

Agile Environments – Some Patterns for Agile Software Development Facilitation

Klaus Marius Hansen

*ISIS Katrinebjerg
University of Aarhus
Aabogade 34
DK-8200 Aarhus N*

Abstract

Agile software development demands an agile working environment in contrast to traditional, specification-oriented software development. Pair programming, continuous user involvement, and ad-hoc meetings are examples of work practices that require facilitation for new ways of working. We present patterns that facilitate the creation and evolution of a working environment for agile development.

1 Introduction

Many system development projects are facing uncertainty with respect to which solutions to create, how to create these solutions, or even which problem to solve. Traditional prescriptive and specification-oriented development processes such as the early object-oriented development methods (Booch, 1991), (Jacobson et al., 1992), (Rumbaugh et al., 1991) fail to address such commonly occurring project situations.

There have been continuous reactions to such (earlier and later) processes starting with the early participatory design work (Nygaard and Berge, 1974), going over Joint Application Development (Wood and Silver, 1995), to current methods and processes such as Adaptive Software Development (Highsmith, 2000), Crystal (Cockburn, 2001), Dynamic System Development Method (Stapleton, 1997), Extreme Programming (XP; (Beck, 1999)), Feature-Driven Development (Coad et al., 1999), and SCRUM (Schwaber et al., 2002). These processes share a number of

Email address: marius@daimi.au.dk (Klaus Marius Hansen).

characteristics: They aim at providing working software in uncertain and complex situations through a focus on, e.g., iterative development, incremental addition of functionality, and active stakeholder involvement.

Based on the observation of such common characteristics, the agile manifesto (Beck et al., 2001) establishes four fundamental values for agile software development:

- *individuals and interactions* over processes and tools,
- *working software* over comprehensive documentation,
- *customer collaboration* over contract negotiation, and
- *responding to change* over following a plan.

The environment, physical and virtual, in which agile software development is performed is important in that it may contribute significantly to working with these values: A work environment that favors personal communication may help facilitate individuals and interactions, the physical environment may help developers focus on developing thus aiding in making working software, locating a development team close to a customer site may enhance customer collaboration, flexible arrangements of project rooms may help in responding to change by allowing dynamic reconfiguration. Since the principles of agile software development go back to the early days of participatory design and since the work environment may make a significant difference to agile software development projects, it makes sense to capture and express the experience of agile software development facilitation, i.e., the practices of helping bring about agile software development easier, through. Thus, this paper tries to do that through the formulation of a set of patterns for agile software development facilitation.

1.1 Form

The major part of this paper consists of a set of patterns for agile software development facilitation. They are written for software developers and managers, a distinction that may become blurred, however, in agile software development.

For the description of patterns, we use a slight variation of the narrative Portland Form (Cunningham, 2002): Each pattern name is followed by a problem and a set of associated forces for that problem. The solution to the problem that the pattern represents is started by a *Therefore:* and a short solution. Finally, the resulting context is discussed. References to patterns are shown in italics.

The last part of this paper consists of a problem/solution summary of the presented patterns.

2 Patterns for Agile Environments

The following patterns provide a set of starting points for relating experience in agile software development facilitation. Currently, the patterns are divided into four categories:

- *Context* encompasses patterns that are not specific to facilitation within agile software development, but are prerequisites for agile facilitation.
- *Place* concerns social properties of locations: One location may be used for concerts during the week and church ceremonies on Sundays yielding different places (Harrison and Dourish, 1996). The patterns in this category are then concerned with facilitating social aspects of the physical development environment.
- *Space* is concerned with the physical properties of locations: How much room is there, how is the lighting, where is the furniture placed etc (Harrison and Dourish, 1996).
- *Detail* contains facilitation patterns that are additions to the Space and Place patterns.

Within each category, the patterns are organized according to importance.

2.1 Context

2.1.1 *Team Chooses*

You need to provide the best working environment for your development team. Each project is unique. Your team is motivated and dedicated since they will be working in an agile manner, moreover you may be working with a process (such as XP) that requires and should maintain high discipline.

Agile software development favors individuals and interactions.

Therefore: The team decides how their work environment should be and sets rules for how development should be facilitated.

This pattern is at the core of agile software development: responding to change requires that individuals are able to make changes as required.

With respect to facilitation, this means that the team - including a project manager and possibly a customer - should be able to decide how to balance the forces of the subsequent patterns in this collection. If your process requires high discipline (such as XP) this is not in contradiction to that. This pattern just acknowledges that

project members are competent professionals and allow them to inform their own work situation.

The agile values should constrain what the team is able to decide. Focus on individuals and interactions, working software, customer collaboration, and responding to change is mandatory and decisions on facilitation should not compromise this. Having the *Customer Close By* (Section 2.2.2) or *One War; One Room* (Section 2.2.1) are examples of patterns that are almost required when applicable.

2.2 Place

2.2.1 *One War; One Room*

Developers and stakeholders on your project need to communicate as effectively and efficiently as possible. The frequency of and need for communicating is high. The team is small.

Therefore: Place developers in the same room. Also put *Customer Close By* (Section 2.2.2).

For large projects it is impractical to place all developers in the same room, but for smaller projects (at least up to ten developers), it maximizes the potential face-to-face communication and awareness. This arrangement is also ideal for pair programming. If corporate culture makes it impossible to have such a room, creating a persistent design room (Beyer and Holtzblatt, 1997) in which design meetings and, e.g., *Writing on the Wall* (Section 2.2.3) may take place, may serve as a compromise. Geographically separated teams makes facilitation more difficult.

Even if the team is heterogeneous through having, e.g., object-oriented developers, ethnographers, usability experts, and customers on it, co-location may work very well (Christensen et al., 1998).

Care must be taken in also providing space for more private activities, such as reading and phone calls through, e.g., *Public and Private Spaces* (Section 2.3.2).

2.2.2 *Customer Close By*

Stakeholder requirements are initially vague and are constantly prone to change. You need the development team to discover and keep in sync with requirements. The customers needs to know the progress of the team.

Therefore: Place the customer and developers close by and make room for that.

If customers are co-located with, or readily accessible to, the development team, there should be a workspace for the customer for doing ordinary work. The development team should have access to the customer at all times. XP advocates this kind of arrangement.

A complimentary approach is to locate the development team within the customer organization. If this is possible, this is more effective in terms of learning about and designing for the practice of the customer (Christensen et al., 1998). DSDM advocates this kind of arrangement.

This kind of stakeholder involvement in the project will probably not suffice. It may, e.g., be beneficial to include ethnographic studies of real work when dealing with complex problem domains (Christensen et al., 1998).

The nature of the project may make customer co-location difficult: for contract development it may be hard economically to convince the contractor of this kind of arrangement. For in-house development co-location may be easier, in particular if provisions can be made for the customer to work effectively on her day-to-day work while located near the development team. For off-the-shelf products, it may be impossible to be co-located with customers.

2.2.3 *Writing on the Wall*

Developers and customers should have easy access to writing material for discussions, thinking, and visualization. You want to maintain awareness and access to the results. You want to capture ideas and suggestions as they evolve.

Therefore: Place whiteboards and paper for writing in development rooms.

Make sure you have dedicated space for these activities so that it is possible to quickly go to and from such discussions without interrupting the work of others.

For whiteboards, you will need plenty of space for sketching and modelling. Also, if you have *Space for Pairs* (Section 2.3.4) you may consider having a whiteboard for each pair. An option is also to use brown paper or even whiteboard wallpaper to cover an entire project room with writing surface...

Mynatt (1999) investigated the use of whiteboards in a personal and a public office setting. She points to four characteristics of office whiteboard use:

- (1) Whiteboards are effective for thinking and pre-production tasks: ideas and thoughts such as how a graphical user interface should be designed or which responsibilities a class should have can quickly and easily be written or erased.
- (2) Use of whiteboards leads to clusters of persistent and short-lived content: most of what is written on a whiteboard is really ephemeral, but some content such

as to do list persist for much longer.

- (3) Whiteboards contain everyday content as well as formal drawings.
- (4) Whiteboards are and can be used for information that ranges from being semi-public to private.

An extensive use of whiteboards or other analog writing material may cause problems, however: content cannot be saved or restored and can only be manipulated in very primitive ways. For saving, and partly for restoring, a digital camera may help. It is common practice to capture contents of whiteboards using such means and later use the images for recall or for more formal documentation (Andersen et al., 2000). If you frequently need to update your content, you will either need massive amounts of whiteboard space or these tools may be too *Simple Artefacts* (Section 2.2.4). Another problem is that whiteboard and paper are not good at documenting precise decisions due to their ephemeral character. Also, although these materials are better at representing the development of a solution, or discarded proposals, they are not good at capturing rationales.

Using electronic whiteboards can help overcome some of the weaknesses of analog whiteboards. Electronic whiteboards capture the pen input made by users and transfers this input to a computer. There are two major variants of electronic whiteboards: Whiteboard replacements, such as the SMART Board (<http://www.smart-tech.com>) are complete, standalone replacements of traditional whiteboards. They are typically combined with a computer and a projector so that a computer image may be projected onto their screens. These technologies are expensive and mostly applicable if the writing requires a lot of interactivity. Whiteboard augmenters, such as the Mimio (<http://www.mimio.com>), can be plugged onto any existing whiteboard after which they can capture what the user draws on the whiteboard if the special pens are used. They can be used either in non-projected or projected mode. In non-projected mode, the whiteboard acts just as an ordinary whiteboard, but enables the saving of drawings on the whiteboard. In projected mode, the whiteboard augmenters work much like the electronic whiteboard replacements. Whiteboard augmenters are much less expensive than whiteboard replacements and enable the reuse of existing whiteboards. They are, however, more problematic to set up, particularly in projected mode. In either case, if project mode is to be used, consider using *Simple Artefacts* (Section 2.2.4) in this mode.

Ambler (2002) introduces the Agile Modeling methodology which has a number of detailed suggestions for supporting object-oriented modelling in an agile manner by organizing project rooms with plenty of simple, readily available, and flexible writing surfaces and writing material. Concretely, Ambler advocates that a room to be used for modelling should contain dedicated space, significant whiteboard space, a digital camera, modelling supplies, a bookshelf or storage cabinet, a large table, a computer, chairs, wall space to attach paper, a projector, reference books, food, and toys. In general, Ambler stresses the use of simple and flexible tools, such as whiteboard and paper, over more complex tools, such as CASE tools. This

works well for simple situations that require more writing than reading; if there is need for extending previous models or using previous models for reference, it becomes problematic to use just analog material. For this Ambler recommends having a project, a computer with a CASE tool, and a "CASE jockey" who operates the CASE tool to capture changes drawn on the whiteboard and to show previous models. Electronic whiteboards, as introduced above, may improve on this situation and make the choice between of tools much more flexible.

2.2.4 *Simple Artefacts*

You want to make sure developers focus on producing working software instead of, e.g., just working with software. Using Complex artefacts used in software production may detract the developers' attention from this.

Therefore: Provide and use simple artefacts, physically as well as virtually. The simplest tool that solves a problem should be used.

Index cards are ubiquitous tools also in agile development processes. They are cheap, readily available and sufficiently simple as not to detract developers from their primary focus.

Whiteboards and paper, e.g., for *Writing on the Wall* (Section 2.2.3), are simple tools that are ideal for collaboration and communication. For some situations these may be too simple, cf. *Writing on the Wall* (Section 2.2.3).

Use virtual tools that fit your work processes as closely as possible, preferably maximizing support for communication, feedback, and customer collaboration with a focus on working software. For modelling, Ideogramic UML (Damm et al., 2000), which combines electronic whiteboards and computer-aided software engineering, may be ideal. For programming, an Integrated Development Environment (IDE) that is flexible and allows you to combine just the needed components, may be ideal.

It may be argued that this pattern is actually the XP value of Simplicity used on facilitation.

2.3 Space

2.3.1 *Flexible Furniture*

You want *Space for Pairs* (Section 2.3.4) and *Space for Groups* (Section 2.3.3). Ordinary office equipment and arrangements makes flexible adaptation of spaces for these purposes difficult.

Therefore: Provide flexible furniture and allow developers to rearrange it through *Team Chooses* (Section 2.1.1).

Chairs should be movable so that people can pair and participate in group meetings. You might also want space for "standing" meetings (as advocated by DSDM).

Tables with adjustable legs are good for flexible pairing when programming.

Artefacts used for *Writing on the Wall* (Section 2.2.3) should also preferably be movable.

2.3.2 *Public and Private Spaces*

You want to facilitate group and pair communication while still catering for the needs for private communication and thoughts. You may have *One War, One Room* (Section 2.2.1).

Therefore: Provide a mix of public and private spaces.

These spaces should intermix with *Space for Groups* (Section 2.3.3) and *Space for Pairs* (Section 2.3.4).

An effective arrangement in XP has been to provide small, semi-private bull-pens along one wall of the room and have the middle of the room be common.

Often, if this is not provided, developers tend to create their own, virtual privacy by, e.g., listening to music using headphones. This practice hampers communication within the team.

2.3.3 *Space for Groups*

Communication with customers and brainstorming among developers is essential. You want to provide awareness of such (ad-hoc) meetings to all project members.

Therefore: Make space for groups. Make the space visible to all.

Meeting with an on-site customer, for, e.g., the planning game of XP, may be one of the reasons for having Space for Groups. This space can be combined with *Customer Close By* (Section 2.2.2). Awareness of results of group work may be made through *Writing on the Wall* (Section 2.2.3).

2.3.4 *Space for Pairs*

Developers need to communicate, collaborate, and coordinate effectively also in the small. Private offices or inflexible personal office space makes hampers this.

Therefore: Make room for pairs so that pair discussions and pair programming are facilitated.

This may involve having Flexible Furniture and is facilitated by having *One War, One Room* (Section 2.2.1).

2.4 **Detail**

2.4.1 *Perks*

Morale is high because of *Team Chooses* (Section 2.1.1) and the agile work process. You want to keep morale high and you want to support lateral thinking in the project group.

Therefore: Give developers perks in the form of food and toys.

Create a place in which it is possible and welcome to relax and where it is possible to eat food, drink coffee, play with toys, read magazines etc. The team should make sure they do not spend too much time on this.

3 Problem/Solution Summaries

<i>Problem</i>	<i>Solution</i>	<i>Pattern Name</i>
You need to provide the best working environment for your development team. Each project is unique	The team decides how their work environment should be and sets rules for how development should be facilitated	<i>Team Chooses</i>
Developers and stakeholders on your project need to communicate as effectively and efficiently as possible	Place developers in the same room	<i>One War, One Room</i>
Stakeholder requirements are initially vague and are constantly prone to change	Place the customer and developers close by and make room for that	<i>Customer Close By</i>
Developers and customers should have immediate access to writing material to be used for discussions and for thinking	Place whiteboards and paper for writing in development rooms	<i>Writing on the Wall</i>
You want to make sure developers focus on producing working software instead of, e.g., just working with software	The simplest tool that solves a problem should be used	<i>Simple Artefacts</i>
You want <i>Space for Pairs</i> , <i>Space for Groups</i> , and want to allow for adaptation to the current project situation	Provide flexible furniture and allow developers to rearrange it through Team Chooses	<i>Flexible Furniture</i>
You want to facilitate group and pair communication while still catering for the needs for private communication and thoughts	Provide a mix of public and private spaces	<i>Public and Private Spaces</i>
Communication with customers and brainstorming among developers is essential	Make space for groups. Make the space visible to all	<i>Space for Groups</i>
Developers need to communicate, collaborate, and coordinate effectively also in the small	Make room for pairs so that pair discussions and pair programming are facilitated	<i>Space for Pairs</i>
You want to keep moral high and you want to support lateral thinking in the project group	Give developers perks in the form of food and toys	<i>Perks</i>

References

- Ambler, S. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. Wiley.
- Andersen, C., Hansen, K., Sandvad, E., Thomsen, M., and Tyrsted, M. (2000). Tool support for iterative system development activities: Issues and experiences. In *Proceedings of NWPER'2000*, pages 1–21.
- Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, R., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for agile software development. <http://www.agilemanifesto.org>.
- Beyer, H. and Holtzblatt, K. (1997). *Contextual Design: A Customer-Centered Approach to Systems Designs*. Academic Press/Morgan Kaufmann.
- Booch, G. (1991). *Object-Oriented Design with Applications*. Benjamin/Cummings.
- Christensen, M., Crabtree, A., Damm, C., Hansen, K., Madsen, O., Marqvardsen, P., Mogensen, P., Sandvad, E., Sloth, L., and Thomsen, M. (1998). The M.A.D. experience: Multiperspective Application Development in evolutionary prototyping. In Jul, E., editor, *ECOOP'98 – Object-Oriented Programming. Proceedings of the 12th European Conference*, pages 13–40. Springer Verlag.
- Coad, P., Lefebvre, E., and de Luca, J. (1999). *Java Modeling in Color with UML*. Wiley.
- Cockburn, A. (2001). *Agile Software Development: Software Through People*. Addison-Wesley.
- Cunningham, W. (2002). About the Portland Form. <http://c2.com/ppr/about/portland.html>.
- Damm, C., Hansen, K., and Thomsen, M. (2000). Tool support for object-oriented cooperative design: Gesture-based modeling on an electronic whiteboard. In *Proceedings of CHI 2000, ACM Conference on Human Factors in Computing Systems*, pages 518–525.
- Harrison, S. and Dourish, P. (1996). Re-place-ing space: the roles of place and space in collaborative systems. In *Proceedings of CSCW'1996, Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 67–76.
- Highsmith, J. (2000). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House.
- Jacobson, I., Christerson, M., Jonsson, P., and Övergaard, G. (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press.
- Mynatt, E. (1999). The writing on the wall. In *Proceedings of INTERACT'99*, pages 196–204.
- Nygaard, K. and Bergh, O. (1974). *Planning, Control and Data Handling: Textbook for the Trade Unions. Part 1 Initiation. (In Norwegian)*. Tiden Norsk Forlag.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Loresen, W. (1991).

- Object-Oriented Modeling and Design*. Prentice Hall.
- Schwaber, K., Beedle, M., and Martin, R. C. (2002). *Agile Software Development with SCRUM*. Prentice Hall.
- Stapleton, J. (1997). *DSDM Dynamic Systems Development Method : The Method in Practice*. Addison-Wesley.
- Wood, J. and Silver, D. (1995). *Joint Application Development*. Wiley.