

# What do software architects do?



*Philippe Kruchten*

Electrical and Computer Engineering

University of British Columbia

Vancouver, BC, Canada

April 2006

V1.0

“—Mr. Beck, what is software architecture?” asked a participant at an OOPSLA workshop in Vancouver in the fall of 1992. “—Software architecture?” replied Kent, now famous for being the father of xP, “well, it is what software architects do.” (Chuckles in the audience.) “—So then, what is an architect?” “—Hum, ‘software architect’ it’s a new pompous title that programmers demand to have on their business cards to justify their sumptuous emoluments.”

In the following four years I was going to lead a rather large team of software architects and I will ask myself that very question often: “what do architects really do?” and was asked this by my management and my customers. Since then I have seen many architecture teams in many countries, in companies of all size and various domains, and I have witnessed a wide range of good, not-so-good and really bad answers to this question.

## ***Architects design the architecture, no?***

The first obvious answer that comes to mind is:

- *Software architects should design, develop, nurture, and maintain the architecture of the software-intensive systems they are involved with.*

It is not a simple, satisfactory answer, since there is no universally accepted definition of what software architecture is. And this would bring us back to the original question.

So a software development organization, based on its context: domain, culture, assets, staff expertise, etc. must come up with some delimitation of what constitute software architecture, and what is beyond software architecture, and this definition, this “thin line in the sand” that separates architectural decisions from all other design decisions, including the detailed ones captured in the code, must be made visible to all parties involved. And may have to be revisited, redefined, adjusted as an architecture emerges, and as the expertise of the organization grows, or the size of the team grows.

Let us assume for now that the responsibilities of the architects is the part of the design and of the design decisions that have long-last impact on some of the major quality attributes of a software intensive system: cost, evolution, performance, decomposability, safety, security, etc, and still able to support the functionality expected by its end user.

Then this is what software architects should be focused on, this is what software architects should do: making design choices, validating them, capturing them in various architecture related artifacts.



But things are not so simple. There are several “antipatterns” that will make a software architect or software architecture team miserably fail if they were to only do this: design the architecture.

### **Antipattern: Creating a perfect architecture, for the wrong system.**

A software architect that is not communicating regularly with the customer, the end users, or whomever represent them (the product manager) is likely to miss the target, in particular as the target is moving, or rather, as the target is only gradually understood.

### **Antipattern: Creating a perfect architecture, but too hard to implement.**

A software architect who does not understand the (maybe limited) skills, capability and experience of the implementation team(s) that will continue and finish the work will create enormous level of stress and frustration, and likely not deliver a quality product in time. The architectural effort has turned into a computer science research project.

### **Antipattern: Ivory tower**

The worse combination is the architecture team that lives isolated in some other part of the organization—another floor, another building, another country—and who comes up after some months with a complete architecture, out of the blue. To their complete surprise, they will experience rejection: an apparent misfit on both front, functional and implementation. This is especially the case if the developers (the non architects) had a few months to make some progress and they have in some ways made some architectural decisions, under some other name. A special case of this antipattern is the architecture group that only scouts technologies and provides recommendations to other groups, but is not making design decisions and is not accountable, as I have witnessed in two large telecommunication companies, the architecture watch..

There is another issue that cannot be completely ignored; it has to do with *whom* you have chosen to be the architects. It is very likely that you have in this role some of your most talented staff: good at manipulating abstractions, wide experience of a range of systems and technologies, good communication skills, domain knowledge, etc. and you may want to use some of these skills for other tasks than just building architectural views. You want them to speak to the new prospective customers, to show off the organization technical expertise, you want them to help this or that team that experiences a difficult technical issue, you want them to review the architecture of another project, to take part of a due diligence process to acquire a company, to present papers a conference to strut your stuff, or merely extinguishing some nasty fire. But if you are not careful, this leads to another “antipattern:”

### **Antipattern: The absent architects**

No or little architecture design progress is made: the architects are always away, doing fascinating things, or fighting fires. It is very easy to slip in this mode, especially after some initial good progress and early successes, which brought some fame on the architects.



## ***Roles and responsibilities of an architect (or an architecture team)***

The role and responsibilities of an architect can be usefully be captured in some kind of a team “charter” or “mission”, that must be adjusted to each organization or project. The list below is derived from the charter of a large team I led in the mid-1990’s (Kruchten, 1999).

1. Defining the architecture of the system  
All the usual technical activities associated with design. Understanding requirements, qualities, extracting architecturally-significant requirements, making choices, synthesizing a solution, exploring alternatives, validating them, etc.; For certain challenging prototyping activities, the architects may have to use the services of software developers and testers.
2. Maintaining the architectural integrity of the system  
Through regular reviews, writing guidelines, etc. and presenting the architecture to various parties, at different levels of abstraction and technical depth.
3. Assessing technical risks  
These role is in support of project management, but on very technical risks, managers may not have the expertise to identify and
4. Working out risk mitigation strategies/approaches
5. Participation in project planning
6. Proposing order and content of development iterations  
For many effort estimation aspects, or for the partition of work across multiples team, managers need the assistance of architects.
7. Consulting with design, implementation, and integration teams  
Because of their technical expertise, architects are drawn into problem-solving and fire-fighting activities that are beyond solving strictly architectural issues.
8. Assisting product marketing and future product definitions  
The architects have insights into what is feasible, doable, or science fiction and their presence in a product definition or marketing team maybe very effective.

As you see, beyond item #1, many activities involve some other party: project management for example, and are not merely focused around the architecture, the design, the architectural prototype.

We need also to keep in mind that the good architects should bring a good mix between domain knowledge, software development expertise, and communication skills.

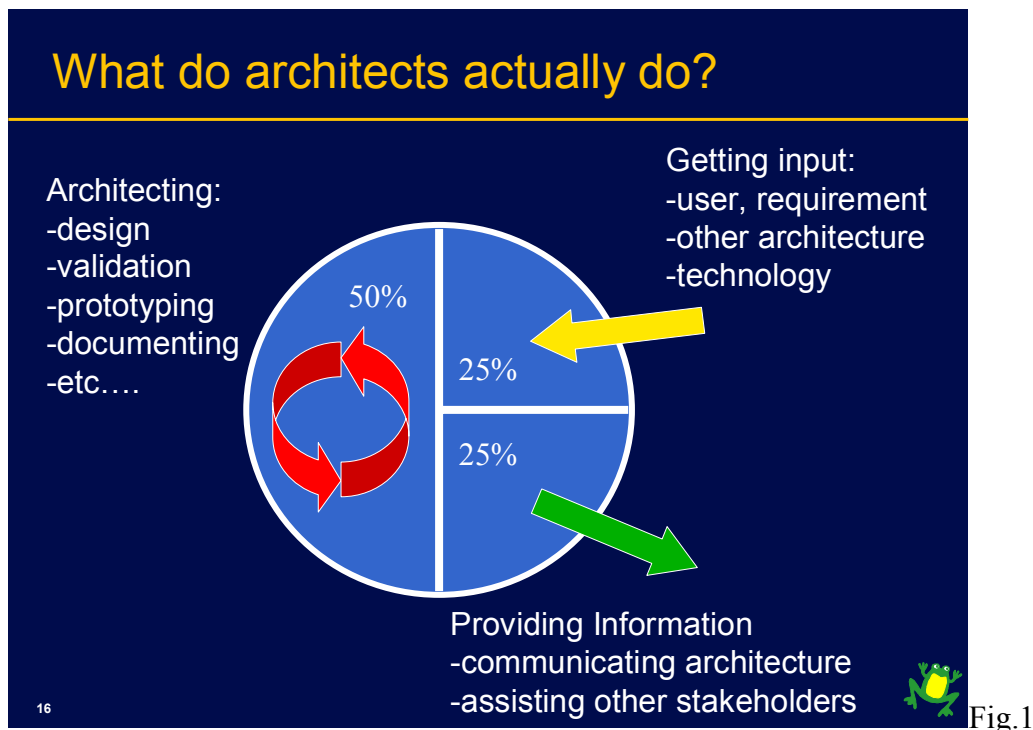
Once we identify (and possibly refine) the long list of what we expect the architects to be doing, the next question is: how do we keep a good balance between all these activities. How do we avoid the temptation to always, day after day, week after week, solve the most urgent problem, or the most interesting problem, or extinguish the latest fire? (The squeaky wheel syndrome) Or conversely bring forward the question: do we have the right people with the right expertise in our current software architecture team?

## ***Allocating time***

To avoid falling in any of the traps or antipatterns mentioned above, and to help maintain a delicate balance between all the forces that an architect is submitted to, I came up in the



mid-1990's with a simple time-management practice, summarized in the figure below, extracted from (Kruchten, 2004).



It assumes that you are collecting timesheets, to account of where the architects spend their productive time. In general, globally across the whole architecture team (if you have more than one architect), and on average over the lifecycle, my recommendation is that the architects should have:

- **Internal focus:**  
About 50% of their time focused on architecting *per se*: architectural design, prototyping, evaluating, documenting, etc.
- **External focus:**  
About 50% of their time interacting with other stakeholders. This in turn has two facets:
  - **Inwards:**  
25% getting input from the outside world: listening to customers, users, product manager, and other stakeholders (developers, distributors, customer support, etc.). Learning about technologies, other systems' architecture, and architectural practices.
  - **Outwards:**  
25% providing information or help to other stakeholders or organizations: communicating the architecture: project management, product definition..

The numbers come from my experience in managing a 10 person architecture team in 1992-1995. This apparently crude, out of the cuff, partitioning of time has drawn lots of comments, feedback, and push-backs from my colleagues and customers since then, but in the end, unless your situation is really very special, I have not been convinced by any



substantive evidence to change the numbers [50, 25, 25] over the last 10 years. (But as often in software engineering, I do not have a scientific proof of my little theory.)

Let us revisit some of our antipatterns, by simply contrasting the 3 ratios [internal, inwards, outwards].

**[60, 30, 10]**

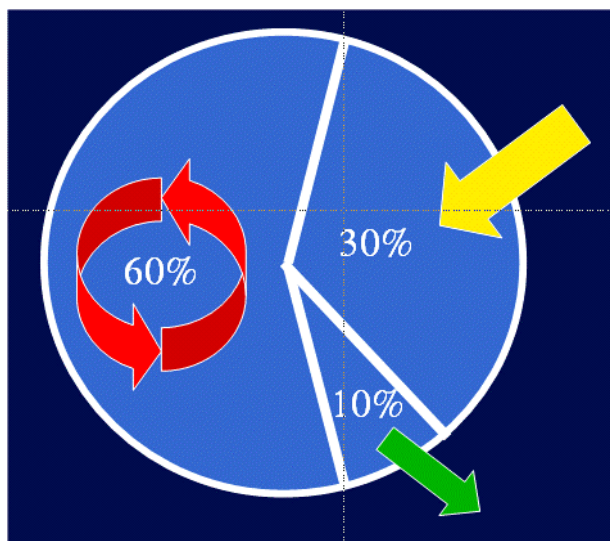


Fig.2

This software architecture team is not engaged enough with its users: the developers, in particular. They are probably doing a good technical job, as they are getting plenty of input, but if they do not regularly provide value to their immediate environment, they will bring something too late, and be ignored. They have to consistently provide value to the team.

**[70, 15, 15]**

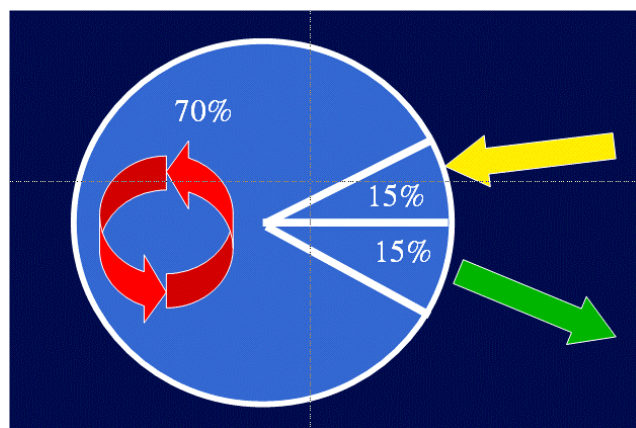


Fig.3

This is a software architecture team that has isolated itself; it has far too much navel focus. They may enjoy themselves, but they are simply not engaged enough with external stakeholders; they are not getting enough input from the users and developers, and are not



providing enough value to their software development organization: advocating the architecture, providing assistance to other organization. Even if they do a good job technically, they will rapidly fall off the radar screen, and will be seen as not bringing value.

**[30, 40, 30]**

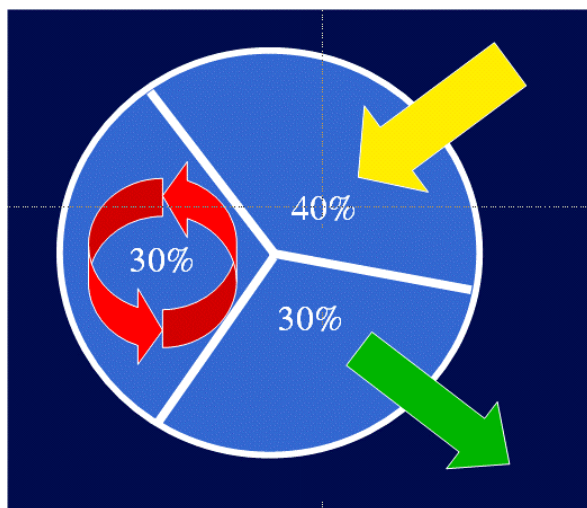


Fig. 4

This is a software architecture team who is spending far too much time traveling the world. Unless this is a very mature system that requires very little architectural work (in which case, maybe the team is overstaffed?), they will run into architectural difficulties.

**[25, 25, 50]**

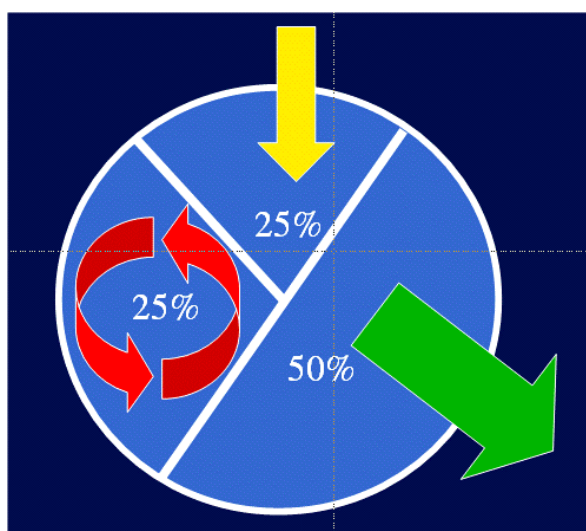


Fig.5

This is a software architecture team that is acting more as an internal consulting shop; or simply they have a too big travel and conference budget. If their focus is helping



internally, maybe this should be made explicit; if their focus is helping outside, is this cost-effective?

This is certainly a case where you may start questioning some of the activities, or the organization, that is, the architecture team composition. Are they doing the job of the product definition team, or should they be simply integrated in one of the development team?

## Variations

Over time the ratios will fluctuate, but not dramatically. Anything approaching one the antipatterns above starts to be suspicious, indicative of some underlying pending issue, some organization imbalance, or a misplaced focus. Yes, the ratio will fluctuate over time, and from individual to individual. There will be more internal focus in the elaboration phase of the first development cycle of a rather novel system. There will be more outward focus during construction and transition, to assist the development teams. While the important point is the overall ratio for a whole team, and various individual will have different time usage patterns, I would also worry if an individual architect would never go outside his office, never see a user, or on the opposite spend all of his time outside.

## Pragmatics

This is not rocket science to implement. If you are in a large company that has a time reporting system, then have ‘them’ create the 3 categories above, or map existing ones into these three bucket (if meaningful). Except absences (holidays etc), have all activities of the architects fall in one of the 3 buckets, with no exception. Accuracy down to the minute is useless—the day or the half-day is often enough. Architects will at the beginning be somewhat puzzled, so define simple guidelines:

- Who were you working with? (Other architects, customers, analysts?)
- Who benefited the most? Us, the architects? Or another party? Were you primarily listening and learning, or were you informing, presenting, preaching, convincing? Were you acting as a consultant to another organization?
- Express “blends” in triplets.

For example:

- Defined API for authentication services: [100,0,0] x 2 days
- Explained the architecture to a potential vendor: [0, 10, 90] x 4 hours
- Had a workshop with the database team: [10, 50, 40] x 8 hours
- Attended a software engineering conference: [0,100, 0] x 3 days
- Planned iteration 4 with PMO: [0, 0, 100] x 2 hours

## Conclusion

Tracking the productive time spent by architects, sorting it in 3 categories: internal (architecture design), external (inwards and outwards communication) and keeping them over time roughly in the ratio [50, 25, 25] keep an architecture team well-focused and balanced in all its expectations.



## **References**

Kruchten, P. (1999). The software architect, and the software architecture team. In P. Donohue (Ed.), *Software architecture* (pp. 565-583). Boston: Kluwer Academic Publishers.

Kruchten, P. (2004). *Training material for the course 'Software architecture and iterative development – Principles and practice'*, Vancouver, BC: Kruchten Engineering Services Ltd. (<http://www.kruchten.com/sitte/courses.html>)