Ignored Architecture, Ignored Architect

Klaus Marquardt

Email: pattern@kmarquardt.de

Being a software architect is fun. Besides the obvious challenges in technology, architects often face interesting team situations, and can have more influence and impact on the project's success than their manager.

Let's face it: Having to deal with a software architect can be, well, also funny – but not necessarily constructive. Any influential person is highly visible and evokes emotional reactions. Personality clashes can lead to mutual ignorance, and team members might deliberately choose to ignore the architect at least occasionally. Sometimes the architecture itself gives good reasons to ignore it. Sometimes the "architect" is not really the architect, and other developers or managers are answering the tough questions, or worse: nobody does.

This paper aims to help architects and other project participants find out what is going on in the project and to explore whether and why the architect or the architecture is being ignored. A variety of counter measures are available, some of which should be applicable even in the most difficult situations.

Introduction

There is an expert role in software development that involves a high level of communication, a visionary view on the system and a broad understanding how to build it. Where this role has a name, it is typically called software architect[1]. A software architect also knows how to bring the vision to life, and takes responsibilities to educate people and integrate the system until the vision has become reality.

Real projects might encounter deviations from this ideal. When projects fail, the reasons are mostly attributed to non-technical issues [*CACM02*]. This paper examines whether a tendency of the development team and management to ignore the architecture or the architect can be such an issue. It aims at giving hints for all project participants to identify and name yet hidden problems.

This paper tackles a difficult topic of primary concern to the architect: an assessment of the architecture and the architect. She needs to interpret early signals about her own work and role. Filling one of the lead positions of a project can make it difficult to learn from others. As few people are able to reflect upon their own role and behavior, most issues can better be identified during retrospectives [*Kerth01*]. When these retrospectives are conducted during the project's course, the architect can initiate effective counter measures and improve the situation before somebody complains about obvious deficiencies [*Lind+03*].

All of the situations are described from the viewpoint of a software architect. However, some insights are more obvious to the people who have to deal with an architect, rather than to the architect themselves. Most measures require joint understanding and effort of managers and architects. Insightful observers, especially architects, senior developers, consultants and managers, will find hints to improve their current or future projects.

About this Paper

This paper aims at helping architects and other project participants to find out what is going wrong in their project and why, and what they can do about it.

For this purpose the problem is described in a form appropriate for a medical disease, as a diagnosis. Known resolutions or measures are introduced as therapies. Similar to the medical world, a complex problem might have more than one solution, and a solution might help to solve more

[1] The literature distinguished software architects at application, system, and enterprise scope, and categorizes into functional or application, infrastructure or technical, integration, and security architecture. The described role is independent of these dimensions and applies potentially to all key developers, independent of a job title.

than one particular problem. Diagnoses and therapies do not stand on their own but are cross-linked back and forth.

Following this medical metaphor brings some unique features. A wealth of vocabulary becomes accessible. The metaphor also shows the limitations we face: none of the presented solutions might actually cure a particular system. Some therapies are only effective when applied preventively, others are merely palliative or might at best lead to a remission.

In this paper, diagnosis names are written in UNDERLINED SMALL CAPITALS and therapy names in SMALL CAPITALS. Names used but not listed herein are marked (↗) and can be found in the references. Both diagnoses and therapies follow their own pattern formats including sections that contribute to the medical metaphor.

- The description of a diagnosis starts with a small summary and a picture. Symptoms and examination are discussed and concluded by a checklist that enumerates mandatory, typical and occasional symptoms. A description of possible pathogens and the etiology closes the diagnosis.

  Each diagnosis comes with a brief explanation of applicable therapies. This includes possible therapy combinations and the kind of effect: curative, palliative or preventive. Where available, treatment schemes are described that combine several therapies. These are suggested starting points for a successful treatment of the actual situation.

- Therapies are measures, procedures or other medications applicable to one or several diagnoses. Their description, in full following a canonical pattern form with medical and pharmaceutical extensions, is abbreviated in this paper.

Diagnosis: Ignored Architecture[2]

The project team pays lip services to its architecture, but the project is organized around other criteria, the team commits to no architecture and raises obstacles to any architectural attempts.

You are the dedicated architect of a project. Having said that, the actual role behind the job title does not meet your expectations. The range of decisions that you are empowered to make is limited. Before anything gets implemented or changed, most team members ask the project manager for approval.

The project manager emphasizes that the project needs a sound architecture; however, his goals are only related to the timely release of the first software version. Whenever issues arise that are relevant to his goals, he makes decisions himself. Neither the project team nor the management has a clear understanding what they expect from the architecture and exactly why they need it.

> *Effective managers tend to plough through everything perceived as an obstacle, unless it is sufficiently resistant in which case it needs to be negotiated. The project's own architecture and its consequences can become an obstacle e.g. by increasing the amount of work required before the next milestone. In that case the architecture will be overrun or undermined whenever possible, just as the rat ploughs through the walls of the maze negating their original purpose.*

[2] A previous version of this diagnosis can be found in [*Marq03b*]

The team consists of several specialists in the application domain. These specialists do not merely create software but work closely with product management, marketing, and customers or their representatives. They view software architecture as an activity that does not add value. Discussions with the architect are welcome as long as they clarify issues but do not cause changes to their existing software or increase the effort to deliver the domain functionality to the system. They perceive the rules and obligations imposed by the architecture as an unfriendly intrusion into their home area. When architectural changes are discussed, the domain specialists start to worry about the impact on the progress of their assigned tasks.

Being the software architect becomes frustrating over time. While all project team members rely on your knowledge and insight, your contributions are hardly acknowledged. Even though a sound architecture reduces the total cost of ownership, management is more interested in short term goals. Despite the fact that a sound architecture enables parallel and independent development, agreement on the necessary infrastructure can not be reached and it is implemented and used merely half-heartedly and partly avoided.

Finally when changes become unavoidable, however well motivated, it is found that the internal structure of the applications does not live up to the principles that the architecture has demanded. Measures to limit the effect of changes, such as the open-closed principle [*Meyer97*] or dependency inversion [*Martin96*], have not been taken. The changes to the system require tremendous effort and become the nightmare that everybody was afraid of. Even though common software structures and shared code would have increased system usability through a common style, look and feel together with easier maintenance, each application specialist has resorted to his own style, opinions and habits within his private area.

Symptoms:

- The architect has little or no decision authority.
- Short-term goals are in the focus of management.
- The purpose of the architecture is not obvious to the project participants.
- There is a strong focus on domain and application knowledge.
- Architecture is perceived an intrusive obstacle to progress.
- The structure of the software does not support the intended system qualities.
- Little attention is paid to consistency and reuse across different project areas.

There are different pathogens responsible for IGNORED ARCHITECTURE. They all result from clashes of expectations or interests that have not been resolved.

- A technically ↗INSUFFICIENT ARCHITECTURE is one possible reason to ignore the architecture. In dramatic cases such ignorance might even increase the project's chances to deliver on time or at all.
- The architecture is usually closely associated with the person advocating it and so their credibilities are mutually related. Team members ignoring the architect themselves is a common cause of an IGNORED ARCHITECTURE. A software architect will always have to cooperate with managers and most developers and needs to have a strong influence.
- Another cause of an IGNORED ARCHITECTURE is a poor architectural process that fails to make the architecture explicit, as in ↗IMPLICIT ARCHITECTURE.
- Bad decisions that an architect has made in the past, are likely remembered by the team and management.

Finally, the architecture needs to both address and manage the stakeholders' expectations and so it is mostly about communication. The architect's communication is almost always an area that can be improved regardless of any other underlying problems. A typical example is the communication with the experienced developers on the team. This group is allergic to an architecture passed with a "throw it over the wall" attitude, but conversely they dislike their software portions being micro-architected.

Differential Diagnosis

It is important to clarify the underlying cause of an IGNORED ARCHITECTURE so that the therapies can be chosen. If the problem originates from an IGNORED ARCHITECT, you would need to address all personal issues first.

- Do developers go to the architect to ask for clarification, and invite her to discuss their ideas? That would indicate an ↗IMPLICIT ARCHITECTURE or IGNORED ARCHITECTURE, but not an IGNORED ARCHITECT.
- Is the architect perceived a nay-sayer? That would indicate an IGNORED ARCHITECT.
- Is the architect an accepted software developer? That would indicate an IGNORED ARCHITECTURE rather than an IGNORED ARCHITECT.
- Is the architect able to dedicate time for architectural work? If not, that would indicate an ↗IMPLICIT ARCHITECTURE and possibly an IGNORED ARCHITECT.
- Is the system perceived as complex? That would indicate ↗INSUFFICIENT ARCHITECTURE or IGNORED ARCHITECTURE.
- If external observers can spot problems with the architecture just by looking at available documentation, that would indicate ↗IMPLICIT ARCHITECTURE or ↗INSUFFICIENT ARCHITECTURE.

The first set of therapies focuses on increasing the acceptance of the architecture – not of the architect, but that may follow as a consequence. The key to acceptance is the value that other participants receive. An effectively communicated BIG PICTURE ARCHITECTURE is a good way to establish the key architectural elements and feedback mechanisms – provided the architecture is technically sound and sufficient. Together with VISIBLE QUALITIES, a subconscious awareness of the architecture can be achieved that avoids some of the frustration on both sides.

If the project has DEFINED ARCHITECTURAL GOALS, it is easier to recognize that the architect has delivered these with the defined architecture. Such goals also help to communicate to developers and management why the project benefits from the architecture and how different elements of the architecture relate to the goals. Make sure you EXPOSE your PROCESS to avoid the management impression that an effective architecture happens by accident. It also helps when you need to get buy-in for your activities or for the demands you make on the developers.

An effective and influential architect needs to combine technical and social skills. This valuable competency needs to be acknowledged by employers. In some way the ARCHITECT IS REWARDED, or rather should be rewarded. Otherwise the company basically gets what it pays for – not enough in this case. There are many ways to embody this principle, such as a technical career path; a goal related bonus policy; or special project opportunities and education.

Initial Treatment

Before you start to diagnose much and wait for serious symptoms, there are a number of therapies that are never wrong, and that help both with IGNORED ARCHITECTURE and IGNORED ARCHITECT. You can initiate them preventively, or when the first signs of problems emerge.

- A BIG PICTURE ARCHITECTURE is helpful in any project situation and needs to be installed as early as possible. Even spending minimal effort to establish seemingly trivial rules helps to get the project and its treatment started.
- A DEFINED NEGLECTION LEVEL prevents you from digging too deep into areas that are uncritical to the project's success, and empowers developers to take over architectural responsibility in their work area.
- DIVIDE ET IMPERA is the butter and bread task of all architects. Never forget the symmetry to the final activity, integration.
- VISIBLE QUALITIES is a great communication means to transport what is really important.

Therapy Overview

	Applicability	**Effect**	**Related therapies**
BIG PICTURE ARCHITECTURE	Any time during the project, preferably early.	Palliative. Supportive.	The essence of ↗DIVIDE ET IMPERA.

	Applicability	**Effect**	**Related therapies**
DEFINED NEGLECTION LEVEL AKA NEGLECT THE LEVEL BELOW	Preferably early in the project.	Palliative. Remission possible.	
VISIBLE QUALITIES	Any time during the project, preferably early.	Palliative. Supportive.	
DEFINED ARCHITECTURAL GOALS	Any time during the project, preferably early.	Remission possible.	
EXPOSED PROCESS	Preferably early in the project.	Remission possible.	
ARCHITECT IS REWARDED	Any time during the project. Requires management support.	Preventive. Remission possible.	

Suggested Treatment Schema

In psychology, a major criterion that governs whether a patient is worth treating is the personal degree of suffering. While an architect might suffer from her work being ignored, she could indirectly strengthen her position if she manages to make others suffer as a result of them ignoring it. However, this might take too much time to be considered satisfactory – and it contradicts the positive and productive work ethics that most people value from software architects.

The steps you can take are very much dependent on the personalities involved, and so no definitive or proven treatment schema can be offered.

A good starting point seems to be an EXPOSED PROCESS when it comes to creating an architecture, so that development team and management can understand what kind of work the architect does. While the EMPHASIZED ARCHITECTURAL BENEFITS could be more than enough to get the required level of buy-in, most architecture stakeholders can be won over with a concise set of DEFINED ARCHITECTURE GOALS that match mostly with their own goals.

To depend the architect's salary and career on goals that match mostly with those of the direct manager is also one of the more elegant possibilities to implement ARCHITECT IS REWARDED. When the architect is perceivably interested in getting her points across, she will be more open to focus on the important aspects – which is key to a successful architecture. BIG PICTURE ARCHITECTURE and a DEFINED NEGLECTION LEVEL are helpful to avoid micro-architecting the developers.

Big Picture Architecture[3]

Define a compact architecture outline and make it become part of the project jargon. The software architecture outline must cover the top level of the technical structure, the key domain abstractions, interfaces and interactions, and the order and stability of development.

You need to illustrate the most important issues in a simplified way. The simplifications should match with the developers' experience and scale up to a large extent. Examples from the technical domain are a Document-View or a layered architecture. The domain model typically comprises less than 20 classes and their relations. The most relevant interfaces can also be categorized and sketched. The order and stability of development can be expressed in packages and their dependencies.

Metaphors can help you to outline parts of the BIG PICTURE ARCHITECTURE with a few words that evoke guiding associations. Graphics and diagrams are typically most appropriate for components, interfaces and interactions. Resist the temptation to use buzzwords and technology phrases as they do not help you to distinct your unique system.

In the absence of other common vocabulary, an architect introduced an extensible architecture with the notion of "colored boxes". Each box represented an extension component, the color indicated its particular purpose with respect to techniques and application. Within each component, a Model-View-Control pattern (MVC) came into place, and within the MVC participants one more level of substructure was defined. After some time, the vocabulary and dependencies became obvious to the team, and each developer was able to place a given class at the correct logical location – or to tell what was wrong about it.

Defined Neglection Level[4]

Decide on the level of detail that is controlled by the architect, and neglect all levels below that. While you are serious about the architectural rules at high levels, relax your control on all levels below the one that is of architectural interest. You might define a different neglection level for each area of architectural interest.

[3] A full version of this therapy can be found in []
[4] A full version of this therapy can be found in []

While there is no commonly valid rule across all projects and domains, here are some starting points how to determine your initial level of neglection. Be aware that this neglect is subject to change during the project, according to identified risks, workload, and customer needs.

- **One Level Below Basic Components**. As a first approximation, look at the components you defined in DIVIDE ET IMPERA and take care of no more than one level below that one.
- **Listen**. Unless the team is particularly inexperienced, they usually know quite well where critical points are, what aspects they need to care for, and what level of design responsibility they are capable of.
- **Don't Interfere Experience**. In the areas assigned to the most experienced developers, you may neglect most activities – provided you have a common understanding of priorities and architectural goals.
- **Be Arrogant**. After accepting other people's competence and wisdom, you need to be sufficiently arrogant to go for your own viewpoint in case of any doubt.

A large Plug-In based project introduced design conventions that banned bidirectional or cyclic dependencies among packages. These rules were checked with automated tool support. Within a package, dependency cycles were explicitly accepted. This allowed for a refactoring when cycles were appropriate to the solution, and kept the high-level dependency structure manageable.

Visible Qualities[5]

Make your system's internal qualities visible. Similar to sound risk management practice, maintain a list of your top five qualities. Define measures to achieve them, and determine frequently to what extent you have reached your goal.

The key issue is to raise awareness amongst the team and in management of the existence of these qualities and their relative importance to the success of the project. Ask the team come for a list of possible qualities and discuss their value and advantages, especially when the internal system qualities are unbalanced. The team should order them according to their priority, taking into account the expectation of the project's sponsor, the daily work of the team, ans the effects on the maintenance and costs of ownership. Do not

[5] A full version of this therapy can be found in [] Performitis

mind if the qualities you perceive as important are not the topmost – you will go through the list every week or two and re-evaluate.

"Most team members were new to object-oriented design, so we discussed a lot about the promised qualities it should deliver and how to achieve them. We started to do ↗JOINT DESIGN at the white board, and explored different alternatives how extensibility could be reached, how testability could be increased, and what amount of decoupling would require what effort. When the team size increased, ↗DESIGN REVIEWS became an essential part of the project. Initially I participated in most, and we established an ordered catalogue of criteria to check. With this catalogue, the process was accepted and carried by the team. Closer to the end of the project, the team decided to focus on other issues and reduce the ceremony level of the design reviews. By that time, the project lasted for more than two years; all team members had significant expertise and shared a common sense of the important qualities to take care of, and how to address them."

Defined Architectural Goals

Define goals that the architecture should meet, and get management agreement. These goals should directly support the goals that the project manager follows, and extend them towards a sustainable architecture minimizing the total costs of ownership. It is important to negotiate the goals with all clients (development and management) explicitly to avoid subsequent conflicts.

Openly communicate the purpose of the architecture, and what the project gains when compared to having no explicit architecture. Match the purpose to the overall technical and business goals of the project.

Exposed Process

Explain for each activity why and how you do it.

To some project stakeholders, architecture may seem like an artist work with little relevance to the project success. Developers may feel pampered and controlled when the architect demands adherence to defined policies.

To explain why you take initiative helps to overcome this. Work in the open to avoid that outsiders may assume a hidden agenda or other politics.

Exposed Process needs to be combined with other therapies that help to create acceptance and trust, such as VISIBLE QUALITIES, DEFINED ARCHITECTURAL GOALS, and ARCHITECT ALSO IMPLEMENTS.

Architect is Rewarded

Compensate the architect based on the technical and business goals of the project. This can take the form of a success-based salary bonus compensation similar to those used for accountable project managers, a technical career path, or any other benefit of interest to the architect such as education or more choice in their next assignment.

Provide opportunities for acknowledgement within the company that are independent of the career path for managers. Make these opportunities attractive in terms of both status and money. Prevent technically excellent people from becoming managers or leaving to join competitors because they see no avenues for personal growth.

Acknowledge the key developers by raising their status and distinguishing them from less experienced technical staff.

Diagnosis: Ignored Architect

The project organization has established a software architect role, and assigned a person. However, the architect does not have a strong voice in important decisions, and his advice is not considered.

„doctor, people are ignoring me.“

„next please!“

You are the dedicated architect of a project. Having said that, the job title does not seem to give you any influence within the project. More weight is given to the opinions of other developers and they are more frequently asked for advice during the course of the project.

The development team is aware of the need for a consistent design. Frequent design meetings have been established, or the team convenes ad hoc design sessions. While you are a frequent guest at those meetings, you neither moderate nor instigate them – because the design issues are not deemed to be at your level.

> *"In an international project we had some difficulties in establishing a common understanding in system architecture. Finally we formed a team of three: an external consultant, one architect from the other location, and me. The other team was living in its familiar habitat and was under management pressure to complete early. All discussion and concept change that the architecture team initiated was propagated to both teams. While my team stayed in close contact with me, or vice versa, all of us still learning, the other architect lost attention – because of the new stuff was deemed irrelevant to that team's immediate needs. They established a design team and occasionally invited him to give presentations, but did not approach him for the ad hoc decisions.*
>
> *"Several months later the project was cancelled due to overall failure to cooperate, integrate, and deliver. The only reusable remainder of the project was a functional team at this location,*

> *and a wealth of concepts and implementation that served as a starting point for the next project. At the follow-on project, that software part formed a solid foundation that did not happen to be on the critical path of the overall effort once."*

In an attempt to compensate, you might try to increase your influence by raising the tough questions, intervening in decisions, and prescribing what others have to do. However, this effectively deepens the gap as you are still not approached and some developers actively avoid you. The manager does not interfere, as she perceives that the team functional and is taking care of the architecture by itself.

A common way for a manager to ignore a software architect is to perceive and define him as a nay-sayer, a person who tries to prevent progress – acting against common wisdom and without decision authority. Seasoned developers often suffer a particularly allergic reaction against a "throw it over the wall" attitude that arises when the formal technical work of the architect is OK but the communication of the results is insufficient. In other organizations, this might be the exact way that architects are expected to behave.

Symptom checklist:

- The architect is not approached to answer questions.
- Other team members' have a higher impact on decisions.
- The architect does not lead design efforts.
- Management does not care who fills which role.
- The architect is perceived a nay-sayer.
- The architect is accused showing a "throw it over the wall" attitude.

There are two major pathogens responsible for IGNORED ARCHITECT, corporate culture and personality traits. A common factor is that they are both dependent on the architect's personality, but that he himself has a limited ability to influence them.

The architecture is usually closely associated with the person advocating it, and so the credibility of one is related to the credibility of the other. Team members ignoring the architect themselves is a common cause of an IGNORED ARCHITECTURE. A software architect will always have to cooperate with the project's management team and most of the developers, and so the architect needs to ensure that they have a strong personal influence on these groups.

A cure for an inappropriate personality is beyond the scope of this paper. The suggested therapies can at best be palliative, and possibly lead to remission in the long term.

Due to the close relation between the architect and the architecture, it is wise to provide a sound architecture and promote it in a balanced way. Check the therapies suggested in IGNORED ARCHITECTURE and ↗INSUFFICIENT ARCHITECTURE, and the hints in the Initial Treatment box.

Avoiding obvious mistakes in the architecture does not make you an accepted project team member. PART-TIME ARCHITECT allows you to gracefully step back a bit and limit your personal exposure, while ARCHITECT ALSO IMPLEMENTS can bring some peer acknowledgement. You are not in a position to start ↗ARCHITECT ALSO COACHES, but DEFINED ARCHITECTURAL GOALS help to get developers' buy-in. It might also be wise to have an external MENTOR join the team who could mitigate personal, process related and technical issues

Therapy Overview

	Applicability	**Effect**	**Related therapies**
PART-TIME ARCHITECT	Any time during the project, preferably late. Requires management support.	Remission possible.	Combine with ARCHITECT ALSO IMPLEMENTS.
ARCHITECT ALSO IMPLEMENTS	Continuously during the project.	Supportive.	Counter indicated before DIVIDE ET IMPERA or BIG PICTURE ARCHITECTURE are in place.
DEFINED ARCHITECTURAL GOALS	Any time during the project, preferably early.	Remission possible.	
MENTOR	Any time during the project. Requires management support.	Possibly curative, but with latency.	

Part-time Architect[6]

Allow yourself only a limited time to care for the system in the architect's role. Let go as soon as the system can prosper with less care, and stay in control of only the very essentials of the architecture.

This can be done in several ways that also give other advantages. You could spend more time doing actual coding (as in ARCHITECT ALSO IMPLEMENTS), helping to finish the system. You could move forward to another project or other unrelated tasks. You could invite other team members to take some architectural responsibilities, giving them adequate career opportunities.

When your time as full time architect of the particular system has passed, strange things will happen. Colleagues and managers will pay less attention to your suggestions, some colleagues might try to occupy your position, your focus on structure might lead to over-design and unnecessarily slow down the development pace – in short, you potentially harm the system and your career.

The difficult part is to identify the right moment to let go. In a similar way to movie stars and politicians, it is possible for observers to tell that the moment has already passed. It takes a very self-conscious and ego-less person to declare herself partially superfluous, and to admit that the project would run faster without her work. If you initiate a change actively, you avoid the risk that others perceive this lessening of effectiveness first, and you can decide yourself to go back into full time mode when integration problems arise or major extensions are planned.

For a significant functional extension of a medical product, an architecture team was formed of four developers – one of each development team. One developer was the initial architect of the product. All architects remained the lead developers of their respective teams. This lead to a very quick exchange of experience, and the team established a productive and informal working basis. Product development proceeded smoothly and combined a successful follow-on product with a consistent architectural vision of the complete system. The four architects found a productive way to cooperate. While the initial architect acted as a MENTOR and improved his team and change management skills, the developers also raised their architectural skill level.

[6] A full version of this therapy can be found in [*Marq02a*]

Architect also Implements[7]

Make the architect a developer, a primo inter pares[8] - in addition to her architectural tasks. Assign development tasks to her that are influenced by architectural decisions. It is common practice to let the architect implement the most difficult system parts, but take care to keep him off the critical path and plan some slack time for unforeseen architectural issues.

Leading by example brings a lot of short and long term benefits. You get a consistent system and a well educated development team, and a number of high quality feedback loops. This will most likely increase your system's development speed, its internal quality, and decrease its maintenance costs. On the other hand, an architect switching between different tasks might be less effective, and some of the tasks can not be completed as quickly as usual.

In huge industrial projects, it is fairly uncommon to have an architect do anything but architecture – whatever your organization defines this term. However, ARCHITECT ALSO IMPLEMENTS is a common policy in smaller teams. When different teams and organizations begin to cooperate, for example in projects crossing geographical and cultural borders, stating this policy explicitly is of major importance. Otherwise, misunderstandings will occur with respect to roles, responsibilities, influence and availability that can seriously hinder a successful cooperation.

Defined Architectural Goals

Define goals that the architecture should meet, and get management agreement. These goals should directly support the goals that the project manager follows, and extend them towards a sustainable architecture minimizing the total costs of ownership. It is important to negotiate the goals with all clients (development and management) explicitly to avoid subsequent conflicts.

Communicate openly what the purpose of the architecture is, and what the project gains compared to having no explicit architecture. Match this purpose to the overall technical and business goals of the project.

[7] A classic, first documented in [*Copl95*], then in [*CoHa04*]. A medical version is in [*Marq02b*]

[8] Latin: first among peers

Mentor[9]

Bring in an external expert who is experienced with large systems and is accepted by your architects. The mentor's task is to teach your team good habits and working style so that it can succeed next time on its own.

Given that the architects themselves need mentoring, the mentor could become a consulting lead of the architecture team, in order to encourage the architects to make the right decisions on their own. His working knowledge must cover multiple working styles and be sufficient to allow the team to also make decisions. He judges whether the team decision can lead to success, and intervenes when the team diverges from a path that could possibly be successful. He leaves early enough to allow other team members to take over responsibility and pride of ownership.

The key difficulty about mentoring is to detect that you need it at all. You can only spot the need when you retreat from your daily work. But this distant look is among the working habits that a mentor is expected to teach you, and probably the least experienced team members are the last ones to recognize this. So those teams who know by themselves they need a mentor, might be those who need him least.

The next key issue is more obvious: how do you find a good mentor? Do not look for star developers; more important are communication skills and a mindset to help other people.

[9] A full version of this therapy can be found in [*Marq02a*]

Acknowledgements

Wolfgang Keller shepherded this paper to EuroPLoP and insisted that he understands it before bothering a workshop, and would not fall asleep during the lecture. Further thanks to Andy Longshaw for his support in language precision and elegance. If you find flaws, I introduced them after his review.

Thanks to the workshop participants at EuroPLoP for their comments: Lise Hvatum, Allan Kelly, Andy Longshaw, Alan O'Callaghan, Jorge L. Ortega-Arjona, and Anne Villems. Andy Schneider provided valuable feedback after the conference.

"There is no success like failure, and a failure is no success at all" [*Dylan65*]. Thanks to the unnamed colleagues who enabled me to learn from my own mistakes, and allowed me to learn from theirs. More thanks to those who agree that you need to have success to learn how to do it again.

The copyright holder of the picture illustrating IGNORED ARCHITECTURE could not be determined; hints are welcome.

References

CACM02	Article in CACM raising the notion that the team is a key deliverable of a project. 2002
CoHa04	James Coplien, Neil Harrison: Organizational Patterns of Agile Software Development. Prentice Hall 2004
Copl95	James Coplien: A Generative Development-Process Pattern Language. In: Pattern Languages of Program Design, Addison-Wesley 1995
DeMa01	Tom DeMarco, Timothy Lister: Peopleware. Second Edition, 2001
Dylan65	Bob Dylan: Love Minus Zero / No Limit. In: Subterranean Homesick Blues, CBS Records 1965
Foote+00	Brian Foote, Joseph Yoder: Big Ball of Mud. In: Pattern Languanges of Program Design 4, Addison-Wesley 2000
Kerth01	Norman Kerth: Project Retrospectives. Addison-Wesley 2001
Lind+03	Lowell Lindstrom, Kent Beck: XP and Culture Change Part II "It Gets Worse Before It Gets Better: Changing to XP". In: Cutter IT Journal, February 2003, Volume 16, No. 2
Marq02	Klaus Marquardt: Architecture and Organization: Structure, Problems, and Solutions. In: Proceedings of EuroPLoP 2002
Marq02a	Klaus Marquardt: Supporting the Software Architect: Selected Patterns Covering Different Perspectives. In: Proceedings of EuroPLoP 2002
Marq02b	Klaus Marquardt: Patterns for the Practicing Software Architect. In: Proceedings of VikingPLoP 2002
Marq03a	Klaus Marquardt: Performitis. To be published in: Proceedings of EuroPLoP 2003
Marq03b	Klaus Marquardt: Neglected Architecture. To be published in: Proceedings of VikingPLoP 2003