

Appropriating Patterns for the Activity Theory Toolkit

Elizabeth S. Guy

University of Brighton, School of Computing, Mathematical and Information Sciences

Lewes Road, Moulsecomb, Brighton, BN2 4GJ, UK

e.s.guy@bton.ac.uk

ABSTRACT

This paper investigates a method for modelling computer-supported cooperative work, to provide a common language for users and developers collaborating in design. The research is grounded in an empirical study of the in-house development of groupware and the work practice of system developers. Through an appropriation of Christopher Alexander's architectural pattern language, it is proposed that patterns have the potential to be a practicable tool that both embodies the principles and methodology of activity theory, and fits the requirements of this design process.

Keywords

Design patterns, pattern language, activity model, computer-supported cooperative work

INTRODUCTION

Over the past fifteen years there has been a steady growth of interest in activity theory (AT) among a new audience in the fields of Human-Computer Interaction (HCI) and Computer-Supported Cooperative Work (CSCW). Despite a growing corpus of research using it as a theoretical approach, there has not been a corresponding initiative in developing AT as a *practicable method*. By 'practicable' we mean useful to practitioners in real-world projects for the design of information systems (IS), user interaction and the associated work practice, rather than academic research. One of the reasons for this is possibly that AT is perceived, by its new audience, as being "[...] hard to learn" (Nardi, 1996, p.9). Mastering AT can be demanding for someone with no background in its philosophical traditions. The question of whether it is realistic to expect developers and users to be receptive to methods based on such seemingly alien principles is a valid one.

Despite the difficulties, researchers who are convinced that AT is well worth the trouble might feel obliged to engage with this problem. This means developing tools that are not only grounded in AT, but suit the needs, culture and constraints of practitioners' work. In order to explore the problem we first look at a precedent - the way that equally "hard to learn" theory has been effectively embodied in traditional IS development methods. We then consider the

essence of AT and its implications for tools that support the application of AT in design. The practical needs of systems developers for lightweight tools that support collective reasoning about design are considered, based on an empirical study of the in-house development of groupware to support information sharing. Next, we present a partly principled, partly opportunistic appropriation of Christopher Alexander's architectural pattern language (Alexander *et al*, 1977; Alexander, 1979). This is justified as a design tool that both embodies the principles and methodology of AT, and has the potential to fit the practice of systems developers and interaction designers. In the final sections the method is described in the form of a practical guide, followed by a discussion of the further work that remains to be done to evaluate it. In the Appendix we have included a subset of four patterns as a demonstration of the technique.

COMPARISON WITH THEORY IN SAD METHODS

The failure of researchers to develop AT as a tool for practitioners suggests that many of us, after the effort of applying it ourselves, may feel that AT is simply too difficult to be used in this way. However, there are several precedents in the way that other theoretical approaches have been explicitly and implicitly embodied in practitioner methods in the fields of IS and HCI. To take just one example, the way that traditional systems analysis and design (SAD) methods have incorporated the concepts of systems theory is a heartening example of how this can be achieved. Systems thinking, with its origins in many disciplines such as theoretical biology, cybernetics, and information and communications theory (Lilienfeld, 1978; Checkland, 1981), is a hodgepodge of abstractions about social organisation as 'systems'. Systemic concepts are embodied in SAD methods and techniques, such as Checkland's Soft Systems Methodology (SSM) (ibid 1981; Checkland and Scholes, 1996), perhaps the purest embodiment of systems theory as a practical method for modelling "human activity¹ systems". SSM has a set of procedures and a number of modelling techniques, including a Formal Systems Model that is a checklist of the characteristics that are said to describe a system.

¹ Activity here is used in a very different sense to the way that it is used in AT. Checkland's concept of a human activity system is that of an ideal, notional construct rather than real, concrete activity (Checkland, 1981, p314).

In the past, SAD methods have been practiced in the professional development of large transaction processing systems and, more recently, corporate databases. Practitioners have apparently not been deterred from using them by their problematic theoretical basis. Indeed, many text books on systems analysis present the techniques without any exposition of their foundational theory at all. This is not to say that in order to be acceptable to practitioners AT-based methods and techniques should adopt this approach and, like Clark Kent, keep their real identity and super-powers a secret. It does suggest that it is possible to embody theoretical principles in practicable methods, without making the unrealistic demand on practitioners that they should first become experts in the theory, before being able to use them.

DESIGNING ACTIVITY THEORY TOOLS

The SAD example may demonstrate that the project is achievable, but, unlike systems theory, AT is not a set of abstract propositions with which to describe the world. Rather it can be understood as a methodology that seeks to understand activity in the process of change and development, and as a tool for intervention in change. It is therefore vital, in seeking to present AT as an accessible and usable method, not to fall into the trap of using it as a set of categories for merely describing activity, similar to the approach of Checkland's Formal Systems Model. Some recent discussions of AT have tended to move in this direction and to emphasise its undoubted usefulness as a tool to describe the totality of the social organisation and context of work (Nardi, 1996, p.3).

AT uses the conceptual tool of contradictions (Engestrom, 1987; Bertelsen and Bodker, 2003) to reveal the underlying dialectical relations that drive development. Designing a method that embodies the principles of AT requires not just that it should be able to describe the categories that are the outward form of activity. It should also support the revelation and explanation of the dynamic inner processes within activity systems, so that developers and users may be able to use it as a tool for change-oriented design. In order to do this, it should support the historical analysis of these processes; reconstruction of the process of change; and identification and representation of innovative ways of working with tools as they emerge from contradictions, using them to inform the design of future systems.

Vygotsky, writing about the development of a method adequate to explain the nature and development of psychological processes, talks about method as something that is "[...] simultaneously prerequisite and product, the tool and result of the study" (Vygotsky, 1962, p.65). This suggests that a method founded in AT should not be a fixed or rigid thing, like a prescriptive SAD set of procedures and techniques, but rather an artifact that can be dynamically configured and adapted to the requirements of the design project. It should be able to evolve historically as the field of design changes and developers learn from experience.

SYSTEM DEVELOPMENT PRACTICE

An AT-based method has not just to be consistent with the methodology of AT, but also to fit the practical needs of developers. The case study and 'facilitator' set of patterns presented in this paper (see the Appendix) come from the findings of a wider research project into systems development practice, carried out between 1997 and 2002 (Guy, in preparation). The field site for the research was the international centre of 'GreenFam'², a global non-governmental organisation which campaigns for human rights, based in London, UK. The work of the centre involves centralised administrative support for the GreenFam movement: coordination of global campaigns; research carried out by teams organised around world regions and sub-regions; and the mainly in-house provision of information technology (IT) support for these activities. The objective of the extended study was to study the methods and techniques used by GreenFam's IT Program (ITP), during a major, long term project. This involved supporting teams with tools for information sharing and the coordination of campaign work, using the technology of LOTUS NOTES™ (Notes).

The GreenFam study was carried out in two phases: participant observation over eighteen months during the initial stages of the Notes project; returning three years later to evaluate several Notes databases as they had evolved through use, in order to produce guidelines to inform future design projects. One of the outcomes from the project, in particular from the evaluation phase, is a GreenFam pattern language (Alexander *et al*, 1977) which is intended to represent some of the findings in a form that can be used as a tool for design. The patterns in the Appendix are a subset of over twenty GreenFam patterns which cover the design of common information spaces (Bannon and Bodker, 1997) for the support of work with information artifacts.

The Right Tools for the Job

Throughout the Notes project it was evident that the developers at GreenFam lacked the right tools for the job. They were experienced in the process of "rolling out" standard software to users organisation-wide, and had envisaged a modified roll-out where a template Notes document and discussion database would be provided for each team or campaign, without much need for further customisation. This method was inadequate in a context where the requirements and culture within and between teams varied enormously. The ITP, it transpired, in fact knew very little about the complex ways in which users worked on and with information. An alternative approach adopted in one project of intensively "working with users", involving prototyping and group meetings over a period of

² Names have been changed in order to protect the identities of the research partner and the people working there.

several months, was a valuable learning experience, but had too high an overhead to be practicable in every case.

The GreenFam developers needed what they articulated as “a new kind of analysis” and tools that would help to capture configurable, reusable design solutions, when designing each application from scratch was not feasible. This resonates with John Carroll’s justification for the method of designing with scenarios - “Systems development is now in need of a guiding middle-level abstraction, a concept less formal and less grand than specification, but that can be broadly and effectively applied.” (Carroll, 2000, p.17.)

The vision for a new, more collaborative way of working at GreenFam had been set out in a document which evaluated the different software alternatives before the project began. Scenarios envisioned a future way of working, where Notes databases would be used to share information and to create a permanent archive that could be accessed as a repository of past experience. The vision was communicated top-down from GreenFam’s decision-making bodies and was embraced by the ITP. For them the scenario was “[...] a prototype of [...] future-directed action – in which the future is more than the blindly inevitable fact of succession in time and includes some envisioned goal as its content.” (Wartofsky, 1979, p.141.)

This was not a vision that was shared throughout the organisation and it represented a big departure from current practice for many teams. Many individuals took responsibility for managing their own cases and personal networks of information sources, and did not perceive a need to put information in common. Where information was shared, existing tools such as email were preferred to the overhead of adopting a new tool and way of working.

As the project progressed and the ITP experienced the difficulty of gaining acceptance for the new tools, doubts and dissent surfaced among the developers. The Notes project crossed the functional boundaries which separated development of databases, and support for document production and management, into two specialised teams. Each team interpreted Notes in the light of their previous experience and it became clear that there was no shared understanding of what Notes was or how it should be used among the members of the ITP either.

The problem for the work of the ITP was a lack of appropriate secondary artifacts (Wartofsky, 1979) – models for representing alternative design solutions that could mediate between different groups and viewpoints. Taxén (see these proceedings) describes the importance of a method or tool to establish a “working consensus” among actors in complex, distributed projects. In particular, participants need to establish shared meanings for abstract concepts – such as whether Notes is a database or a document management system or, indeed, an environment for collaborative work. Bertelsen and Bodker’s (2002) metaphor of the “parallel rooms” of design practice and use

practice (*ibid*, p.410) is apposite for describing this problem. They relate the discontinuity between these parallel rooms to the heterogeneous groups who participate in design. At GreenFam we found that these discontinuities extended beyond the dyad of design and use, to the different practical cultures within the ITP, that had become established over time.

It follows from the GreenFam study that artifacts to support collaborative design must satisfy several conditions:

- Models must be equally accessible to developers and users, and be a *lingua franca* for dialogue and discussion (Erickson, 2000b).
- They must mediate between a vision of a future way of working with new tools, and past, present and emerging practices, in order that developers and users can engage with this design space and co-construct new solutions.
- They must support representation and resolution of the dynamic and contradictory features of mediated work.

IDENTIFYING AND APPROPRIATING PATTERNS

One of the benefits of a lengthy program of research in one organization is that after a while strong, recurring patterns of what works and what does not can be identified as they emerge in new practices with evolving tools. For example, there were some instances where Notes databases were used by teams as a real shared information space and became integrated with their work. Such examples were always associated with an enthusiastic individual who acted as a facilitator and took on the responsibility of posting information to the database, directing other team members to it. Crucial to this work was the ability to make use of email, the historically embedded and preferred tool of the organization, to send hyperlinks in messages. These, when clicked, took the recipient directly to the document in the database. As we began to identify patterns of activity in our field study, we were inspired to represent them in the form of a pattern language, modifying a method that has a growing following in software engineering (SE) and HCI.

Christopher Alexander, the creator of the architectural design patterns that are currently in vogue in object-oriented SE (Gamma *et al*, 1995), use case specification (Adolph *et al*, 2003), web site design (Graham, 2003; van Duyne *et al*, 2003), interaction design (Borchers, 2001a, 2001b), CSCW (Erickson, 2000a; Martin *et al*, 2001) and participatory design (Dearden *et al*, 2002), is a rather strange bedfellow for AT. It is beyond the scope of this paper to give a detailed review or critique of his work (Alexander *et al*, 1975; Alexander *et al*, 1977; Alexander, 1979; Alexander *et al*, 1985), but also not necessary. Just as with these other approaches that have appropriated the patterns idea, our development of patterns as a tool for AT-based design departs opportunistically from Alexander wherever his philosophy contradicts that of AT. Alexander’s guiding principles of the timelessness and naturalness of archetypal patterns that have “the quality

without a name”, and the “invariance” of design solutions that have this quality (*ibid*, 1979), are antithetical to the method of dialectical materialism. However, other Alexandrian principles are more directly transferable:

- The empowerment of users to shape their own environment through the tool of an accessible, shared pattern language with the support of an expert facilitator.
- The definition of a pattern as a “three-part rule, which expresses a relation between a certain context, a problem and a solution” (Alexander, 1979, p.247). This contextualisation of problems and solutions, which may be historical or to do with the current conditions, is consistent with the method of AT.
- The definition of a problem in context as being caused by a system of forces which arises in that context (*ibid*, p.253). This lends itself to being translated into the language of AT’s contradictions.
- The scope, modularity and unity of the systemic pattern language, which is made up of a network of related patterns. Patterns can be written for different levels of activity, actions, and for the conditions for operations (Leont’ev, 1979, 1981) while retaining the integrity of AT’s unit of analysis.

THE ACTIVITY PATTERNS METHOD

Having outlined the origins and theoretical basis of activity patterns, and discussed the empirical study in which the method is grounded, this section will briefly describe a practitioner method for designing activity with patterns. The context for which this method has been developed is the in-house development and customisation of proprietary groupware in a non-profit organisation. This has coloured the approach, but we believe that using patterns in the analysis and design of collaborative systems has a wider relevance. They could, for example, be used for modelling activity in the application domains of collaborative learning environments, interactive web sites, or new media for domestic or leisure applications. The method is well-suited to in-house development as it supports user participation in the design process. Consultants and software producers – who work in contexts where software and interaction design patterns are already being generated – might also find that activity patterns are a useful addition to their repertoire of techniques, because of their reusability.

Activity Patterns in the Design Process

An open question for this project is whether practitioners will be prepared to make the effort to learn AT. In the method proposed here it is necessary to learn some basic principles of AT, but the trade-off is that activity patterns require no special skills to write and are very versatile. They have the potential to be used in any part of the design process where an understanding of tool mediated work in its context is required:

- generalising data from a requirements investigation – literally discovering patterns in users’ work;
- specifying and documenting the higher level conceptual design of systems;
- participatory design and prototyping;
- as criteria for the evaluation of prototypes or systems in use;
- representing the findings of evaluation in a design-oriented way;
- as a benchmark to evaluate the support delivered by software products before purchasing.

Activity patterns are modular and can be re-used in similar contexts, thereby cutting down on the workload of systems development. They interface well with lower level system specification techniques, such as UML use cases or interaction design pattern languages (for example Borchers, 2001a; van Duyne *et al*, 2003). The patterns presented in this paper are addressed to the problem of broadly representing activity and actions, but not to the detailed modelling of how actions are accomplished as sequences of tool mediated operations (see papers by Bertelsen, Harris in these proceedings, also Harris, 2004).

Embodying Activity Theory Principles in Patterns

The unit of analysis of AT (Engestrom, 1987) is not something that can be decomposed, but is a dynamically related system. A *pattern language* preserves this unity. Although discrete patterns can be written for the elements of an activity system – for example, the design of *tools*; the work of a *subject*; *rules* such as organisational policy and procedures; the roles within the *division of labour* or *community* of the workgroup – each pattern is related to the larger patterns it helps to complete, and the smaller patterns that complete it. This is done by stating the associations between patterns in the definition of each pattern (for how this is done see the template in the Appendix, *Table II*). In this way a systemic pattern language is constructed, consisting of a network of explicitly related patterns. It would be a nonsense to consider any pattern in isolation: for example the ONE CLICK HYPERLINK or EMAIL ALERTS patterns in the Appendix are only meaningful in the context of a specific social and technical context. The same goes for the higher level patterns. The policy of treating INFORMATION AS COMMON PROPERTY is the policy of a concrete activity system; it is realised through its related lower level patterns.

We have used another AT concept in the activity patterns – the idea of hierarchical levels of activity (Leont’ev, 1978; 1981). This is used in a similar way to Alexander’s use of ‘scale’ to structure the scope of his patterns for the built environment. An activity is motivated by a human need to transform some aspect of the material world. Activities are realised through concrete actions which are directed to achieving specific goals. Actions are accomplished through a series of operations which are performed under the specific conditions in which an action takes place. As they are learned, operations are internalised and become

automatic, providing that the tool supports them in intuitive ways. The levels of activity, actions and operations are summarised in *Table I*.

Activity	Motive	E.g. Sharing information; organising a campaign
↑ ↓	↑ ↓	↑ ↓
Action	Goal	E.g. Posting a news article in the information space; writing an email alert message
↑ ↓	↑ ↓	↑ ↓
Operation	Conditions	E.g. Clicking a hyperlink to open a document or an action button to send an email

Table I. Levels of activity with examples from GreenFam.

The levels of activity – although they are hierarchical – cannot be decomposed. Patterns can be written for different levels of activity and the relationships between patterns once again preserve this unity. We have included patterns in our examples to demonstrate how they can be used to model these different levels. INFORMATION AS COMMON PROPERTY relates to the *activity* level; EMAIL ALERTS represents a tool-mediated series of *actions*. ONE CLICK HYPERLINK is at the level where the action could then be modelled as a series of operations, using a unit of analysis focussed on the individual (see Harris, these proceedings).

The concept of contradictions is fundamental to AT, and central to the representation of activity in the patterns. Christopher Alexander’s definition of a pattern is, as we have seen, a rule about how to resolve a “system of forces” which always arise in a given context. Alexander’s approach is often profoundly idealistic and it is not always clear exactly what he means by a system of forces. Although he defines a number of different kinds of forces relevant to the configuration of space in the built environment (Alexander, 1979, p.248) there is no coherent theoretical foundation. His concept of a force is certainly very different from the understanding of a contradiction in AT. In a statement which is anathema to AT he defines his holy grail - the “quality without a name” - as the freedom from inner contradictions (*ibid* p. 26).

In contrast AT sees activity systems as being in essence dynamic, undergoing a process of change and development. The driving force behind that dynamic is contradiction – the continual breakdown and temporary resolution of the inner relations within the system. There is no such state as freedom from inner contradictions in activity systems: to hold this would be to deny that people are continually re-shaping their environment through their activity and, in so doing, changing themselves as they learn through their experiences. We have therefore reinterpreted Alexander’s definition of a pattern and the problem statement at its core:

An activity pattern is a three-part rule which establishes a relationship between a context, a contradiction that arises in that context, and its resolution, which takes it from its current state to a more developed one.

We do this with the proviso that any pattern only resolves a contradiction temporarily and that a pattern language must evolve as activity and its context changes. Some patterns will have more longevity than others – depending on the rate of change in the activity they represent – but none will be timeless. It is the ability of a pattern language to change and develop that makes it such an apposite tool for AT.

In summary, activity patterns embody the concepts of AT – its unit of analysis; the hierarchical levels of activity, actions and operations; and the contradictions which are the driving force for change, which have been made a central part of the pattern. We will now go on to describe a framework for writing patterns and developing a pattern language through applying it in the design process.

DESIGNING ACTIVITY WITH PATTERNS

It is a fundamental principle of the activity patterns method that the pattern language evolves continuously as it is put to use. It is both a *tool* that is applied in projects and an *outcome*, as what is learned from the project is fed back into further development of the language. This feature of the tool fits well with the rapid pace of change in technologies to support complex work. It also means that it can be used to capture and share the design knowledge and expertise of developers as they learn. We will now describe how to develop an evolutionary pattern language as a network of actions which are shown below in *Figure 1*.

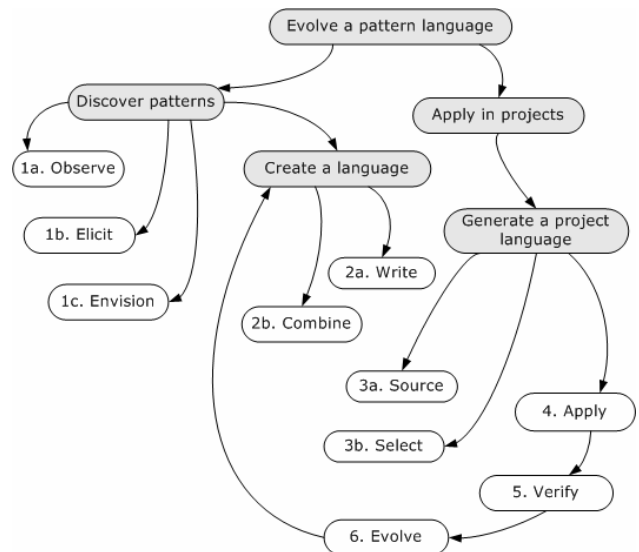


Figure 1. Evolution of a pattern language.

1 - Discover Patterns

Generally patterns are derived from experience and so may be discovered in two ways – either from direct observation of how people are using tools in context, or from the accumulated experience of developers and users. However, we also see the potential of patterns for visualising and designing future systems and ways of working. They can be used in this way to change people’s perceptions of what is possible, and bridge the space between the current activity and what is desired for the future. We therefore

describe three ways in which patterns may be discovered – by *observation*, *elicitation* or *envisioning* the future.

1a - Observe

Empirical data about how users do their work is typically gathered in the preliminary requirements investigation phase of a project; or through evaluation of systems in use; or, if a participatory approach is adopted, through co-design and prototyping. Material for patterns could be discovered through any of these activities, using qualitative research techniques such as observation, workshops or interviews carried out with users as they work.

Requirements investigation has the potential to generate huge amounts of data about the work of individual users, and analysing this data is a process of discovering relevant themes. Patterns are an effective way of generalising some of the main findings in a form that is oriented to design. Once written up, the patterns can be used to check and validate the findings with users, and then be improved. Patterns are a technique that help to make sense of the rich and often inconsistent data from a workplace study, without moving prematurely into lower level modelling techniques that represent the operational level of work.

Being aware of the concept of contradictions can sensitise developers to look for the sources of a failure to progress and tensions in activities, and how these might be resolved. They should ask questions such as – ‘What do the groups and software solutions that ‘work’ and are developing in line with expectations have in common?’ ‘What is going wrong or is absent in the examples where change is not occurring or is proving difficult?’ Positive examples may often point the way to possible resolutions of contradictions – by redesigning the tool, or by taking steps to change working practices.

1b - Elicit

“Your pattern language is the sum total of *your* knowledge of how to build.” (Alexander, 1979, p.203.) In a sense all designers use established patterns as a resource whenever they bring their design knowledge, the sum of their experience, to bear on a new problem. Users have domain knowledge, and proven solutions to problems that they use again and again. Eliciting the expert knowledge of developers and users in a pattern writing workshop is another way of getting started with the creation of a pattern language, in order that experiences can be shared and used by others.

1c – Envision

Patterns of future activities – like future scenarios – can be written as a tool for investigating how activity is going to change in order to realise new outcomes, and how activity systems should be designed to support it. Patterns in this sense are experimental and should be written collaboratively with users in order to involve them in discussions of how their work could develop.

The pattern INFORMATION AS COMMON PROPERTY sets out a space for development between an envisioned future activity system and what is happening in the present. When GreenFam adopted the policy of working more collaboratively with information, patterns could have been written to represent how this change could be realised. In fact the developers learned through a process of trial and error, implementing collaborative information spaces, some of which were more successful than others. Future patterns have to be realistic and achievable, and so must be grounded in a thorough understanding of how work is done in the current situation.

2 - Create a Language

Writing patterns requires no special skills other than the ability to think and write clearly, although this does not mean that it is easy or straightforward to do. Unlike other system specification techniques, the fact that higher level patterns need no specialist technical knowledge means that users are likely to be just as skilled at writing them as developers (if not more so). For that reason they are a good technique for involving users in a design project.

Alexander and other writers (e.g. Dearden *et al*, 2002) see a pattern language as a tool to empower users to participate in designing, but we would recommend going one step further and involving users in actually writing patterns. The object-oriented software community has developed the practice of writers’ workshops, where writers present their patterns for discussion and critique, in order to refine them. Workshops are lead by facilitators who are experienced pattern writers. Users and developers could adopt this practice and co-write workplace patterns in a participatory workshop.

2a - Write & 2b – Combine

These two actions are performed together. Patterns are written and then combined with the other patterns to form a unified *pattern language*. This is done by explicitly defining how they are related to higher and lower level patterns in the introductory paragraph and conclusion of the pattern (see *Table II*). Pattern writers will probably have to go through several iterations of this process in order to formulate a pattern that is sufficiently robust to be applied and tested in practice. Factors such as writing style, typography and layout are important for the clarity of the pattern: it is a visual representation as well as a text. A pattern template – adapted from Alexander – is shown in the Appendix, *Table II*; this specifies the content and form of a pattern. This template has been used to construct the four patterns that are also included in the Appendix, together with an example of how to represent a pattern language as a map or network in order to show the associations between its patterns (*Figure 2*).

Notes on the Pattern Template

- Naming is part of designing. The pattern name should express exactly what the pattern does in an emotive

way: this helps users to remember it and facilitates use of the pattern language.

- The ranking of a pattern – whether it has two, one or no asterisks – conveys the writers’ confidence in the validity of the pattern and the extent to which it has been verified through application.
- The illustration acts as a mnemonic and communicates how the solution can be implemented in a real example. Because of the intangible nature of systems design (unlike architecture and building) it might be necessary to use a metaphorical illustration for some patterns: in the pattern INFORMATION AS COMMON PROPERTY the picture of the library is a metaphor for a shared information space, with its dual character of information repository and space for collaborative work.
- Alexander says that “If you can’t draw a diagram of it, it isn’t a pattern” (1979, p.267). We have used two types of diagram: variations on the conventional activity triangle (Engestrom, 1987), and UML use cases to specify the functionality of lower level patterns. Other modelling techniques that could be used are: rich pictures, story boards, UML activity diagrams (for patterns of workflow procedures), screen shots, paper sketches of interface objects. The only restriction is that the diagrams should be fairly easy for anyone to understand and draw, and not limited to those with technical know how.

Application in Design

The following actions cover the application of a pattern language in design projects and using this experience to evolve the language.

3 - Generate a project language

The pattern language is used to generate a range of possible concrete solutions for specific design problems. The whole pattern language will not be required in any project, but a subset of patterns from it to fit the project’s scope. Generating a sub-language for a project is the key design activity – this amounts to specifying part of the conceptual design of the activity system by selecting the patterns that are applicable. There are two possible actions for configuring a pattern language for a project.

3a - Source

A growing number of pattern languages for interaction design and software design are currently being published. Developers can source patterns from any of these collections in order to augment their own languages if they need additional patterns for a project.

3b - Select

The main action is to select patterns from designers’ own pattern language. Selecting patterns is designing and this is therefore the most critical part of a project. Selection should be done in a workshop involving members of the development team or, if appropriate, developers and users.

The workshop should be facilitated by someone who is experienced in using patterns; has both domain and technical knowledge; and understands the underlying concepts of AT embodied in the patterns. The job of the designer-facilitator will include interpreting the patterns and giving examples of how they have been used.

- First look at the pattern language map and tick off all the ones that are relevant to the project. Think about the tensions that are likely to occur and the changes that will be needed as a result of introducing new software: find patterns that have helped to resolve these contradictions in the same context.
- Then select a top level pattern that describes the scope and motive of the activity in the way that our pattern INFORMATION AS COMMON PROPERTY defined the activity and motivation of the GreenFam project. This pattern, when it becomes part of the project language, will not be related to any higher level patterns as it sets the overall context.
- Finally select from the available lower level patterns top-down, until the lowest level activity patterns that are required have been identified.

System design workshop techniques such as affinity diagrams or card-sorting can be used to physically interact with the patterns. Following Dearden *et al* (2002) we recommend printing patterns out on cards to facilitate handling them. The cards should have a summary of the pattern on the front (identifier, name, visual representation, context, short contradiction and resolution statements, references to lower patterns); the full pattern on the back.

Patterns are not intended to be a ‘one-size fits all’, off the peg solution to a problem, but a starting point to stimulate design discussions. In this sense they fill the role of what Bodker and Christiansen call “springboards” for design (1997). It is more than likely that new patterns will be needed in any project - the pattern design workshop might have to adapt patterns to the specific context of the project, write new ones and prune patterns that have become obsolete. As well as the pattern map and the patterns themselves, a template for editing and writing patterns is an essential workshop tool.

4 - Apply

The last three actions – Apply, Verify and Evolve - form part of an iterative cycle of testing patterns through use, evaluating the outcomes and using this experience to evolve the pattern language.

Alexander has a number of principles for building construction that are relevant to software design and even mirror methods that are used by practitioners. One of these is the principle of “piecemeal growth” (Alexander *et al*, 1975): developing a large complex of buildings in usable increments as opposed to “large lump development”. This principle resonates with the approach of rapid application development and agile methods in SE. He also has a

pattern for a process of GRADUAL STIFFENING (1977, p.962) for the construction of buildings. This pattern sets out a rule for “[...] weaving a structure which starts out globally complete, but flimsy; then gradually making it stiffer but still rather flimsy; and only finally making it completely stiff and strong.” The rationale for this pattern is that not all aspects of a design can be pre-specified: many changes may have to be made as a design is built and ‘flimsiness’ in the earlier stages facilitates this. Iterative development, starting off with flimsy and easily changed design models such as paper prototypes, followed by software prototyping, is a commonly practiced way of achieving ‘gradual stiffening’ in the design of systems.

5 - Verify

As patterns are implemented in concrete prototypes and new systems they can be verified. The ongoing evaluation of patterns is essential if the pattern language is to develop in line with changes in the field of design. Patterns can be verified not just by observing situations where implementation of a pattern has been a success and helped to support change. It can also be done through observation of counter-examples where a pattern is lacking and contradictions are apparent.

The facilitator patterns have been verified using both approaches. At GreenFam our evaluation of Notes databases in use found several instances of facilitators using the tools defined in the EMAIL ALERTS and ONE CLICK HYPERLINK patterns. In these cases participation in the shared information spaces was much better than the examples where there was no facilitator, or where there was no encouragement of users in this way.

We also had the opportunity of testing our findings in another project, to set up a shared information space for a research group at our University. We used an information space on the University’s student collaborative learning environment as a low cost solution, but found that it did not integrate the researchers’ normal email tool, not did it enable hyperlinks to be made to specific documents. Without these technical facilities people who wanted to involve their colleagues in using the shared space did not have the tools that they needed. This example with its contradictions and breakdowns is depicted in *Figure 2*, which should be contrasted with the resolved activity shown in the diagram for the pattern FACILITATORS ARE THE KEY.

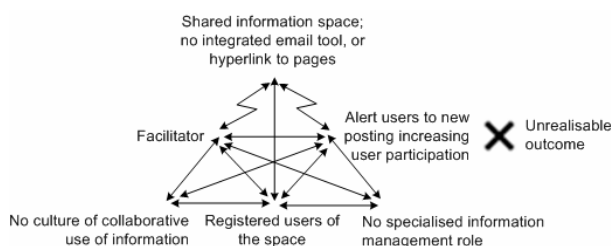


Figure 2. When the facilitator does not have the right tools.

The ranking of patterns by two, one or no asterisks is a way of indicating when a pattern has been validated and the degree of confidence that can be placed in it. Because we have verified the three facilitator patterns in several instances and two different workplaces we rate them fairly highly and feel confident that they should be considered whenever this pattern context is found.

6 - Evolve

Finally, developing a pattern language is evolutionary. In order to ensure that the language keeps pace with change its patterns must be refined, revised, and rejected whenever they are applied as design solutions and verified. New patterns will continually be discovered as activity systems develop and the context of design is changed. Even the application of a pattern will have the effect of changing its context somewhat, and this means that the pattern language will be in a continual state of flux. To cope with this dynamism it is important to adopt tools that support management of changes in the language and even automate the production of a pattern map.

Summary of the Method

This section has described guidelines for the practical application of patterns to the design of activity systems; and for the evolution of a pattern language. Patterns embody the expert knowledge of designers and users, and domain knowledge about how work is carried out in its context. The starting point for those new to patterns is to write patterns based on their own knowledge and experience, and to combine these into an embryo pattern language. Application of patterns involves using the language to generate a subset language specific to the scope of the project; then applying it in design. The techniques of participatory workshops and iterative development, starting with low-cost prototyping techniques that facilitate changes to the specification are particularly appropriate. Finally patterns should be evaluated in use, verified and this knowledge used to develop and update the pattern language.

CONCLUSIONS: VALIDATION OF THE APPROACH

It is proposed that the method of ‘activity patterns’ is a versatile tool that could be used throughout the cycles of use, evaluation and design (Bannon, 1996); as a participatory technique bridging the conceptual and semantic gap between developers and users; as a tool for establishing shared meanings in a multi-functional or distributed project team; to represent work and its organisational and technical interfaces; and to design future mediated, collective work. This approach was the outcome of a program of fieldwork, with the patterns being identified through an evaluation of information sharing tools, as use evolved over a period of time and new practices emerged. It can therefore be stated with some confidence that capturing the lessons of evaluation as patterns, in order to use these to inform future design, is a useful and valid approach. It has not been possible to test

the method at GreenFam, as the research project has now ended.

The appeal of appropriating a tool, rather than designing something completely novel, is that it is possible to benefit from existing resources and a community of users. Developers are not being asked to adopt a completely new method that is not yet part of the practice of software development. With other pattern user-developers in the domains of SE, HCI and participatory design, it is possible to refer to their experiences to validate the approach. However, it must be acknowledged that little evidence exists about how – or indeed whether – patterns are being used in real-world system development practice. Many of the books which are currently being published describe patterns as ‘good design’ heuristics, distilled from professional experience, without reflecting on the processes of developing and applying patterns (for example Adolph *et al*, 2003; Graham, 2003).

The patterns approach outlined in this paper proposes a novel application for patterns - to model computer-supported cooperative work at the level of activity and actions. Much of the recent interest in patterns has been at the level of interaction and software design, although a wider role in socio-technical design has been proposed by some authors (Erickson, 2000a; Martin *et al*, 2001; Herrmann *et al*, 2003). The approach is therefore still largely untested and needs to be trialed and refined in an industrial setting in order to evaluate its potential.

ACKNOWLEDGMENTS

The research was funded by the Engineering and Physical Sciences Research Council, EPSRC Award Reference 95306394 (1995-1998) and the Interactive Technologies Research Group at the University of Brighton (2001-2002). Thanks to all at GreenFam who contributed to the study, particularly Robin van Koert, my collaborator on the evaluation research. Bob Seago, the University of Brighton photographer, took the photos that illustrate the patterns. Thanks also to the workshop participants for their feedback, and especially to Lars Taxén for his helpful reviews.

REFERENCES

- Adolph, S., P. Bramble, A. Cockburn and A. Pols, (2003). *Patterns for Effective Use Cases*. Boston, MA: Addison-Wesley.
- Alexander, C., M. Silverstein, S. Angel, S. Ishikawa and D. Abrams, (1975). *The Oregon Experiment*. New York: Oxford University Press.
- Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King and S. Angel, (1977). *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press.
- Alexander, C., (1979). *The Timeless Way of Building*. New York: Oxford University Press.
- Alexander, C., H. Davis, J. Martinez and D. Corner, (1985). *The Production of Houses*. New York: Oxford University Press.
- Bannon, L., (1996). Use, Design and Evaluation: Steps Towards an Integration. In: Shapiro, D., M. Tauber and R. Traummuller, Eds., *The Design of Computer Supported Cooperative Work and Groupware Systems*. Amsterdam: Elsevier. 423-443.
- Bannon, L. and S. Bodker, (1997). Constructing Common Information Spaces. In: Hughes, J., W. Prinz, T. Rodden and K. Schmidt, Eds., *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work*. ECSCW '97, Lancaster, England. Dordrecht: Kluwer Academic Publishers. 81-96.
- Bertelsen, O. and S. Bodker, (2002). Discontinuities. In: Floyd, C., Y. Dittrich and R. Klischewski, Eds., *Social Thinking, Software practice*. Cambridge, MA: MIT Press. 409-424.
- Bertelsen, O. and S. Bodker, (2003). Activity Theory. In: Carroll, J. M., Ed. *HCI Models, Theories and Frameworks: Towards an Interdisciplinary Science*. San Francisco, CA: Morgan Kaufmann Publishers. 291-314.
- Bodker, S. and E. Christiansen, (1997). Scenarios as Springboards in CSCW Design. In: Bowker, G., S. L. Star, W. Turner and L. Gasser, Eds., *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*. Mahwah, NJ: Lawrence Erlbaum Associates. 217-233.
- Borchers, J., (2001a). *A Pattern Approach to Interaction Design*. Chichester: John Wiley and Sons.
- Borchers, J., (2001b). A Pattern Approach to Interaction Design. *AI & Society* 15 (4): 359-376.
- Carroll, J. M., (2000). *Making Use: Scenario-Based Design of Human-Computer Interaction*. Cambridge, MA: The MIT Press.
- Checkland, P., (1981). *Systems Thinking, Systems Practice*. Chichester: John Wiley and Sons.
- Checkland, Peter and Jim Scholes, (1996). *Soft Systems Methodology in Action*. Chichester: John Wiley and Sons.
- Dearden, A., J. Finlay, E. Allgar and B. McManus, (2002). Using Pattern Languages in Participatory Design. In: Binder, T., J. Gregory and I. Wagner, Eds., *Proceedings of the Participatory Design Conference*. PDC 2002, Malmo, Sweden. Palo Alto, CA: CPSR. 104-113.
- Engestrom, Y., (1987). *Learning by Expanding: An Activity-Theoretical Approach to Developmental Work Research*. Helsinki: Orienta-Konsultit Oy.
- Erickson, T., (2000a). Supporting Interdisciplinary Design: Towards Pattern Languages for Workplaces. In: Luff, P., J. Hindmarsh and C. Heath, Eds., *Workplace Studies: Recovering Work Practice and Informing System Design*. Cambridge: Cambridge University Press. 252-261.

- Erickson, T., (2000b). *Lingua Francas for Design: Sacred Places and Pattern Languages*. In: Boyarski, D. and W. Kellogg, Eds., *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. DIS 2000, Brooklyn, NY. New York: ACM Press. 357-368.
- Gamma, E., R. Helm, R. Johnson and J. Vilissides, (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.
- Graham, I., (2003). *A Pattern Language for Web Usability*. London: Addison-Wesley.
- Grudin, J., (1988). Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces. In: *Proceedings of the Conference on Computer-Supported Cooperative Work*. CSCW '88, Portland, Oregon. New York: ACM Press. 85-93.
- Guy, E. S., (in preparation). *Patterns of Activity: Modelling Computer Supported Cooperative Work*. Unpublished PhD thesis. School of Computing, Mathematical and Information Sciences, University of Brighton, UK.
- Harris, S. R., (2004). Morphological Analysis of HCI Video data Using Activity Theory. In: *HCI 2004: Design for Life*. HCI 2004, Leeds, UK. British HCI Group. 41-44.
- Herrmann, T., M. Hoffmann, I. Jahnke, A. Kienle, G. Kunau, K-U. Loser and N. Menold, (2003). Concepts for Usable Patterns of Groupware Applications. In: Pendergast, M., K. Schmidt, C. Simone and M. Tremaine, Eds., *Proceedings of the 2003 International SIGGROUP Conference on Supporting Cooperative Work*. Florida, USA. New York: ACM Press. 349-358.
- Leont'ev, A. N., (1979). The Problem of Activity in Psychology. In: Wertsch, J. V., Ed. *The Concept of Activity in Soviet Psychology*. New York: M. E. Sharp Inc. 37-71.
- Leont'ev, A. N., (1981). *Problems of the Development of the Mind*. Moscow: Progress Publishers.
- Leont'ev, A.N., (1978). *Activity, Consciousness and Personality*. New Jersey: Prentice-Hall Inc.
- Lilienfeld, R., (1978). *The Rise of Systems Theory: An Ideological Analysis*. Chichester: John Wiley and Sons.
- Martin, D., T. Rodden, M. Rouncefield, I. Sommerville and S. Viller, (2001). Finding Patterns in the Fieldwork. In: Prinz, W., M. Jarke, Y. Rogers, K. Schmidt and V. Wulf, Eds., *Proceedings of the Seventh European Conference on Computer-Supported Cooperative Work*. ECSCW 2001, Bonn, Germany. Dordrecht: Kluwer Academic Publishers. 39-57.
- Nardi, B., Ed. (1996). *Context and Consciousness: Activity Theory and Human-Computer Interaction*. Cambridge, MA: MIT Press.
- van Duyne, D., J. Landay and J. Hong, (2003). *The Design of Sites: Principles, Processes and Patterns for Crafting a Customer-centered Web Experience*. Boston, MA: Addison-Wesley.
- Vygotsky, L. S., (1962). *Thought and Language*. Cambridge, MA: MIT Press.
- Wartofsky, M. W., (1979). *Models: Representation and the Scientific Understanding*. Boston: D. Reidel Publishing Company.

Appendix: Four Patterns For Information Sharing

In order to illustrate how to write patterns and integrate them into a pattern language, we have chosen a subset of four patterns from the GreenFam pattern language. These are represented in the pattern map shown in Figure 3, along with some of the other patterns to which they are related: the illustrated patterns are highlighted in grey. The arrows show the associations between patterns, with the arrow pointing from the higher level to the lower level pattern. The top level pattern is INFORMATION AS COMMON PROPERTY. The template for writing patterns is shown in Table II. This is based on Alexander's pattern form, with the changes that have been discussed in the paper. This is followed by the four patterns –

- INFORMATION AS COMMON PROPERTY
- FACILITATORS ARE THE KEY
- EMAIL ALERTS
- ONE CLICK HYPERLINK

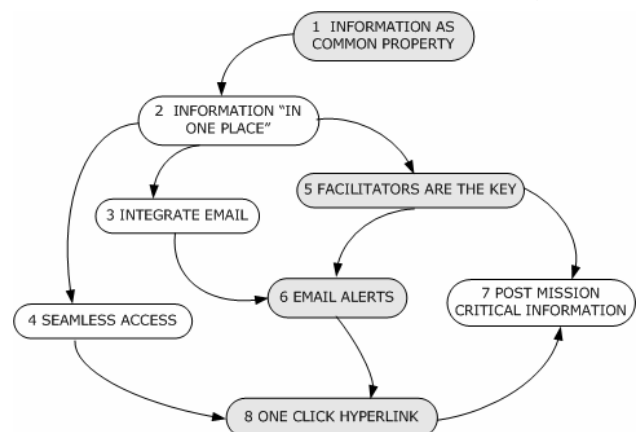


Figure 3. Subset of the pattern language highlighting the illustrated patterns.

Description of content	Formatting instructions
<Identifier> <PATTERN NAME> <Validity ranking>	Numeric identifier; pattern name in small capitals; if ranked use 1 or 2 asterisks * / **. Centre text.
<Illustration of a concrete instantiation of the pattern>	
...<introduction: sets the <i>context</i> of higher level patterns which this pattern helps to complete>	The introduction begins with ellipsis marks ... followed by the first word in lowercase. Higher level pattern names are in small capitals followed by the pattern identifier in brackets e.g. FACILITATORS ARE THE KEY (4). Justify margins.
❖ ❖ ❖	Symbols divide the introduction from the contradiction summary: centre.
<Concise summary of the contradiction>	Bold ; justify; indent first line.
<Detailed description of the contradiction that the pattern resolves, with empirical examples of how it can be manifested. May include illustrations of examples and counter-examples.>	Justify; indent first line of the new paragraph.
Therefore:	Indent.
<Resolution: expressed in the form of an instruction.>	Bold ; justify; indent first line.
<Model of the resolution in an appropriate diagrammatic form>	Centre.
❖ ❖ ❖	Symbols indicate the end of the body of the pattern: centre.
<End: <i>references</i> to the lower level patterns which help to complete this pattern> ...	Lower level pattern names are in small capitals followed by the pattern identifier in brackets e.g. EMAIL ALERTS (7). Ends with ellipsis marks ... Justify; indent first line.

Table II. Pattern template based on Alexander's model.

1 INFORMATION AS COMMON PROPERTY



³Many organisations that work intensively with information have the goal of developing a culture and practice of treating information as common property, rather than as the private resource of departments and individuals. To achieve this goal they have to change attitudes and established ways of working, and introduce new collaborative tools to support information sharing.

³ As a top level pattern INFORMATION AS COMMON PROPERTY starts with the contradiction summary rather than a list of related higher level patterns.

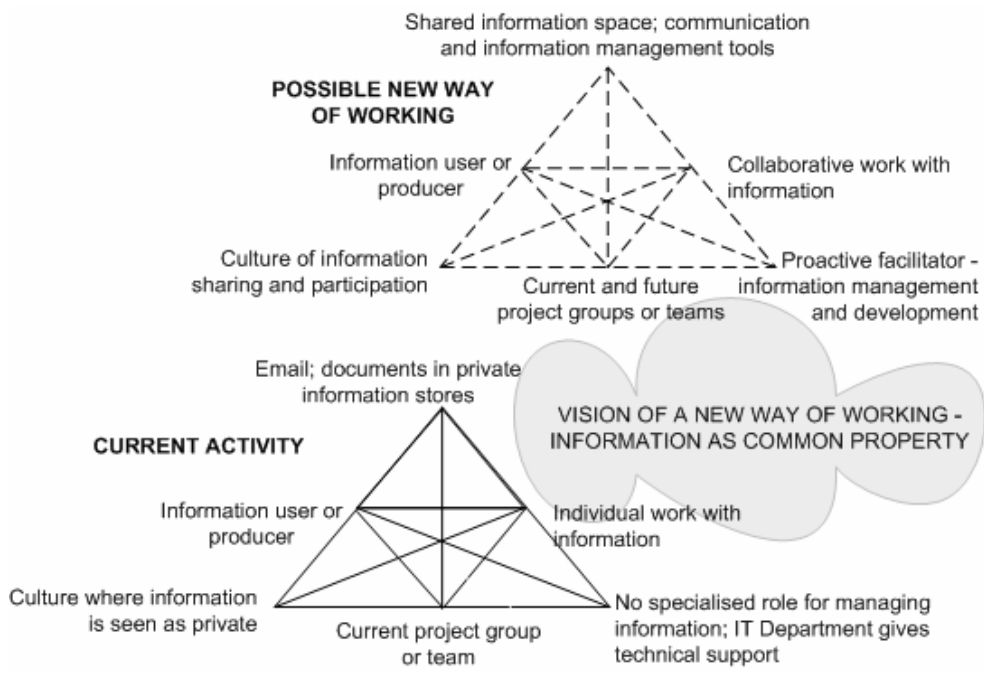
In many organisations a practice of treating information as the private resource of the organisational unit or individual who ‘owns’ it prevails. This prevents the sharing of information freely among everybody who needs access to it in order to carry out their work. The practice might have been established over a number of years and will be supported by a culture which rationalises it in terms of the need for control to prevent unauthorised access, and to maintain the accuracy and consistency of information.

In information intensive organisations the huge amount of information that has to be dealt with – articulated as ‘information overload’ – may make the people who work there reluctant to share information, or to use shared information spaces. Their immediate reaction to a proposal that information is treated as a shared resource might be that this will overload them with even more information and work from which they will not benefit.

Computer tools currently in use for communicating and managing information may not facilitate information sharing, and may fragment it into many private spaces. One example is email where messages are kept in private mailboxes, rather than being stored in a shared archive. Another is personal computers, where information resources are created, stored and managed on the computer’s hard drive: there is an overhead of additional work required to place these resources in a shared space. Another example is the group practice of putting document files in shared folders without proper tools for information management, where it may be difficult to search and retrieve information.

Therefore:

Put information in shared spaces with integrated communication and information management tools, where people can work collaboratively as well as carrying out their own work. Take steps to change organisational culture by providing incentives for information sharing.



The change to an information sharing culture and practice



To facilitate collaborative work and encourage an information sharing culture put INFORMATION “IN ONE PLACE” (2) ...



... cooperative work involving many people working in different locations and times is often coordinated through the use of shared information spaces – a practice that treats INFORMATION AS COMMON PROPERTY (1). By putting INFORMATION “IN ONE PLACE” (2) people have access to all the information resources they need to do a job.



Users often do not readily adopt new tools for information sharing when placing information in common involves additional work and means they must change the way they work.

When new collaborative information sharing tools are introduced to workplaces they will require changes in existing ways of working. For example –

- A new organisational policy may mandate that information is treated as a shared resource, while departments, work groups or individuals have been used to keeping their information resources to themselves.
- The person doing the additional work of posting information in the space is often not the person who directly benefits from using it (Grudin, 1988), so that busy users have no incentive to participate.
- When new tools for collaborative information sharing are implemented they will have to compete with tried and tested ways of working with information such as email, and other communicative practices which are embedded in the culture of the workplace.

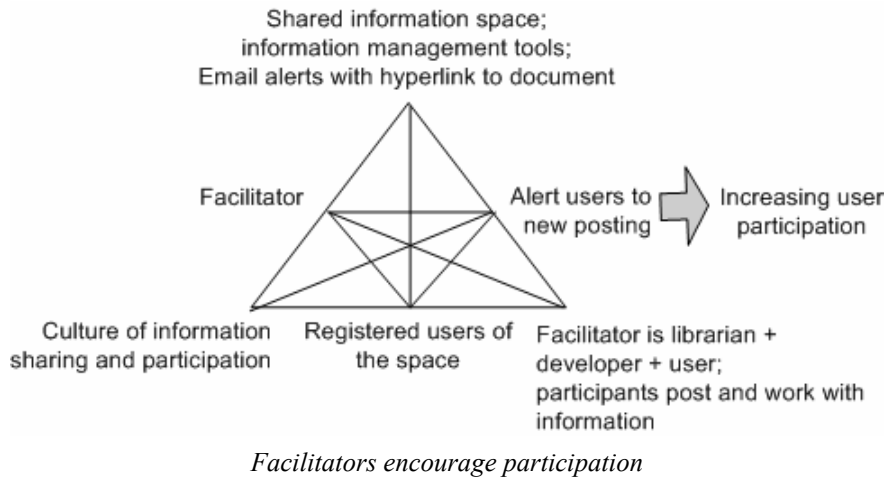
Where information spaces are adopted and used as intended it is often because an enthusiastic and motivated individual volunteers to take on the role of facilitation, encouraging the adoption of the new tool in their workgroup. They may do this by posting important working documents in the new information space, whether instead of, or in addition to, communicating them through established modes; by sending an email to the group alerting them to the new document on the database and directing them to it by means of a hyperlink; by organising the space – archiving out of date documents, evolving use-centred classification schemas, and repairing mis-classified information; by encouraging and exhorting the group to use the information space and thereby nurturing the emergence of a new culture of information sharing.

Where a facilitator does not emerge the space may never be used effectively, as the workgroup continues to work in the old ways. This will also happen if the facilitator does not have the tools to carry out their role, or the right skills. Individuals who take on this role voluntarily may quickly become demoralised if the workgroup does not begin to use the space.

The new role of facilitator requires additional work, for example communicating the same information in different ways and sending email alerts to users. In order to do this effectively the facilitator needs the right tools for the job.

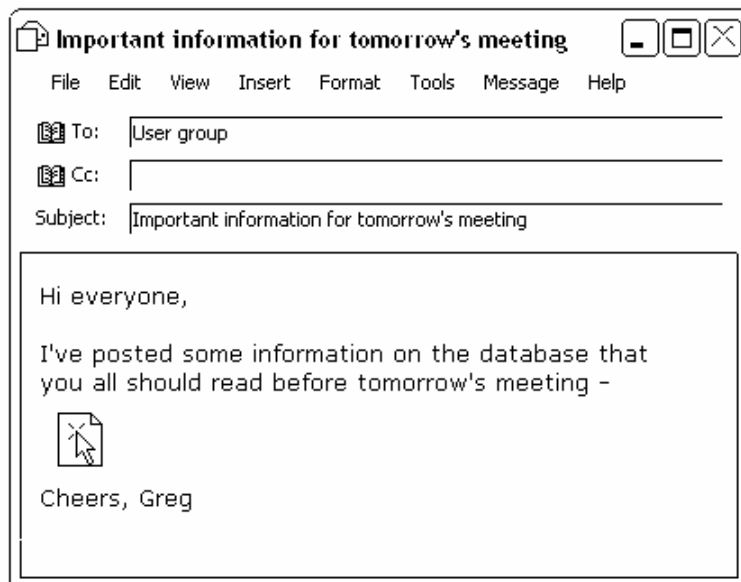
Therefore:

Appoint a facilitator for the database and ensure that this role is recognised as a part of their job and valued. The facilitator must be allocated the necessary time to do the job properly and given training in the technical and information management skills required. They will be responsible for seeing that the objectives for the space are achieved; encouraging participation by registered users; and for information management and development of the space. The information space should have integrated tools that facilitators need to do the job.



Facilitators alert users to new material they have posted by means of EMAIL ALERTS (6), which draw them into the space and open the required document by means of a ONE CLICK HYPERLINK (8). Facilitators POST MISSION CRITICAL INFORMATION (7) that everybody needs to read to carry out their work – such as agendas for meetings – so that users have to respond to the email alert and visit the space ...

6 EMAIL ALERTS **



... Where organisations want to change from an email paradigm to a shared information paradigm and put INFORMATION “IN ONE PLACE” (2) with tools which INTEGRATE EMAIL (3), FACILITATORS ARE THE KEY (5) to encouraging users to adopt the new way of working.



It can be difficult to get users to change from using email to communicate information, to placing it in a shared information space where everybody can access it. Facilitators need the right tools to involve users and encourage participation.

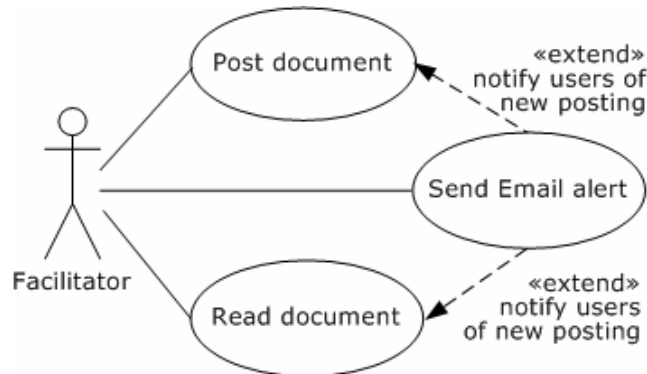
Email is a tool used to coordinate work and communicate information that is generally popular with users. It has the drawback that email messages are stored in personal mailboxes where the information cannot be shared. When new tools for collaborative information sharing are introduced to organizations they will have to compete with tried and tested ways of communication such as email, which are embedded in the culture and practice of the workplace and which are easy and quick to use.

If email is integrated in the information space the facilitator can easily email users after she has posted a new document in the space, especially if it contains mission critical information that they need to read. Or maybe she has just read an interesting document that someone else has posted and wants to draw it to the attention of one or more users who may not yet have got into the habit of checking the space regularly for new information.

Users who are accustomed to using email in their day to day work will read the message and, by means of the hyperlink to the document, are transported directly to the shared information space. By the means of email alerts users are accustomed to the space, which gradually becomes more widely used.

Therefore:

Make a virtue of the popularity of email by using it to alert users about new information on the database and directing them to it by means of a hyperlink.



Use cases for EMAIL ALERTS



The email message contains a ONE CLICK HYPERLINK (8): a mouse click on the hyperlink icon takes the user directly to the relevant document in the database ...

8 ONE CLICK HYPERLINK**



... the facilitator of the shared information space sends EMAIL ALERTS (6) to users, which contain a hyperlink to a document posted in the space. Users must automatically be logged on to the space when they log onto their computer, so that there is SEAMLESS ACCESS (4) to the space when they click on the link.

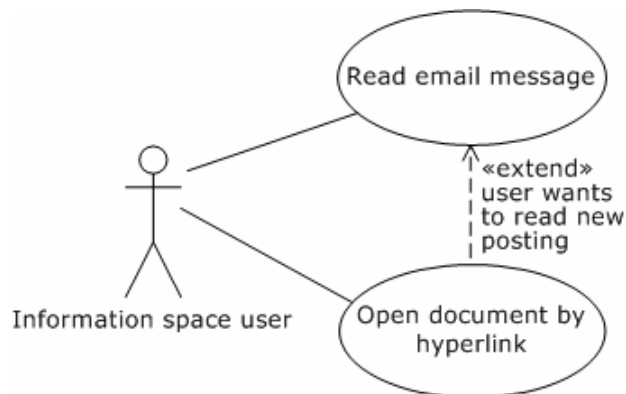


It can be difficult to get people to adopt a new tool such as a shared information space in their daily work, if it requires them changing established ways of working.

It can take along time before users get into the habit of using a new tool, particularly when they already have tools such as email that are established in their daily routine and do not see immediate benefits of changing the way that they work. If the space is perceived as being “yet another software tool” that they are supposed to learn how to use they may not log onto the space every day to check for new information. One of the jobs of the facilitator is to get users into the habit of visiting the space regularly and to make them familiar with it: this is a first step towards encouraging active participation. A hyperlink in an email alert message can be used to direct users to a document in the shared space. That way they do not have to remember to log into the space and check it each day – by one click use of the space becomes seamless with reading an email.

Therefore:

Put a hyperlink to a document in the shared information space into an email alert, so that when the user reads the message they can click on the link and be transported directly to the space.



Use cases for ONE CLICK HYPERLINK



As a further incentive to use the space the facilitator should POST MISSION CRITICAL INFORMATION (10) and email users a hyperlink to the document ...