Human Aspects in Software Processes
A collection of former Pedagogical Patterns

Version 1.0, December 12, 2001

Markus Völter
MATHEMA AG
Syrlinstrasse 38
89073 Ulm, Germany
Phone: +49 / 731 / 140 128 – 23 555 93540
voelter@acm.org

ABSTRACT
A software project can only be successful if the developer team and its leaders form a working group. In small project teams, these "human issues" usually happen automatically if some preconditions are met. In larger projects however, these human aspects have to be explicitly facilitated.

This situation is similar to teaching a group of people. This can also only work if the human aspects are taken into accout.

This paper presents a set of pedagogical patterns rewritten in the context of project management to provide evidence for the assumption, that both domains share many common problems and solutions and that therefore the two disciplines should cooperate more closely.

1.1 Keywords
Project management, Processes, Patterns, Pedagogy

2 SOFTWARE PROCESSES AND PEOPLE
Running a software project is a difficult task. To simplify and structure this task, several processes and methodologies have been proposed. The V-Model, the Unified Process (and its commercial incarnation, RUP), eXtreme programming or Crystal Clear. All are about making sure that the project meets its goals and expectations, usually, finishing on time, within budget and meeting the customer's initial requirements, while still being fun for the developers.

The different processes and methodologies use different techniques to achieve these goals. Some use a relatively formal approach based on extensive specifications, analysis and documentation, others focus more on the produced code itself and employ a more incremental approach to software development. I don't want to delve too far into the details of the different processes, these things are well-known and need not be repeated.

In spite of all the differences among the processes, a project always has people as their most important resource. It is thus only natural to make the people an important part of the processes. This has been observed before, of course, for example in DeMarco's Peopleware [2] and by many successful project managers. However, it is not since long that this fact has been explicitly taken into account formally in software development processes. More traditional approaches consider the people as replacable resources that have a certain qualification and a task in the project.

The agile processes [1,4] that became popular relatively recently explicitly put the people in the center of all considerations. They realize that only with a well-working team of people (not resources!) can a project eventually succeed. The idea of focussing more on people than on formalism and paper currently spreads throughout the software process community and also "infects" the proponents of the more traditional approaches.

3 HUMAN ASPECTS IN LARGER PROJECTS
In small projects, it is relatively easy to make the members happy:

- you can make sure that the people on the team actually like each other, or at least find a way how they can constructively work with each other.
- every developer sees himself part of the team and strives to achieve the common goal

- ✍ they can easily talk to each other, exchanging their thoughts about the system and their opinions easily
- ✍ they usually have direct and regular contact with the leader(s) of the project (chief programmer, architect, project manager)
- ✍ and you can easily let these few people talk to the customers directly.

However, in larger projects, say, 30 to 150 developers, these human issues just don't happen:

- ✍ there will always be people on the project that somehow will not fit into the team. They might oppose to team work or are otherwise unable to cooperate with team mates. And you can't just get rid of these people!
- ✍ in larger teams, a specific person often only sees a part of the "whole" and thus loses sight of the common goal.
- ✍ the developers can't just talk to everybody else, it's just too many people. Well-working subgroups might form, but the interactions among these groups is limited. Communication about problem, features or other aspects of the system or the project are limited.
- ✍ the leader is usually one level removed. He cannot know all details, he is not available for everybody at every time; and some decisions might seem unclear to developers.
- ✍ The customer is usually far away from the developers. Communication usually happens very indirectly.

It does not really matter, which process you follow here, be it agile, formal, or no process at all: you have to do something to build the team and make sure the developers really see themselves as a team and work efficiently.

4 PEDAGOGY AND PEDAGOGICAL PATTERNS

When trying to train people, for example in software development, you have to make sure that they are in a situation that allows them to learn efficiently. Efficient Learning is only possible in a group of people that trust each other. There also needs to be a certain amount of trust between the students and the trainer/teacher, to make teaching or training a successful undertaking.

The pedagogical patterns project (see www.pedagogicalpatterns.org) collects successful teaching practices in pattern form. Several interesting patterns and pattern languages are available at the web site. Recently, there is a tendency in the project to focus on really general pedagogy, such as building a group or feedback.

5 AN EXPERIMENT – PEDAGOGICAL PATTERNS AND PROCESSES

We said that software development is primarily about people. So is training and teaching. One way to see seminar is a short project with a goal, a process and a leader[1] (the teacher). Thus, in this paper, I want to provide some evidence for the following assumption:

> *I assume that the same basic human issues apply in teaching and in running a project. In both cases, the relationship between a single person and a group, and between the people in that group, has to be managed.*

Why do I do this? I think once this assumption is accepted, the two domains can benefit from each other, and maybe they can create patterns or pattern languages together, focussing on human issues, group dynamics and people in general.

The way I want to provide this evidence is by rewriting some pedagogical patterns in the context of project management. The patterns I use for this purpose are taken from the SEMINARS pattern language, which I wrote together with Astrid Fricke in 2000 [7]. SEMINARS covers a typical seminar, from beginning to the end. The pattern language was very well received – which kind of proves that the patterns really apply in the teaching domain.

In the following sections, I will reformulate some of these patterns in the context of running a project. The selected patterns focus on proven methods to facilitate some of the communication/human issues in a project. For brevity, I have ommitted some of the examples provided in SEMINARS. Note that I do not claim that the following patterns are really patterns in the project management area – they are patterns in training and teaching and I think these *might also be successful techniques* for running projects successfully. Perhaps, they need to be adapted a bit, because in teaching ("very short projects") you have to do some things differently than in longer development projects.

SEMINARS is a pattern language and not just a collection of patterns. This means that it contains a lot of cross-references among the patterns, and the patterns should be read in a specific order. Because this paper only contains a subset of these patterns, some have "dangling references" to other patterns. I left the REFERENCES TO OTHER PATTERNS in the text to show how things go together, even if in some cases the referenced pattern is not included in this paper. Also, I tried to make as few changes to the patterns as possible to show that the *same patterns* might apply in both domains – I even didn't change things if I'd write them differently (or in a better English ✍) today

Note also that in addition to the rewritten patterns in this

[1] Thanks to Klaus Marquardt for this excellent sentence!

paper, there are also several pedagogical patterns that can be used directly, because the relationship among project lead (manager/architect/chief programmer) and the developers also involves some teaching: explaining requirements, explaining the architecture, et cetera. This is basically a kind of training where patterns like DIFFERENT APPROACHES, DIFFERENT WORK FORMS or CHANGE MEDIA apply neatly. These are not included here.

A last note: The described techniques are not new (they are pedagogical patterns, anyway) and many of the issues might have been discussed in the processes community (for example, see [6]). The point I want to make is to show the synergy among the domains.

6 THE PATTERNS

This section contains the rewritten patterns. They are grouped into five subsections. These are:

- Patterns about Leadership
- Environment Patterns
- Keeping the Project Focused
- Group-Building Patterns
- Feedback Patterns

6.1 Patterns about Leadership

In most projects, you have some kind of leader – be it the architect, the project manager, or just the most experienced programmer. Having a leader and a group of people led by the leader is always a potential source for problems. You should try to minimize these.

INVISIBLE LEAD

Usually, the lead is the central person of a project. He explains, decides, organizes. The developers are passive, listen, do not ask questions; often, they do not try to follow you in your thinking. Collaboration is most effective if the developers think and work on their own. Otherwise, the developers start to feel bored, ignored, unimportant and motivation is low.

Therefore, put the developers into the center of the project. Your job as a lead is to help the developers work effectively. The best leading method is not to lead at all, and help the developers work on the project on their own. You should offer the developers help whenever they have problems.

It is also important that you do not show off with your skill. This „show off" problem is especially critical with highly skilled leads, or gurus. A guru in a certain domain is not necessarily the best project lead, there is more to good project leadership than good domain knowledge..

LET THEM DECIDE

You are not completely sure about how to do certain things in the project regarding the contents, the schedule or the methodology. Sometimes it is impossible to make these kinds of decisions in advance because the exact requirements, skills or interests of the developers are not known.

Therefore, involve the developers in the planning of the project, or give them some alternatives at the beginning of the project they can choose from. The developers will feel better and the project will be more successful, because the developers were involved in decisions.

NOBODY IS PERFECT

At some time in the project, the developers will ask a question you cannot really answer. This might be unpleasant for you as the project lead. You might feel tempted to block such questions or give evasive answers. You can be sure that the developers will notice that!

Therefore, do not try to be perfect. In particular, if you cannot answer a question, admit it! This is even true if the question is directly about the project content you as the lead should know. But nobody is perfect. If you try to seem perfect, nobody will believe it. Tell the developers freely that you do not know the answer. Tell them that you'll look it up for them, or explicitly ask the fellow developers whether one of them can answer the question. You can also try to work out the solution together with the developers. Be sure not to create the impression that the question is overly difficult, and that everybody who tries to answer it cannot possibly know it because even you, the lead do not know it. Otherwise, nobody will answer such questions!

A lead who tries to be perfect is not very credible. Try to be honest and learn how to say „I don't know!" with grace.

6.2 Environment Patterns

To work efficiently, you don't just need nice colleagues and a well-functioning group. A pleasant, friendly and supportive work environment is also necessary. Here are some successful patterns.

COMFORTABLE ENVIRONMENT

The environment in which a project is run should support the creative process. Creativity and efficiency is only possible in an environment where a person feels good. In a situation or setting where a person feels stressed or otherwise uncomfortable, people cannot work effieciently.

Therefore, the project should be run in a comfortable environment that facilitates creativity. Do not try to create a living room atmosphere! The room should be large enough; it should be painted in friendly colors, it should be quiet and have large windows, et cetera. Of course, often it is not possible to modify the room to suit

your needs, but it may be possible to make it better at least.

Some ways to improve the atmosphere in a project room without renovating it could be the following: it is important that the room is tidied up, there should be only those things in the room that are relevant to the project. A wardrobe for the developers to put their jacket and bags is also a good idea. A very important aspect of the comfortable environment is TABLE ARRANGEMENT.

During the project, work results and other related information could be attached to the wall, thereby making the room a part of the project (DISPLAY RESULTS AND PROBLEMS). Cork strips can be attached to the walls of the project room to facilitate this.

It is also a good idea to add shelves for related books and material. Do not use cupboards to lock the books away. Make them available to the developers.

If you can select a room for use in the project, try to find a room with many and big windows, because daylight is better for work than artificial lighting. If you need lighting regularly, be sure not to use neon lamps because they create a very artificial kind of light. Use pleasant lighting. Consider placing some plants in the room. Do not make the room look like a hospital by using white walls, white tables, et cetera. Use wood or other friendly material, and add curtains to the windows. Also take care of the infrastructure. Make sure that there are enough power outlets and LAN plug sockets. A good projection surface for the overhead projector or beamer is also important, it should be large enough that the contents can be read from the back of the room.

Some of Christopher Alexander's patterns in [3] might also be worth a look, especially those from 159 onwards. They deal with the problem of making the inside of a building comfortable.

DISPLAY RESULTS AND PROBLEMS

The developers should be able to identify themselves with the project. This requires that everybody is aware of results, achieved goals, milestones and issues and problems. In addition to just REFERENCING THE PLAN (which states goals and objectives), everybody should see that the project produces usable and visible results (and thus reaches the goals and objectives).

Therefore, make results, reached goals, objectives as well as open points and problems visible and obvious to all developers. Use lists, posters, et cetera to do so. Every developer should be allowed to add open points or work products (in a structured way). Everybody can write down important topics and problems. As a consequence, the developers will be able to recognize what the group has achieved, and which points are still open. Questions do not have to be asked again and again, and the developers can see that the issues are not forgotten and will be adressed. This can also serve as a SUMMARY. Using this pattern can also help to build a COMFORTABLE ENVIRONMENT.

For example, put the problems and achieved results on a poster on the wall for everybody to see. A bulletin board can be used to collect questions and open points. If it is not possible to „decorate the room" this way, use sheets of paper that can be edited by everybody and keep them is a publicly accessible place. Copy these papers for everybody on a daily basis. Modern technologies like the WWW, news and email offer additional possibilities. For example you could use a project web page that everybody can edit over the web, using tools like WikiWiki.

6.3 Keeping the Project Focused

It's important to keep a project on track – after all, that's what project management primarily does. While each process has its own way of achieving that, here are some rather general guidelines.

PROJECT PLAN

Usually, you are very familiar with the goals and objectives of the project you run. You might not realize the problems the developers might have because you have dealt with the project objectives for a long time and know them by heart. Often, this results in an unstructured approach to the project which is hard for the developers to follow. You should not believe that explaining a lot compensates for an unstructured approach, because you might not distinguish between important and not so important topics. In addition, you will not want to say things like„Oh, I forgot to say when we were talking about XY....".

Therefore, create a project plan, or an agenda. This helps you to determine a general strategy for your project. The plan will keep you from drifting off into non-issues and unimportant details. Try to create plans on different abstraction levels, a high-level plan for the complete project and more detailed plans for each subtopic. The plans should highlight important goals, objecttives and techniques.

During the project, try to explicitly REFERENCE THE PLAN. If the atmosphere gets bad or if new features or requirements arise during the project, be flexible enough to LET THE PLAN GO.

A good plan is the result of experience. Be sure to adapt the plan from project to project, to incorporate new experience. During a project, continuously check whether the plan is still current and whether you are still on schedule. If not, the plan has to be adapted.

REFERENCE THE PLAN

Usually you have a PROJECT PLAN, but this plan is often not obvious to the developers. The developers ask themselves What does the project lead want to tell us? What is the goal of the project? How does this design/objective/topic relate to the overall goal?

If the developers do not know where you will lead them, they cannot easily follow you through the project. The project might seem unstructured although you have a PROJECT PLAN. Real motivation cannot build up.

Therefore, tell the developers about the PROJECT PLAN. Show it to them regularly. Briefly tell the developers where their current work is located in the overall context. Also reference the plan when you reach a goal or milestone. The developers will never be surpassed and they will be able to put the current topic into a larger context.

As always, there are different possibilities. You can show the PROJECT PLAN, and show where the developer's work is located. You can just tell what's comimg up next. Another alternative is to put the PROJECT PLAN on the wall and always have a pointer to the current location.

BUFFERS

During a project, unforeseen problems and issueswill arise. Developers might have important contributions, clients will come up with new requirements. You should be able to incorporate these additional issues (at least to some extent) without ruining your PROJECT PLAN. You should not ask the developers to „add an extra hour/day" to include the additional issues or requirements.

Therefore, when creating the PROJECT PLAN, keep these unforeseen issues and requirements in mind by including time buffers. Because unforeseen issues will come up in any project, it is safe to incorporate these time buffers from the beginning. If not, you can finish early! However, be sure not to drift too far from your PROJECT PLAN. If a requirement is completely out of the scope of the project, don't include it.

6.4 Group-Building Patterns

The aspect of keeping the group's integrity was the starting point for writing this paper. People can only work well if they feel good in their social environment. Problems among the people on a team are one of the most typical reasons for failing projects.

INTRODUCTION SESSION

To make PERSONAL COMMUNICATION possible during the remainder of the project, the developers need to learn something about each other and the project lead at the beginning. Usually, the developers are a little bit shy and you, the project lead, should start this process. In addition, you might want to learn something about the developers, to ADAPT TO THE PARTICIPANTS BACKGROUND or to LET THEM DECIDE.

Therefore, take the time at the beginning of the project to let everybody introduce him-/herself to the others. The developers should be given a chance to state their expectations towards the project and tell the others about their professional background, their company, et cetera. This session should be held in an informal context, which can be achieved by using a suitable TABLE ARRANGEMENT. It is also possible to use GAMES at the beginning of such a session. To make a start, you should begin the session by introducing yourself. Be sure to introduce yourself, not the project.

There are different ways how this introduction session can be held. The most common form is that everybody introduces himself to the others, including his name, employer, his field of activity, his role in the project, et cetera. It is a good idea to let the developers decide what they want to include in their introduction. An alternative form is to let one person interview another person and introduce this other person to the group. The introduction session usually ends with everybody attaching a NAMEPLATE to himself.

PERSONAL COMMUNICATION

In addition to creating a physically COMFORTABLE ENVIRONMENT, it is also important that people feel well from a social point of view. They need to talk to each other informally. This is especially true if they work in the same part of the project and come from different companies, because they will want to talk shop with each other. If you do not give them the time explicitly, they will do it during the project work. It is also very important that there are times where the distinction between you (the project lead) and the developers becomes less important. Informal talking is necessary to build up a harmonious group.

Therefore, make sure that the developers have time to talk! Create spare times where all developers are together in an informal setting. These spare times should not be filled with project issues. It is important that you are also present during these spare times, but not playing the project lead role, you should be „on the same level" as the developers. Try to avoid talking about the project.

Personal communication can help to solve problems among the developers or between you and developers. To facilitate personal communication be sure to use NAMEPLATES. Playing GAMES can provide a good environment for talking.

The group could go to lunch together or visit a pub in the evening. The COMFORTABLE ENVIRONMENT can offer small tables, where people can stand, drink coffee and talk to each other.

ANONYMOUS MAILBOX

There are people who are not able to solve problems by PERSONAL COMMUNICATION with other developers or you, the project lead. You should still give these people a chance to communicate their concerns. And sometimes, real problems have risen in the group that cannot be solved just by talking while having a cup of coffee.

Therefore, create an official way for developers to communicate to you anonymously. Important information might be communicated to you via this path that would otherwise not reach you at all. If the group works well, PERSONAL COMMUNICATION will suffice to solve problems and the mailbox will stay empty. If this pattern gets used heavily, this indicates that there are serious problems within the group or in the relationship developers/project lead.

An example for this pattern might be a real mailbox somewhere in the class room where the developers can put notes. Email and web technology offers other possibilities, of course. At the end of the project or an iteration, offer feedback forms that can be filled by the developers anonymously.

NAMEPLATE

If you want to create a good atmosphere and facilitate PERSONAL COMMUNICATION, you should give the developers and you a way to call everybody else by name.

Therefore, be sure that everybody wears a nameplate on his shirt, so that everybody's name is present all the time. Nameplates on the work place are not suitable, because during breaks and other informal times when the names are most important, they are not available. Be sure to find a consensus whether to use first or last names. After people get to know each other, the nameplates can be removed, of course. Using the pattern is especially important in the beginning of a project.

An ideal and simple way to create nameplates is to use Tesa Krepp (a German brand of adhesive tape usually used for nameplates) and write the name on it with a thick pen.

GAMES

You realize that it is important to have PERSONAL COMMUNICATION. To make this possible, a harmonious group, a „team", must be built up, especially if the group works together for a longer period of time and they did not know each other before the project. Such group-building processes should be explicitly assisted at the beginning of a project. An atmosphere of trust must be created among the developers to allow effective work. This is part of a COMFORTABLE ENVIRONMENT.

Therefore, consider playing games in the group. The games should emphasize achieving things together. Games that aim at distinguishing winners and losers are not suitable. The games should leave enough room for PERSONAL COMMUNICATION. Games improve group quality, and they are usually not very demanding, from an intellectual point of view, serving as a good way to relax. However, games can be dangerous if the developers do not like them or consider them childish or otherwise useless. Never use games if the developers do not want it.

The positive effects of playing games in a group can be observed at the EuroPLOP conferences. A special person (George) is there to organize and coordinate games. Real, mature people play them, and this is one of the reasons why the EuroPLOP is so special.

6.5 Feedback Patterns

Everybody wants to know whether his work is good or what could be impoved. You must make sure to tell your developers what they do well, and, more importantly, where they have to improve.

FEEDBACK

The developers want to work as effectively as possible in the project. But often, they are not sure whether there work is good enough and appreciated. Such a feeling of uncertainty can reduce motivation significantly.

Therefore, give the developers feedback about their work. The feedback should be differentiated and objective. Always begin with the positive feedback. Criticism should always help the developers to improve the criticized aspect. Be sure to give the feedback in time, later feedback is not effective. Positive feedback can significantly increase motivation.

HONOR QUESTIONS

Often, developers do not ask questions because they fear that the question is „silly" in the eyes of the fellow developers or the project lead. They fear to show weakness. This is unfortunate because questions show that the developers think about a topic. A question can also give a good hint on a new problem that has not yet been taken care of.

Therefore, motivate the developers to ask questions, also if they seem silly or if they show that a developer does not fully understand the topic. If NOBODY IS PERFECT, the developers need not be perfect, either. Always honor

questions, not bright answers. If you have a working group of developers and if they accept you as a partner, the courage to ask „risky“ questions raises.

This pattern was inspired by Prof. Joseph Bergin at EuroPLOP `99. He gives „tokens of approval“ to „participants who reveal that they are struggling with new concepts“ .

PROJECT DEBRIEF

When a project is over, you will have made some good and bad experiences, you will have discovered problems and mistakes on your side and on the developer's side. Probably you will have made some positive experiences, too. You do not want to make the same mistakes again, and you will want to repeat the good parts in the upcoming projects.

Therefore, be sure to do some kind of debrief after a project. All positive experiences and the obvious mistakes should be captured for reuse or avoidance in future projects. This debrief should be valuable for the lead and for the developers.

You could either just think about the project and capture the content on a piece of paper. It is also a good idea to have some kind of predefined checklist that asks certain key questions.

SUMMARY

The developers should leave after a day (or after an iteration) with a feeling of accomplishment. They should know exactly what they have accomplished what problems might have risen. Sometimes, the important parts get lost in the details that have been discussed during the day or iteration. Always make sure that the developers can see the big picture.

Therefore, at the end of each day or iteration, provide a summary that restates the achieved goals or encountered issues. Relate these topics to things accomplished in previous days or iterations and illustrate how the rest of the project will build on these achievements. Be sure to emphasize essential topics and show big picture in the PROJECT PLAN. If your project is broken up into DIGESTIBLE PACKETS, summaries are simplified.

7 CONCLUSION

In larger projects, some of the human aspects just don't come automatically and they have to be factilitated explicitly. Based on the hypothesis that efficient teaching and efficiently running a project is both an undertaking that has to emphasize the "human aspects", and based on the observation that in both contexts the good things don't come automatically, I have presented a set of pedagogical patterns in the context of projects management.

The goal is to show that we should actually focus on patterns about how to keep a group of people work together efficiently – no matter in what setting.

I'd like to hear your opinions on the patterns, and on the idea of reusing pedagogical ideas from teaching in projects. You might want to have a look at the original SEMINARS [7] paper to compare the patterns to their original counterparts.

8 ACKNOWLEDGEMENTS

I'd like to thank several people for their reviews and comments on this paper, especially Klaus Marquardt, who has provided me with comments on several iterations of the paper. The others are, in alphabetical order, Jutta Eckstein, Malte Finsterwalder, Manuela Nagel and Alexander Schmid. The papers has been revised significantly based on their comments.

REFERENCES

1. Alistair Cockburn; *Agile Software Development*, Addison-Wesley, 2001
2. DeMarco, Lister; *Peopleware – Productive Projects and Teams*; 2nd Edition, Dorset House, 1999
3. Alexander, et al, *A Pattern Language, Towns, Buildings, Construction*, Oxford University Press, 1977
4. Kent Beck*; Extreme Programming Explained: Embrace Change*; Addison-Wesley 1999
5. Matt Stephens; *The Case Against Extreme Programming*; see http://www.bad-managers.com/Features/soapbox/xp.shtml
6. Don Sherwood Olson, Carol L. Stimmel; *The Manager Pool: Patterns for Radical Leadership*; Addison-Wesley 2001
7. Astrid Fricke, Markus Voelter; *SEMINARS - a Pedagogical Pattern Language about teaching seminars effectively*; Proceedings of EuroPLoP 2000 or at www.voelter.de/seminars