongly interested in the result of testing their participatory design approach in an IT department, management and the journalists themselves expected some supporting system. While RUC's image of prototype was something rather provisional and incomplete, the radio people expected a system version which worked in a real environment.

4. Work Practices in Comparison

The five cases reveal different aspects of heterogeneity and multiplicity in design work. They also illustrate different ways of establishing boundaries and of arranging for boundary crossings.

4.1 Heterogeneity

The *Document* and *Foss Electric Cases* point to complexity as an aspect of design work which on the one hand requires to mobilise a multiplicity of knowledges, and on the other hand to bound the design space so as to make it manageable. De Michelis (1996) talks of two sources of complexity in organisations: multiplicity (of voices, knowledge, perspectives, etc.) and autonomy (of the actors involved).

Developers' task in the *Document Case* was to produce highly complex standard software. Their strategy for 'taming' the artefact was to build an homogeneous development environment and to install best practices. This effort at standardising was partly fired by the team's commitment to basing their work on method; partly this seemed the only way to cope with the complexities of the product. "It was an artificial world with its own rules", one of the developers characterised it in retrospect. The team stubbornly stuck to a time-consuming, method-based approach instead of satisfying customers' needs with a less ideally designed product. This detachment from the pragmatics of aligning one's approach to more mundane concerns such as deadlines reflects the team's autonomy which was successfully defended against management.

At Foss Electric the cumulative effect of unfavourable circumstances - the product-to-be-developed was completely underspecified, the team had no expertise with new methods, there was no support of software engineering practice within the company, and the team's ability to cope with their task was questioned - added to the complexity of the development task. This created a strong pressure to establish standard procedures, on the temporal (integration period) and organisational level.

We also see that there is a temporal aspect connected to the development of a project. While in the early design phase of a product there is a stronger need to mobilise heterogeneous resources in order to capture all the potentially relevant views and knowledges, when the design space is gradually narrowing down, issues of coherence, stability and

transparency of a solution come to the foreground. Then coding conventions, reporting and documentation procedures, including consistent descriptions of how different parts of a software or modules fit together, transparent practices of handling bug reports, change requests and design changes, version control, etc. become a practical necessity (Tellioglu and Wagner 1997). In the *Document Case* a unified code which could be read by all and a shared development environment allowed for flexible forms of cooperation - team members were able to switch roles and responsibilities, new developers had to be integrated. This is in contrast to the *TV Case* where the focus was on creating an executable program the inside of which was of little interest as long as the two developers knew how to handle errors within their own modules.

Another (and in practice often related) source of heterogeneity is the existence of multiple voices within a project, within the development team itself, and brought into it from relevant actors outside. The *TV Case* highlights the relevance of personal differences regarding coding styles, practices of handling time, etc. To allow for those differences in the way we described may only be practicable in a very small project team. In the *Dispatch Case* a multiplicity of visions and commitments was maintained over several years, since the perspectives on the artefact differed so much, between giving practical, "idiot-proof" support to dispatch workers and improving their practices on the one hand, and complying with the performance standards of the IT department on the other hand.

In the *Danish Radio Case* there was a clear mission of the research team to establish and test participatory practices of systems design within the organisation. Part of this was to hear multiple voices so as to gain a fuller understanding of the work to be supported. Multiple methods (from ethnographic studies of work to data modeling) were combined. At the same time, the heterogeneity of commitments and views made it difficult to shape the product. While the researchers thought of a prototype as enabling practical experiences with some central features of a supporting system, management and radio workers expected something which could already be used in a real work situation. Also, the applicability of the participatory design approach, although of principle interest, had less priority for them than for the research team.

Table 2: Managing heterogeneity.

CASE	HETEROGENEITY
TV	Multiple voices in writing code Separate object worlds, merging through integration work and testing One vision of the product (dictated by customer)
FOSS ELECTRIC	Producing a homogeneous object world through temporal and organisational measures Standardisation self-imposed (as response to external threat), unified product versions enforced within the team
DOCUMENT	Multiple voices in design vs. strict discipline in implementation Highly homogenised, transparent object world and work practices within developer team Disciplining on the semantic (vision) and procedural (best practices) level Standardisation self-defined and by conviction (in response to the complexity of the product and the lack of a stable platform)
DISPATCH	Heterogeneous object worlds (different technical regimes) and work practices Different visions of the artefact and its workings as unstable (in its code) and stable ("idiot-proof" in its support of work practice)
DANISH RADIO	Heterogeneity of approaches: MUST approach (RUC), data modeling (IT), work practice (journalists) Different visions of product: research report, working prototype, new practices of IT design

4.2 Boundary Management

We found different types of boundary objects with different qualities. The bug report form in the *Foss Electric Case* is an example of a single artefact designed for boundary crossings. In the *Document Case* the shared development environment (including a configuration management tool) was designed in a way which provided a platform for implicit communications. The standards embedded in this environment enabled the creation of a shareable artefact which could be examined and talked about. Another central artefact designed for boundary crossings was the technical documentation. This document crossed the boundaries to testers and documentation people, to vendors, and from there to the customers. It also was used in public presentations of the product.

Heterogeneity is purposefully made visible and an object of analysis in the *Danish Radio Case*. The kind of artefacts created by the participatory design group at *Danish Radio* - interview transcripts, thinking aloud protocols, data model, prototype, etc. - allow the joining of different forms of knowledge and this in ways that are useful for different purposes. They help to create a common vision of the work to be supported. The different voices have been built into the design of the project itself, in the form of multiple products (research report, prototype) each of which invites multiple readings.

Finally, we saw that the artefact-in-design itself may invite different interpretations and that these in turn influence in which ways it is used and further developed. This is particularly relevant in those cases in which an artefact is handed over to others (e.g. from Dispatch to the IT Department, from developers to end-users who do their own amendments thereby creating different local versions). The first working version of the dispatch system was described as "weird or unique, because of these boundary crossing attributes" (1998, p.20). Situated between two different technical regimes and being designed in the beginning to cross boundaries, the system at some point became both seriously anomalous and technically obsolete (Clement and Halonen, 1998).

How different voices and commitments are handled within a project obviously also depends on power relations, on how these are perceived and enacted. In some of the cases these are political in the sense that the relevant social groups have different world views, interests, and goals. While in some cases compromise may be achieved, in others difference cannot be simply negotiated away.

In the *Document* and *Foss Electric Cases* boundaries were tightly controlled by the designer-developer team itself - e.g. only developed versions were allowed to cross boundaries. At *Danish Radio* power relations were explicitly addressed as part of the cooperation agreement of the social groups involved in the project. The two designers in the *TV Case* carefully designed boundaries, subdividing a task into functions which were developed separately and integrated in the very last minute.

In the *Dispatch Case* boundaries were managed by handing over the artefact. Responsibility for the system switched several times, mirroring different alliances, their success to pull product into their perspective. The *Dispatch Case* is one in which an IT view shaped by standards of good software is competing with dispatch workers' interest in receiving stable and simple to handle support for their work. This is an ethical problem and it is not easy to judge (given our limited knowledge of the project), if increasing the stability and transparency of code would improve the performance of the system with respect to the work it is supposed to support (it seems it made it more difficult to use).

Table 3: Boundary management.

CASE	BOUNDARY MANAGEMENT
TV	Within the team temporarily during informal meetings, intensely in integration and testing period With customer while defining requirements, during testing, and while staging the life performance
FOSS ELECTRIC	Protective walls until delivery of the developed software Intrusions from outside in the form of changing specifications
DOCUMENT	Chief developer as mediator and boundary manager Carefully controlled boundary crossings during design and implementation Only completed new versions crossed boundaries to management, vendors and customer organisation No shared electronic space, official meeting spaces
DISPATCH	The artefact itself possessed strong boundary crossing attributes Responsibility for the artefact switched, from Dispatch Department to regional dispatch workers to IT Department
DANISH RADIO	Participatory design meetings and ethnographic studies of work Production of boundary objects (e.g. interview transcripts, loud thinking protocols, posters, data model, report, prototype)

4.3 The Role of Places and Regions

Spatial arrangements encapsulate and shape practices. They provide boundaries and opportunities for boundary crossings. The places we identified are formative of social practices, and this in various ways.

In the *Document Case* the strong internal orientation of the team found its spatial analogue in the kitchen on the one hand, and more formal meeting rooms for prepared and staged interactions with the outside world on the other hand. The kitchen was the place were the informal life of the team was concentrated and the rule to endorse the vision of the team was most clearly present. Failure to comply with this rule would result in being excluded from the kitchen. The kitchen was always crowded and used in the same way as in a private home where it often becomes a center of activities. In comparison to the kitchen, the more formal project related discussions, within the team as well as with managers, client representatives, visitors, etc., took place in the company's meeting room. This was a slightly more public place, furnished with the kind of equipment which supports more accessible forms of representation (e.g. flip charts) and offers some degree of anonymity. People who enter there leave their own work spaces, eventually carrying some artefacts with them, prepared for more controlled boundary crossings.

There is a parallel to the *Foss Electric Case* where the development team under pressure also sealed itself off from engineering and management. It is tradition in this company that all people involved in a project share one large physical space. The room fits the size of the project. The software team used the facilities of moveable walls for creating an enclosed space. The temporal and organisational analogue to this enclosed space is the so-called integration period when all developers meet in one room with no one leaving before an agreement has been found.

The separate worlds of the two developers in the *TV Case* and their relationship to their customer are reflected in their choice of place for different activities - development, integration work, testing and communication with the customer. Developer B works at home and uses the company's premises only to test the code or to link it to the rest of the application. At home he is master of his own time and style, protected from inspection and intervention, and closed off from the kind of casual sociability his colleague appreciates. When developer A is not on the move, he prefers to work in the company's office space shared with others over working at home. This space is equipped with a variety of partly specialised machines which are more or less accessible to all. In contrast to his developer-friend, he is always connected through his cellular phone. As both developers most of the time work in separate places, they make use of e-mail and the telephone and only meet to discuss urgent problems and the next steps in design and development, preferably in a coffee house. After such a meeting one of them writes up a memo and sends it to the other one via e-mail.

The Viennese coffeehouse is a special place. It is semi-public, intimate and neutral at the same time. It is a place filled with friends, semi-acquaintances and complete strangers, most of them unconnected to one's work and not interested in overhearing or participating. People may walk in to contact you knowing that this is the place where you might be found at certain times of the day. Developer A uses his cellular phone for conversing from the coffeehouse table with clients, colleagues and friends outside. Finally, there is the stage of the studio in front of the running cameras where the client performs using the technology designed for him and the two developers act as his visible background support.

Table 4: The role of regions and places.

CASE	REGIONALISATION	QUALITIES OF DIFFERENT PLACES
TV	Spatially separate workplaces Different places for different activities (home, office, TV studio, coffee house)	Work space specific, sheltered (home) vs. open, busy (office) Meeting place semi-public, intimate, neutral (coffee house) Performance in public place (TV studio)
FOSS ELECTRIC	One shared, open office space Protective walls against the outside world	Work space specific, sheltered
DOCUMENT	Enclosed office arrangement with a strong common center (the kitchen)	Specific, informal, sheltered (the kitchen) Neutral meeting place for controlled interactions with outside world

5. Conclusions

In our case discussion we have identified heterogeneity as an intricate part of design work.

"Systems of people and machines (are), situated and distributed, and most of all, deeply heterogeneous" (Star 1993).

This not only creates the need to take account of a multiplicity of perspectives and knowledges. It also requires a place for these perspectives. Thematising heterogeneity and boundaries in connection to practices, objects, and places, we argue, helps us to account for the trajectories of artefacts in much richer ways.

There are many good reasons for creating boundaries. One of the most prevalent reasons we identified in the cases is the need or desire to protect one's own perspective on the product to be developed and ways of working (the *Document* and *TV Case*). Related to this, in both cases, is a concern for the quality of the product, as visible in the strong research ori-

entation of the team in the *Document Case* (which made it seem necessary to fend off the outside world) or in the disciplined coding practice of developer B in the *TV Case*. There was also the idea to limit one's own responsibility for the product to internal technical criteria of efficiency and workability and let others (management, developer A) deal with users' perspectives and their usability problems.

Another reason for creating strong boundaries may be the motivation to survive in a threatening environment, as in the *Foss Electric Case*, where the team established a niche for building the expertise needed for proving their capability to develop a good product. In the *Document Case* there was also a need to 'tame' the product, to make its complexity manageable, which seemed to legitimate the construction of strong boundaries.

We can also find boundaries when there is a need to balance multiple and not easily compatible voices. In the *Dispatch Case*, the clash of two technical platforms surrounding the system and the disparities of conceptions are reflected in strong boundaries between the Dispatch and IT Departments. These boundaries were needed for stabilising the meaning of the system - Clement and Halonen describe how handing over responsibility for the system to IT Department led to an "increased instability of the artifact … not only … on the cognitive level - as a way of thinking about, talking about, and describing the system - but also on the level of concrete practices and procedures" (1998, p.12). In contrast to this, boundaries in the *Danish Radio Case*, were permeable, with the voices of radio journalists, the IT department and the researchers being heard as part of the participatory design approach. They reflect the needs of the different communities of practice to pursue their particular goals and vision of the product.

A variety of strategies of managing boundaries in design work can be pointed out. In some cases there are particular individuals who build and maintain connections - the chief developer in the *Document Case* is an example. We also saw that artefacts may assume specific boundary qualities or even be designed for it, like in the *Danish Radio* and *Dispatch Case*. The flexibility of the dispatch system, is not just interpretive in the sense that practices of use shape the conception of the system and of its working. We read its story as illustrating the boundary qualities of the artefact itself. As part of different object worlds different qualities are inscribed into it and it is actually adapted to the world it becomes part of.

Standardisation is a crucial strategy for making multiplicity and heterogeneity manageable. Systems design abounds with standards and in the *Document Case* a lot of effort was spent on developing standard procedures. 'Good standards', so it seems, have boundary qualities. Kjeld Schmidt (1997) emphasises this in his analysis of formal constructs which he sees as playing very different roles, from maps that support orientation and navigation within an emerging system, to scripts or protocols that determine particular activities (without making competent practice, as opposed to simply enacting a plan or applying a tool, obsolete). Forms restrict people's practices on the procedural and on the semantic level. Their boundary qualities reside in the fact that casting activities into a common format may make them usable by different, regionally distributed communities of practice.

In an ethnographic study of a project intended to produce middleware (bridging the incommensurate heterogeneity of other corporate products), Susan Newman describes the team's problem of

"how can 'we give ourselves the necessary insulation' in dealing with all this heterogeneous material" (1994, p.12).

She uses the term *contained heterogeneity* for capturing the need for flexible boundaries and open standards. In a much broader sense Leigh Star addresses this problem by pointing to the necessity of representing

"that which is permanently escaping, subverting, but nevertheless in relationship with the standardized ... In a sense, a cyborg is the relationship between standardized technologies and local experience; that which is between the categories, yet in relationship with them" (1991, p.39).

The cases also point to the importance of place for boundary management. A work environment may be internally regionalised, with small enclosed offices, or open and shared by the developer team. Places for work may be specifically expressive, communicating particular notions of type of work community, life style, etc., or more neutral, facilitating boundary crossings amongst people representing different worlds. In the *Document Case* the inside/outside boundary was reflected in the choice of spatially specific or neutral places for different kinds of interactions.

The specific qualities of people's workspace, with their rich furnishings of computers, manuals, sketches, books, journals, etc. offer a good sense of the type of work that is going on. Often occupants turn the office into an *exhibition space*, decorating the walls with visualisations and inscriptions from previous and current work - charts, post-its, images, schedules, etc. This is a good of implicitly communicating aspects of one's work and related thinking:

"This way of making work visible reminds of ideas to be pursued or further developed, of tasks to accomplish, of standards, etc. It also is an important vehicle for peripheral participation in a project, allowing visitors to enter its context. Conversations are opened up, designers are forced to explain to continuously changing interactors. They can create and communicate their identity without closing it too much" (Lainer and Wagner 1998a, p.198).

On the other hand such a place may be too much entrenched with the specific perspectives of its owners to be supportive of heterogeneity and multiplicity. Here more neutral environment of a meeting room or coffee house may offer an intermediate (uncoded) space for multiple voices to be heard, negotiated and aligned (Lainer and Wagner 1998a).

6. References

Steve Benford et al. Shared spaces: Transportation, artificiality, and spatiality. In Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work CSCW'96, Cambridge MA: 77-86, 1996.

Louis E. Bucciarelli. Engineering design process. In F. A. Dubinskas (ed.), Making Time. Ethnographies of High-Technology Organizations. Philadelphia, Temple University Press: 92-122, 1988.

Peter H. Carstensen et al. Object-oriented modeling of coordination mechanisms. In Proceedings of the 18th Information Systems Research Seminar in Scandinavia (ISIS): 1-15, 1995.

Peter H. Carstensen and Carsten Sørensen. Let's talk about bugs! In Scandinavian Journal of Information Systems, 7(1): 1-20, 1995.

Andrew Clement and Ina Wagner. Fragmented exchange. Disarticulation and the need for regionalised communication spaces. In H. Marmolin et al. (eds.), Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work ECSCW'95. Kluwer, Dordrecht: 33-49, 1995.

Andrew Clement and Chris Halonen. Collaboration and conflict in the development of a computerized dispatch facility. In the Journal of the American Society for Information Science, 49(12), 1998.

Giorgio De Michelis. Cooperation and knowledge creation. Conference on 'Comparative Study of Knowledge Creation', Honolulu, Hawaii, December 12-14, 1996.

Geraldine Fitzpatrick, Simon Kaplan and Tim Mansfield. Physical spaces, virtual places and social worlds: A study of work in the virtual. In Proceedings Computer Supported Cooperative Work'96, Cambridge MA: 334-343, 1996.

Steve Harrison and Paul Dourish. Re-place-ing space: The roles of place and space in collaborative systems. ACM 1996 Conference on Computer Supported Cooperative Work CSCW'96, Cambridge MA: 67-76, 1996.

Brigitte Jordan. Ethnographic workplace studies and CSCW. In D. Shapiro, M. Tauber and R. Traunmüller (eds.), The Design of CSCW & Groupware Systems. Amsterdam, Elsevier, 1995.

Finn Kensing, Jesper Simonsen and Keld Bødker. Participatory design at a radio station. In Computer Supported Cooperative Work: The Journal of Collaborative Computing, 7(3-4): 243-271, 1998.

Rüdiger Lainer and Ina Wagner. Vernetzte Arbeitsräume: Orte, Zwischenräume, An-Orte. In H. Lachmayer and E. Louis (eds.), Work & Culture - Büro. Eine Inszenierung von Arbeit. Klagenfurt, Ritter Verlag: 327-336, 1998a.

Rüdiger Lainer and Ina Wagner. Connecting qualities of social use with spatial qualities. Cooperative buildings - Integrating information, organization, and architecture. In N. Streitz et al. (eds.), Proceedings of the First International Workshop on Cooperative Buildings (CoBuild'98). Lecture Notes in Computer Science. Springer Verlag, Heidelberg: 191-203, 1998b.

Bruno Latour. Visualization and cognition: Thinking with eyes and hands. In Knowledge and Society: Studies in the Sociology of Culture Past and Present: 1-40,1986.

Laura J. Neumann and Susan Leigh Star. Making infrastructure: the dream of a common language. In J. Blomberg et al. (eds.) Proceedings of the Participatory Design Conference PDC'96. Cambridge MA: 231-240, 1996.

Susan Newman. Making difference manageable: Intermediation and control in a 'global' information system. 1994 American Anthropological Society Meeting, Atlanta, Georgia, 1994.

Lindslay Prior. The architecture of the hospital: a study of spatial organization and medical knowledge. In The British Journal of Sociology, 39(1) : 86-113, 1990.

Kjeld Schmidt. Of Maps and Scripts: The role of formal constructs in cooperative work. GROUP'97. ACM Conference on Supporting Group Work, Phoenix, Arizona, 16-19 November, ACM Press, 1997.

Susan Leigh Star. Power, technology and the phenomenology of conventions. On being allergic to onions. In J. Law (ed.), A Sociology of Monsters. Essays on Power, Technology and Domination. Routledge, London: 26-56, 1991.

Susan Leigh Star. Thinking paradoxically. Multiple regimes in the great divide. Paper presented at the Workshop Social Science Research, Technical Systems and Cooperative Work, Paris, March 1993.

Hilda Tellioglu and Ina Wagner. Software cultures. The influence of work organisation, management style and occupational culture on systems designers' approaches in a cross-cultural perspective. Final Report of the COST A4 Research Project. Abteilung für CSCW, Institut für Gestaltungs- und Wirkungsforschung, Technische Universität Wien, 1995.

Hilda Tellioglu and Ina Wagner. Negotiating boundaries. Practices of configuration management in software development teams. In Computer Supported Cooperative Work: The Journal of Collaborative Computing, 6: 251-274, 1997.

Hilda Tellioglu. Configuration management systems as collaborative systems in software production teams. In A. Sen and G. Ernst (eds.), Proceedings of the Sixth Workshop on Information Technologies and Systems WITS'96. Cleveland, Ohio: 152-161, 1996.

Hilda Tellioglu and Ina Wagner. Software cultures. Cultural practices in managing heterogeneity within systems design. Communications of the ACM, December 1999.