# Lead Roles In A Software Project

Andreas Fießer
patterns@fiesser.de
January 2010

This paper discusses patterns about the lead roles of a software development team: The Product Manager who is the interface to the customer, the Project Manager who allocates the resources and the Architect who designs the software and leads the developers.
A fourth pattern discusses how these roles act as hubs and how they relate to each other. The view is based on the project management triangle.

## 1 Introduction

Projects are teamwork. When teams reach a certain size they require organizational structure to be effective. Team members need to set different emphases in their work to make sure the project goals are fulfilled.
This paper illustrates how different roles can be used to organize a software development team. When looking at the flow of communication, it becomes apparent that distinct roles communicate differently.
As Coplien and Harrison describe in their pattern HUB, SPOKE, AND RIM some roles act as hubs [Coplien Harrison 2005]. Typical examples of such hubs are the PRODUCT MANAGER, the PROJECT MANAGER and the ARCHITECT. Depending on the situation roles like developers or testers might also act as hubs.

## 2 Audience

The patterns presented here try to structure communication and responsibilities in a project team. The focus of this paper is planning project teams, not the detailed actions of the respective team members. The patterns might also work in other domains than software development.
The author's experience stems largely from web development in small teams of less than a dozen people. The products created are individually constructed and the development process takes several months. This setup is also the focus of this paper. Nevertheless the patterns also apply to other scenarios. The author welcomes any kind of feedback about the relevance of these patterns in other setups.
For the scope of this paper, creating a mass market product can be treated in the same way as a tailor-made product ordered by a customer. In both cases there are certain needs that are to be met and thus features that have to match the needs.

Copyright retained by author. Permission granted to Hillside Europe for inclusion in the CEUR archive of conference proceedings and for Hillside Europe website.

# 3 Pattern Map

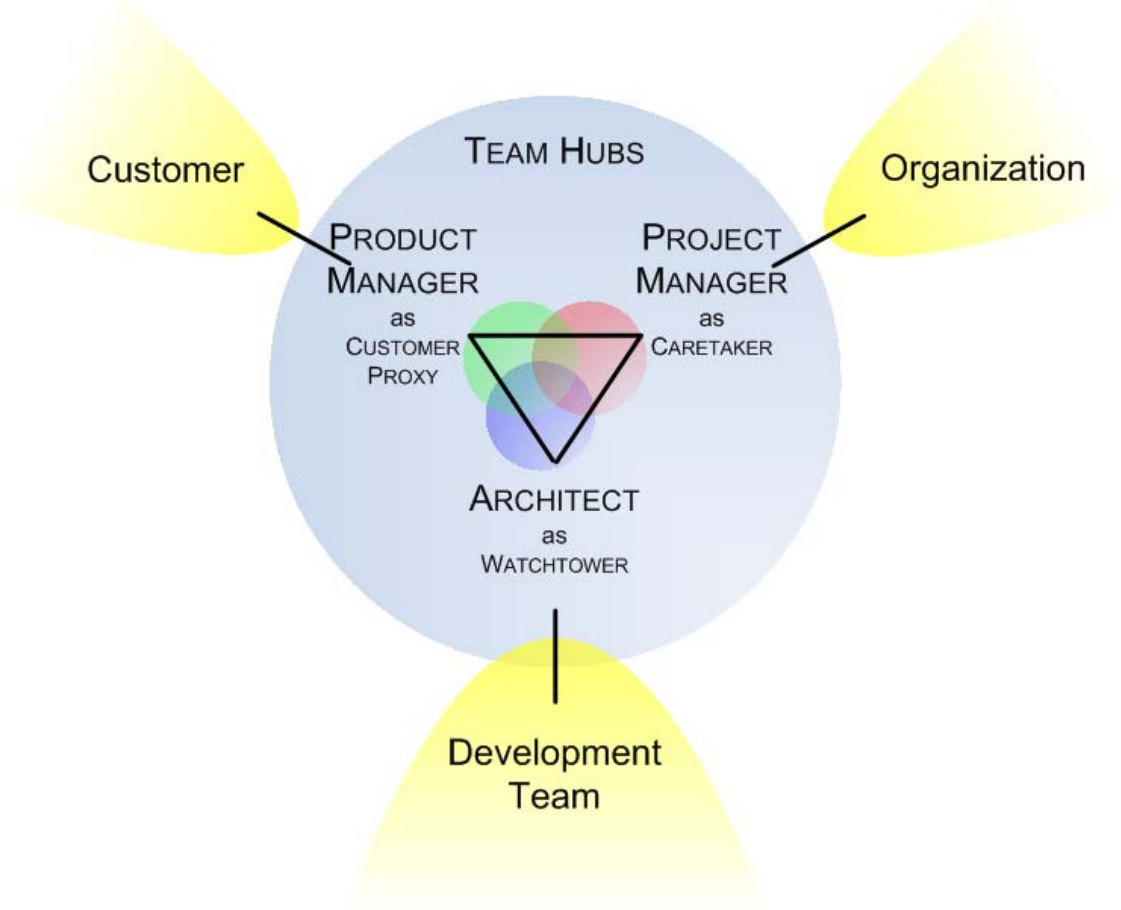


**Fig. 1: Hubs and Stakeholders**

For a software project stakeholders come together from different directions. Their roads cross inside the project team. The goal of the project has to be to avoid conflicts and turn the different views into an advantage.
After taking a closer look at the use of TEAM HUBS in this context, the three roles of the PRODUCT MANAGER, the PROJECT MANAGER and the ARCHITECT are presented in patterns.
The patterns in this paper focus on the hub aspects of the mentioned roles. Although this is only part of the roles, the patterns are named after the corresponding roles coined by literature and practice for readability. Their hub function is expressed by their alias names: CUSTOMER PROXY, CARETAKER, WATCHTOWER.

# 4 The Patterns

## 4.1 TEAM HUBS

### Context

A team of several people is working on a software project that has a limited budget and a deadline. The customer expects the finished product to meet his expectations.
Some tasks in a project are not closely related to coding.

### Problem

**How to organize the flow of information in a project?**

### Forces

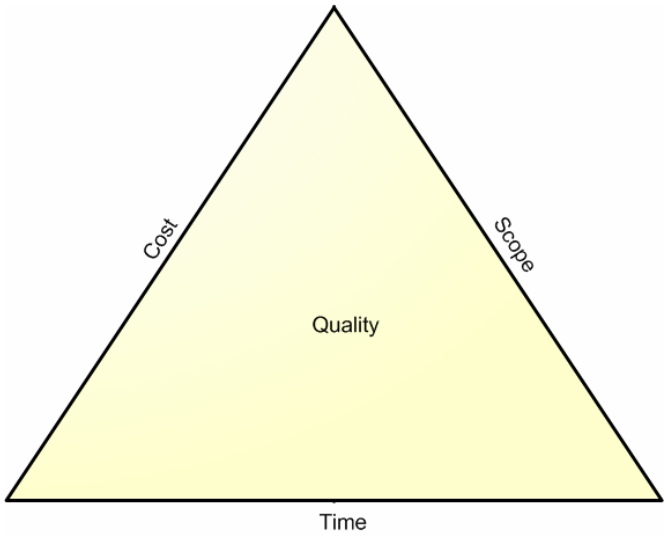


**Fig. 2: The Project Management Triangle**

- As illustrated by the Project Management Triangle [Schwanninger Kircher 2009, Kelly 2008, Wikipedia 2009a] projects happen within certain constraints. You cannot change one aspect without touching another. Focussing on all four conflicting aspects at the same time is very difficult for a single person. Letting every team member acquire all skills necessary to carry out a project is not effective either. Management tasks and coding require completely different mindsets [Kelly 2009a].
- As the developer is the one who actually creates the product, he should play a central role in the process. Nevertheless he needs support by others.
- Communication becomes more complex with an increasing number of participants.

## Solution

**Classify the aspects of a project and channel the communication and tasks they implicate over dedicated hubs.**

These hubs perform all the management activities that are not directly connected to coding. They actively look for tasks that fall to their sphere and support the team with their skills.

To formalize the hubs create several roles, each only focussing on a subset of the four aspects. There are many roles in a software project. Typical examples of hub roles include the PRODUCT MANAGER, the PROJECT MANAGER and the ARCHITECT. These roles bundle the interaction between their respective group and the outside. This does of course include other hubs.

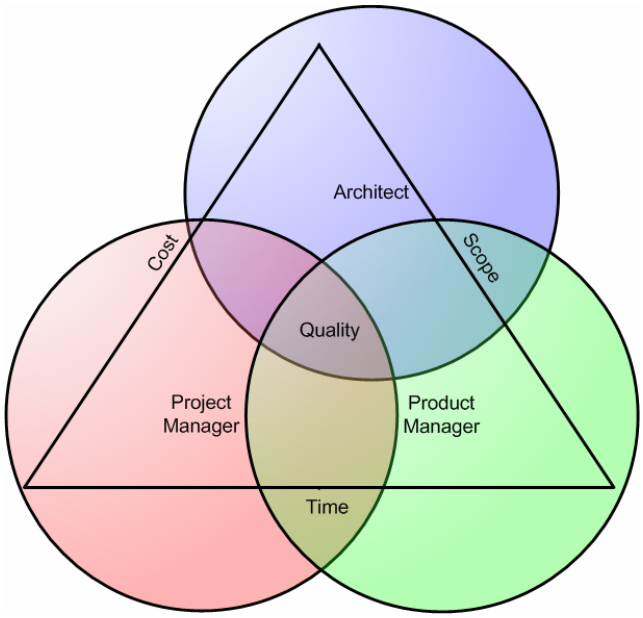


**Fig. 3: Aspects covered by roles**

Sometimes hubs emerge naturally from the characters of the team members. A beneficial formal organization structure fosters these momenta instead of blocking them.

The most obvious way to separate the tasks is to assign the roles to different individuals. If there is nobody available full time, make sure that somebody dedicates at least a share of his time to switch views. Although it reduces the necessary communication efforts this option can only be a workaround as it requires considerable discipline from that person to stay objective in both roles [Kelly 2009a, Kelly 2009c].

## Rationale

Assigning an aspect to a role makes sure that every constraint is sufficiently taken into consideration when making decisions. Overlapping the aspects makes it easier to find a compromise of two positions.

In software development the types of quality that the roles care about are different in focus. As Schwanninger and Kircher explain, high quality for the ARCHITECT means developmental quality, i.e. maintainability of the code. For the PRODUCT MANAGER it is the user perceived quality, e.g. usability and performance. The PROJECT MANAGER focuses mainly on regulatory quality. This means he has an eye on the compliance with standards the company or the project have committed to [Schwanninger Kircher 2009].

## Consequences

### Benefits

- Representing a specific role makes it easy for a team member to have a specific point of view and argue for it.
- It also allows for the roles to be filled not by generalists but by specialists. So a wider range of skills and experience can be considered when making a decision.
- Overlapping spheres might seem like ineffective redundancy at first. But they guarantee that the roles share common interfaces they can use for negotiation.
- Since requirements, planning and implementation are done by different people each of them is more objective as they are less biased by the other side.
- The work of the developers is coordinated and prepared by lead roles with a different focus. This relieves the developers from organizational details and other non-technical issues. They can focus on their strength: Coding. The developers as well as the stakeholders of the project know whom to address with an issue.
- Introducing more than one leading role leads to a division of powers that prevents arbitrariness.
- A team member fills a role with his skillset and experience. If the person leaves the team, it is clear which requirements the replacement person needs to fulfil to be able to take over the role.

### Liabilities

- As a person is asked to fill a predefined role, the team member might feel pressed into a role that does not fit well. It is important to allow the freedom that the role adjusts to the person as long as this does not touch other goals.
- When splitting up into different roles, team members have different responsibilities. This tends to create a hierarchy between managers and producers. As the roles described here have no disciplinary rights, this requires more networking and a more sophisticated style of communication.
- The PROJECT MANAGER and the PRODUCT MANAGER are detached from the code. So it is necessary to abstract communication when talking about technical issues.
- Explicitly separating the aspects of the project creates the opportunity to delegate responsibilities for a problem. If the one the problem was delegated to does not take care of the issue properly, there is a risk that nobody feels responsible any more and the problem is neglected.
- Separating the roles creates a management layer. This requires extra manpower and creates extra cost.

### Example

A caravan manufacturer contacts a web agency with the blurry vision to add a section to his website that can exclusively accessed by his retailers.
Before the web agency's developers start to work, the PRODUCT MANAGER advises the customer on the set of useful features to support his intentions.

After the PRODUCT MANAGER and the ARCHITECT have jointly estimated the required budget, the customer picks the features he wants to get implemented.
The PROJECT MANAGER now allocates the available resources.
The developers start to work within the given structure of requirements and deadlines.

## Related Patterns

The PROJECT MANAGER, the PRODUCT MANAGER, and the ARCHITECT act as TEAM HUBS. The ARCHITECT contributes technical overview. The PRODUCT MANAGER shares his knowledge of the target market and the PROJECT MANAGER contributes monitoring and organization.
Coplien and Harrison have created a number of patterns that give some details on how the TEAM HUBS should work. A productive team is a COMMUNITY OF TRUST in which WORK FLOWS INWARD to the developers so that the DEVELOPER CONTROLS PROCESS (in the team) and the other roles are supporters. Depending on the context this pattern is complemented by HUB, SPOKE, AND RIM. The approach that developers ENGAGE CUSTOMERS finds its limits in a FIREWALL that shields the developing staff from external interruptions and noise and a GATEKEEPER that facilitates the flow of useful information between the team and the outside.
PRODUCT MANAGER, PROJECT MANAGER and ARCHITECT are implementations of these patterns for their domain [Coplien Harrison 2005].
See BALANCE CONSTRAINTS [Schwanninger Kircher 2009] about how to cope with the aspects of the project management triangle.

## 4.2 PRODUCT MANAGER

### Also known as

CUSTOMER PROXY

### Context

The customer has certain needs and a rough idea about how the software should meet them. The customer has limited resources to supervise the project. The ARCHITECT and the development team are focussed on solving technical problems and creating features.

### Problem

**How can you ensure that the product meets the customer's expectations?**

### Forces

- The customer wants the product to meet his expectations but he might not know how to communicate them. The customer might not have a consistent view of his needs and has no overview over the technical possibilities.
- The customer is not interested in the concrete technical implementation. He wants his requirements to be fulfilled.
- The customer is required for feedback but he might not be available when needed or willing to get involved in the development process [Völter Kircher 2008a].
- Developers are not experts in the customer's domain. Because they should be able to relate to the intention of the requirements, they need some understanding of that domain without investing too much time to gather the required knowledge.
- Features that look easy to the customer might turn out to be complicated to implement. This might prolong the development process.
- Taking care of the customer can be a time consuming process [Kelly 2009b].

### Solution

**Bundle communication between the team and the outside.**
Gather market knowledge in one point to create a perspective free from the details of technical implementation inside the development team. This creates the role of a PRODUCT MANAGER who is able to support the customer in specifying his requirements and explain these requirements to the developers. He helps to clean up inconsistencies and point out solutions the customer might not have thought of.

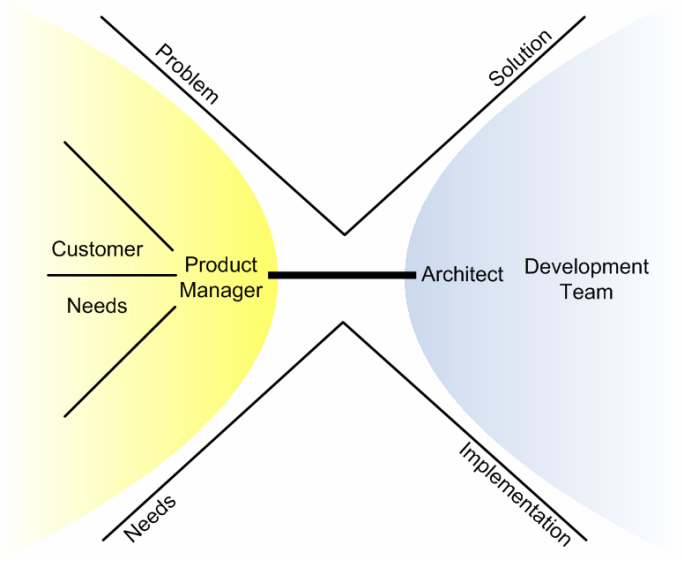


**Fig. 4: The Product Manager at the Focal Point**

The less the actual customer is available, the more the requirements become a base for the PRODUCT MANAGER to act as a SURROGATE CUSTOMER for the team [Coplien Harrison 2005].
In the outward direction the PRODUCT MANAGER lets the customer know when a certain request involves extraordinary technological challenges. This allows them to see why a certain feature creates so much effort or even trade it in for some less complicated features that achieve a similar effect.
Starting from that point and keeping the customer in mind, the PRODUCT MANAGER can turn to the inside of the team and start negotiating with the development team lead by the ARCHITECT about the actual way to implement a feature. In this process the decisions are made, which technology will have to be implemented. The choice might be made in spite of the challenges it brings. Technologies that seem fancy at first may be cancelled because they do not add to the value of the product [Kelly 2009b].
When discussing within the team it is important to make sure that the features planned and their priorities match with those of the customer. Consequently, substantial technical constraints in implementation need to be carried on to the customer to find an efficient solution. It is important to use this option with caution. It is not the sole purpose of the PRODUCT MANAGER to make life easy for the developers. The ARCHITECT must be challenged to look for a solution outside of his comfort zone to prevent bothering the customer unnecessarily.

## Rationale

Now all market knowledge comes together in a single point. At the same time the insight into the product's features cumulates in this point. The focal point in which both sides converge is the place where the overview is optimal.
This position is ideal to mediate between the two viewpoints. In the end the goal of all actions is to create a product that is attractive to the customer. The customer does not care about technical details.

## Consequences

### Benefits

- Customer wish lists and techie talk become structured information in the language of the target before reaching the other party.
- By having an outside view, i.e. outside of the code, the PRODUCT MANAGER is not biased by the effort or the complexity of the technology that is necessary to create the desired features. He can come up with ideas inspired by nothing but the customer needs.
- The customer and the development team now have a clear contact that can explain the other side in their language and forwards their issues to the other viewpoint.
- Somebody watches that the intention of the customer is followed inside the executing team without the customer disturbing. Optimally, the customer can limit himself to reacting to the progress reports and does not even have to take any proactive action.
- The PRODUCT MANAGER has a clear vision of the product but at the same time is part of the team. This allows immediate decisions that are nevertheless robust and objective.
- Outlining the customer needs for the team members motivates them because they know the reason behind a request. The given information also serves as a background for the team and by that improves the quality of the feedback.
- Especially in busy times it becomes a benefit that the resources required for customer care are not drawn from development capacity. Taking your time for the customer does therefore not slow down the coding.

### Liabilities

- As the contact between the customer and the development is limited it is a challenge to assure that information reaches the other end without shifting the message.
- Laying the feature strategy in the hands of one person evokes the risk of subjective influence.

### Example

The caravan manufacturer wants his retailers to reduce support requests, order spare parts and stay in close contact with him. The company is an expert in building vehicles. Even in their marketing department there is no expert on online communication. So they hire a web agency to set up an ecommerce system and a blog. The agency's PRODUCT MANAGER discusses the customer's ideas, presents more options and proposes possible solutions. While discussing the details of the required ecommerce system to order spare parts it turns out that the caravan manufacturer rather wants his retailers to communicate with each other to exchange experience than to bother him. Nevertheless he wants to keep an eye on the issues they are concerned with. So the idea of a blog is discarded and replaced by a bulletin board. To detail the solution the PRODUCT MANAGER consults the ARCHITECT that was assigned to the project.
As the field of the wanted features narrows, the PRODUCT MANAGER starts to write down specifications to prepare the quote. He consults the ARCHITECT on open issues to get a more concrete picture of the actions and effort required.

After a set of features has been authorized, the PRODUCT MANAGER gives feedback to the developers about the features they develop and requests that the board allows to attach files to postings. He is told that this requires considerable coding effort because the software package used as a platform does not provide this feature. From discussions he had with the marketing director the PRODUCT MANAGER knows that the retailers need to exchange manuals, sketches and other documents. So in spite of the effort it causes, the PRODUCT MANAGER insists on the feature. He asks the PROJECT MANAGER to block the developer in charge of the board for some more time.

## Related Patterns

The PRODUCT MANAGER passes on all necessary information the PROJECT MANAGER needs to set the stage for the project. In turn he receives an organizational framework for the project.
The PRODUCT MANAGER strives for maximum customer value. By that he is a counterpart for the ARCHITECT who seeks technical quality.
Depending on project size the role of the PRODUCT MANAGER might not be executed by a single person but a team of requirements engineers.
In Scrum the PRODUCT MANAGER is replaced by the Product Owner [Kelly 2009b].

This pattern is an implementation of Coplien and Harrison's SURROGATE CUSTOMER [Coplien Harrison 2005]
Unlike a PRODUCT MANAGER, a business analyst does not analyze the needs of various customers but of the single company that employs him. It is a double inward view since this is the same company as the one that conducts the software project to satisfy the needs. [Kelly 2009c]

## 4.3 PROJECT MANAGER

### Also known as

CARETAKER

### Context

A team of several people is to work on a project. There are requirements and limited resources of time, staff and budget to implement them.

### Problem

**How can you ensure that a project reaches its goal and stays within its given constraints?**

### Forces

- A feature might turn out to be much more complicated to implement than anticipated. So it will exceed the resources allocated to it. As the main costs in software projects are personnel costs, an increase of project duration commonly means a rise of costs.
- The needs of the customer might change due to changes in his market.
- As time goes by the vision of the product becomes clearer. The customer comes up with ever new ideas how to improve the features. Explicit new features would cause a request for extra budget, so the new feature requests are camouflaged as bugs or minor extensions of existing features. Nevertheless this scope creep requires time and effort to be implemented.
- The customer might be willing to cut features to meet the deadline. But depending on his situation he might instead prefer to either postpone the deadline or even increase the budget to keep scope and quality.
- During the course of the project, unforeseen events and trends might occur.
- There might be a finished third party solution available. Using it instead of developing the module yourself might save time and reduce risk but it might not fulfil the same quality standards as the main project. Licence costs might occur.
- Striving for the perfect product promotes quality but – as the Pareto Principle predicts – the higher the quality requirements are, the less efficient the development process becomes [Wikipedia 2009b].
- Motivated staff is more productive.
- Monitoring and controlling are not contributing directly to the product itself. So the capacities they consume increase the overall costs of a project without having a tangible result.

### Solution

**To ensure project success, have a dedicated person to set the stage within the constraints of the project. While monitoring the project, it is this person's job to create a good working atmosphere to motivate the team.**

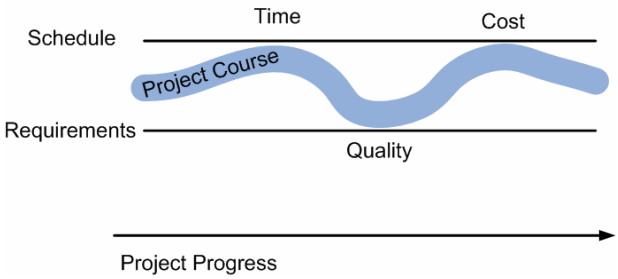


**Fig. 5: Steering the project within the project constraints**

The PROJECT MANAGER creates a plan considering the given resources and defends it. He assigns tasks to the team members and keeps an eye on the project environment. He steers the resources actively, takes decisions when necessary and advocates trade-offs.
The PROJECT MANAGER tracks the progress of the project. Whenever he discovers a weak spot, he takes actions to remove it.
Instruments of the PROJECT MANAGER include cutting features, postponing deadlines, extending funds, replacing tailor-made features by third-party-modules and, with caution, adding manpower [Kelly 2008].
When it comes to trade-offs between the project constraints it is important to know the customer's priorities. To further increase the options there should be some buffer in every plan and actions should be taken as early as possible.
To keep the schedule and reach an optimum in quality, the focus needs to shift during the course of the project. First it is implementing new features to perfection. Later in the project the PROJECT MANAGER needs to make sure that the developers concentration shifts towards debugging of the existing features in order to finalize the product.

## Rationale

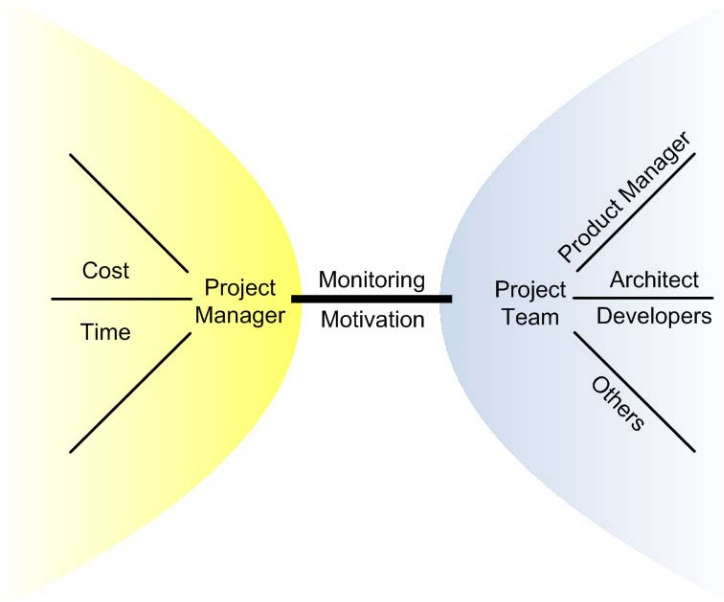


**Fig. 6: The Project Manager shielding the constraints**

Developers want to concentrate on their work. Their focus is on the details of the code. The same is true for the PRODUCT MANAGER who focuses on customer value. So having a separate role that is not dazzled by such aspects makes sure that the other project constraints are not disregarded. Being outside of the creational process, the PROJECT MANAGER can also take care of social aspects.

## Consequences

### Benefits

- The stage is set. The developers can concentrate on coding in a productive atmosphere and the PRODUCT MANAGER has room to focus on requirements.
- The resources of a project get enough attention to be managed with overview and foresight.
- A neutral role can moderate between conflicting interests and facilitates trade-offs.
- High motivation and thus productivity in the team takes tension from the project as it saves time and money.
- Monitoring creates transparency. The positive effect active management can have on keeping other actions within the plan makes it a very profitable activity.

### Liabilities

- The PROJECT MANAGER is far away from the actual development. Functional quality is not his main concern. He sees features as milestones and judges them by their status.
- It is important that the PROJECT MANAGER as the ever monitoring authority is perceived as a supporter not a threat.

## Example

The PRODUCT MANAGER requests more resources for the bulletin board to get the picture attachment feature implemented. The task that would need to be postponed for that is the multi-language support of the parts order tool. The PROJECT MANAGER asks the PRODUCT MANAGER to decide which feature has the higher priority. As not all retailers will use the board but all of them shall use the order tool, and the multi-language support is crucial to the latter, it is prioritized. After evaluating all other options, the file upload feature is scheduled for the buffer time at the end of the development timeline. In parallel the customer is asked whether it would be ok to postpone the feature after the launch if necessary.

## Related Patterns

The PROJECT MANAGER stays in close contact with the ARCHITECT to learn about the effort a feature causes, to update project status and discover problems early. He also interacts with the PRODUCT MANAGER. He knows when requirements change and can be asked to cause it actively.
Coplien and Harrison have created a number of patterns on project management.
SIZE THE SCHEDULE explains how a schedule should be dimensioned.

When the COMPLETION HEADROOM gets out of control, there are means for crisis management: SOMEONE ALWAYS MAKES PROGRESS, TEAM PER TASK or SACRIFICE ONE PERSON. If they are not sufficient, the PROJECT MANAGER should better TAKE NO SMALL SLIPS to keep up motivation of customer and team.
The limit of detail to which the PROJECT MANAGER is concerned is set by INFORMAL LABOR PLAN.
To take care of the social aspects in a team, the PROJECT MANAGER should incorporate the MATRON ROLE and foster DEVELOPMENT EPISODES. [Coplien Harrison 2005]

## 4.4 ARCHITECT

### Also known as

WATCHTOWER

### Context

There is a variety of technical options to implement the requirements and a team of developers to do so.

### Problem

**How should the requirements be transformed into software?**

### Forces

- It is tempting to include as many features as possible in the product. This strains cost and time and it does not even necessarily improve quality.
- The development team concentrates on solving technical problems and creating features.
- A single person can only handle a limited workload. But the more hands are working on a product, the faster it becomes inconsistent, redundant, inflexible and error-prone.
- Established technologies are easy to handle, quick to implement and less buggy. New technologies might offer options that have not been available before. But using new technologies implies risks: Uncertainty, compatibility, security, introduction effort.
- Using it third party features might save but it comes with the risk to lose quality because it is not tailor-made any more.
- Using complex frameworks or third party modules creates a lot of overhead but it may save time.
- Marketing people often do not understand technical details and language.

### Solution

**Have somebody design the big picture. He is the one to check changes in the architecture for consistency with the existing system. He has to keep track of technological innovations and monitor the quality of the code.**

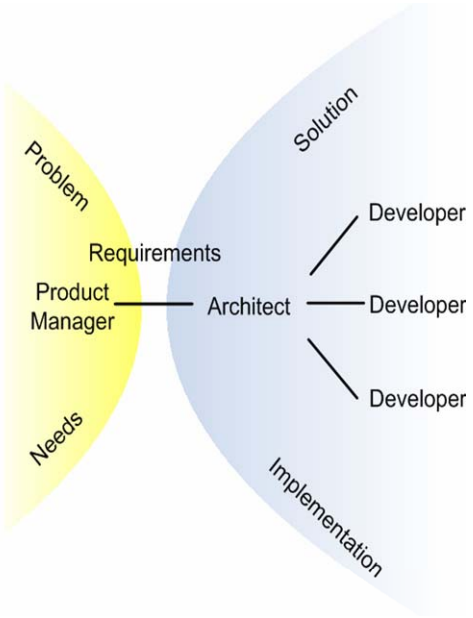


**Fig. 7: The Architect as interface to the outside world**

The ARCHITECT brings all his knowledge into the design. To increase the pool of experience, he orchestrates the ideas of the other developers. The ARCHITECT is to encourage the team to look for solutions off the beaten track as well.
The team has to agree on the technical platforms to use before starting to code. Also to the inside the ARCHITECT makes sure that the features implemented correspond to the requirements and are not just nice to have. He creates coding guidelines to support consistency.
The ARCHITECT's job is to break down complexity and structure issues to make the design decisions accessible for non-technical contacts. He can filter outside requests before they reach the development team.
During the project the ARCHITECT has to abstract on the development to discover when the code drifts away from the goals set and make sure it is corrected. Discussing design decisions before and during development makes sure they are incorporated by the team and not perceived as commands that just need to be obeyed to.
To keep in touch with the code, the ARCHITECT should be a lead programmer or at least at times be involved in coding.

## Rationale

The ARCHITECT is a watchtower in two directions. To the outside of his realm his function is to be the first to discover a change and judge whether it might be a chance or a threat. He then reacts accordingly. Only if action is required he informs the rest of the people.
Guidelines provide orientation. He also watches the inside to make sure that everything follows the rules that were set. Knowing that these tasks are done by somebody, the rest of the team is able to concentrate on their main business without fear.
Proper delegation and division of work allow growth without overloading the single person and still create a coherent product.

### Consequences

#### Benefits

- Requirements and coding guidelines channel the developers ideas and also give them a basis to reflect on their work.
- New technological options may be a leap in product and development quality. The risks they might bring have been anticipated and are mitigated.
- Keeping the overview for them and providing the developers orientation inside the project gives them peace of mind for development. At the same time it does not confine the developers. Whenever they want to get an overview, they know whom to ask.
- The central hub brings clear communication channels that can be used. This makes it easier to expand the team because adding a new member does not necessarily require a new connection to every single team member.
- Shielding outside communication and introducing a hub for outside communication that understands their language frees the developers from the need to translate their detailed view into a non-technical one themselves.

#### Liabilities

- A predefined environment narrows creativity for the single team member. It requires discipline.
- Having a head of development that does the communication with outside requires an extra step in communication.

### Example

Before giving a cost estimate for the parts order tool the ARCHITECT starts an investigation to compare possible third party platforms. He gathers the available experience from the developers and lets them evaluate unknown packages. On this basis the ARCHITECT and his team settle for a solution. The ARCHITECT presents it to the PRODUCT MANAGER and gives a cost estimate for its customization to the PROJECT MANAGER.

### Related Patterns

The ARCHITECT discusses technical feasibility with the PRODUCT MANAGER.
The PROJECT MANAGER requires a plausibility check for his schedules and feedback about progress during development.
The ARCHITECT as a watchtower is an implementation of Coplien and Harrison's ARCHITECT CONTROLS PRODUCT. They also deal with how the role might be split up into an ARCHITECTURE TEAM. They emphasize the necessity that ARCHITECT ALSO IMPLEMENTS to stay up to date [Coplien Harrison 2005].
Depending on the team size the outside view may be supported by a dedicated technologist who researches new technologies. [Völter Kircher 2008b]

## 5 Acknowledgements

My thanks go to my shepherd Michael Kircher for his invaluable domain and pattern expertise and the right hints at the right time. I would also like to thank Allan Kelly for inspiring me to make the first steps towards this paper and the participants of my workshop at EuroPLoP 2009, especially Andreas Rüping, for their incredibly helpful comments.

# 6 References

Coplien, James O. and Neil B. Harrison 2005. *Organizational Patterns of Agile Software Development.* Upper Saddle River, NJ: Pearson Prentice Hall.

Kelly, Allan 2008. *On management.* http://www.allankelly.net/static/writing/OnManagement/OnMngm1-Constraints.pdf. 2008-07-03 (Last accessed 2010-1-21).

Kelly, Allan 2009a. *On Management #4: Caveat Emptor*. http://www.allankelly.net/static/writing/OnManagement/OnMngm4-CaveatEmptor.pdf. 2009-02-20 (Last accessed 2010-1-21).

Kelly, Allan 2009b. *On Management #5: The Product Manager role*. http://www.allankelly.net/static/writing/OnManagement/OnMngm5-ProductManager.pdf. 2009-04 (Last accessed 2010-1-21).

Kelly, Allan 2009c. *On Management #6: The Business Analyst's role.* http://www.allankelly.net/static/writing/OnManagement/OnMngm4-CaveatEmptor.pdf. 2009-06-08 (Last accessed 2010-1-21).

Schwanninger, Christa, and Michael Kircher 2009. *Patterns for Product Line Engineering.* In: Proceedings of the EuroPLoP 2009 conference, Irsee. In: CEUR-WS online repository, http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/.

Völter, Markus and Michael Kircher 2008a. *Roles in Software Engineering I. Episode 110*. http://www.se-radio.net/podcast/2008-09/episode-110-roles-software-engineering-i. 2008-09-18 (Last accessed 2010-1-21).

Völter, Markus and Michael Kircher 2008b. *Roles in Software Engineering II. Episode 112*. http://www.se-radio.net/podcast/2008-09/episode-112-roles-software-engineering-ii. 2008-09-28, (Last accessed on 2010-1-21).

Wikipedia 2009a. *Project Management Triangle*. http://en.wikipedia.org/wiki/Project_management_triangle. 2009-03-22 (Last accessed on 2010-01-21).

Wikipedia 2009b. *Pareto Principle*. http://en.wikipedia.org/wiki/Pareto_principle 2009-03-22 (Last accessed on 20010-01-21).