What do software architects really do?

Controversy Corner
By
Philippe Kruchten
(Author of the Original Paper)
Abstract

• To be successful, a software architect—or a software architecture team, collectively—must strike a delicate balance between
  – an external focus—both outwards: Listening to customers, users, watching technology, developing a long-term vision, and
  – inwards: driving the development teams—and an internal, reflective focus: spending time to make the right design choices, validating them, and documenting them.

• Teams that stray too far away from this metastable equilibrium fall into some traps that we describe as antipatterns of software architecture teams.
Architects & the Architecture

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

- **Difficulty with the answer** - It is not a simple, satisfactory answer, since there is no universally accepted definition of what software architecture is.

- **Let's be practical** - So a software development organization, based on its context—domain, culture, assets, staff expertise, etc.—must come up with some delimitation of what constitutes software architecture, and what is beyond software architecture.

- Further, there is “thin line” 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 these decisions may have to be revisited, redefined, and adjusted as an architecture emerges, and as the team expands and the organizational expertise grows.
The paper assumes that architects’ responsibilities are the part of both the design and the design decisions that have long-lasting impact on some of the major quality attributes of a software-intensive systems:

- cost, evolution, performance, decomposability, safety, security, etc, and still able to support the functionality expected by its end user.

Then the role of software architects demands:

- making design choices, validate them, and capture them in various architecture related artifacts.
Architectural Antipatterns

• There are several “antipatterns” that will make a software architect or software architecture team fail miserably if they were only to design the architecture.

• An antipattern—a concept introduced by Koenig (1995)—documents a common mistake made during software development.

• Literature - Architecture antipatterns
  – Brown et al. (1998), Mowbray (2001) - focus on antipatterns in the architectural design.
  – Coplien and Harrison (2005), Ambler et al. (2005) – focus on antipatterns in organization and the process.
Antipattern: creating a perfect architecture, for the wrong system

- A software architect who is not communicating regularly with the customer, the end users etc. is likely to miss the target, particularly as the target is moving, or rather, as the target is only gradually understood.

- Ambler and his colleagues call it “Goldplating.”
Antipattern: creating a perfect architecture, but too hard to implement

- A software architect who does not understand,
  - (maybe limited) the skills, capability and experience of the implementation team(s) that will continue and finish the work,
  - will create enormous levels of stress and frustration, and likely not deliver a quality product in time.

- Paradox: The architectural effort has turned into a computer science research project.

- Ambler et al. calls it “Strive for Perfection”

- Related is Coplien’s pattern “Architect’s Implement”:
  - Solution: Involving the architects in implementing the architecture would mitigate this antipattern.
Antipattern: architects in their ivory tower

- Worse 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 fronts—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.
- References
  - Ambler’s “Ivory Tower” pattern.
    - A special case of this antipattern is the “Architecture watch”, an architecture group that only scouts technologies and provides recommendations to other groups, but is not making design decisions and is not accountable for anything.
  - Ambler has two antipatterns that are similar: “30,000 ft. and Climbing”, and “Real-world Disconnect”.
Antipattern: architects in their ivory tower

- Organizations/Leadership is responsible:
  - Who has been chosen to be the architects?
- It is very likely that you have appointed this role to some of your most talented staff—
  - good at manipulating abstractions, wide experience of a range of systems and technologies, good communication skills, good domain knowledge, etc.—and
  - you may want to use some of these skills for other tasks than just building architectural views.
- Expected Activities other then Architecture
  - You want them to speak to the new prospective customers, to show off the organization’s technical expertise, 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 to a conference to strut your stuff, or to merely extinguish some nasty fire.
  - But if you are not careful, this leads to another antipattern (see next slide).
Antipattern: the absent architects

- No or little architecture design progress is made:
  - the architects are always away doing fascinating things or fighting fires.

- Caution:
  - 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

• Suggestion:
  – The roles and responsibilities of an architect can be usefully captured in a team “charter” / “mission”, that must be adjusted to each organization or project.

• Kruchten has derived a list below from the charter of a large team led by him in mid-1990’s.
  – 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.
Responsibilities ...cont'd

• 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.

• Assessing technical risks.

• Working out risk mitigation strategies/approaches.

• Participating in project planning.

• Proposing order and content of development iterations.
  – For many effort estimation aspects, or for the partition of work across multiple team, managers need the assistance of architects.

• 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.

• Assisting product marketing and future product definitions.
  – The architects have insights into what is feasible and doable and their presence in a product definition or marketing team may be very effective.
Responsibilities ...cont'd

- 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.

- These activities map well onto Bredemeyer’s Architect Competency Framework (2002) and its five main categories:
  - technology, consulting, strategy, organizational politics and leadership.

- 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 solve the most urgent problem, or the most interesting problem, or extinguish the latest fire? Or,
  - Conversely, it may **bring forward the question**:
    - Do we have the right people with right expertise in our current SA team?
Allocation Of Times

• 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, Kruchten proposes a simple time-management practice (see Fig. In next slide).

• It assumes that you are collecting timesheets, to account of where the architects spend their productive time.
  – This is something done by many/most organizations, though often with task categories not quite adapted to what architects really do.

• In general, both globally across the whole architecture team (if there are more than one architect) and on average over the lifecycle
  – Kruchten recommends that the architects should allocate their time in a 50:25:25 (internal:inwards:outwards) ratio.
Activities of architects in time distribution

These numbers come from an experience in managing a 10-person architecture team in 1992–1995.
Allocating times - Kruchten experience

- **Internal focus: 50%**
  - 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 has 2 facets:
    - **Inwards: 25% - 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% - 25% providing information / help to other stakeholders orgs.**
      - Communicating the architecture: project management, product definition.

- This corresponds roughly to architect's roles described by Fowler (2003).

- **Fact:** This apparently crude, off the cuff, partitioning of time has drawn lots of comments, feedback, and push-backs from colleagues and customers, but in the end, except special situations, there have been no substantive evidence to change the numbers [50, 25, 25] over the last 10 years.
  - But as is often the case in software engineering, there is no scientific proof of this little theory.
This software architecture team is not engaged enough with its users, particularly the developers.

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, their input will be too late and be ignored.

They have to consistently provide value to the team.
This is a software architecture team that has isolated itself; it is doing far too much navel gazing.

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
- they are not providing enough value to their software development organization: such as advocating the architecture, providing assistance to other teams.

Risk: Even if they do a good job technically, they will rapidly fall off the radar screen, and will be seen as not bringing value.
This is a software architecture team that is spending far too much time traveling the world.

Remember:
- Unless it 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.
This is a software architecture team that is acting more as an internal consulting shop; or their travel and conference budget is simply too large.

- If their focus is helping internally, maybe this should be made explicit;
- If their focus is helping externally, maybe they should review their cost-effectiveness?

What should be done?

- A case where you may start questioning the architecture team’s composition and also some of its activities.
- Possibility: 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, and differ from individual to individual.

- Lesson to Remember:
  - Anything approaching one of the antipatterns above starts to be suspicious and indicative of some underlying pending issue, organization imbalance, or misplaced focus.

- 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 phases, to assist the development teams.

- Worrysome factor is either and better AVOIDED,
  - 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 – How 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 buckets (if meaningful).

- Except for absences (e.g., holidays, etc.), have all architect activities without exception fall in one of the three buckets:
  - **Internal, Inwards, and Outwards.**
  - Express “blends” in triplets.
    - Blends are activities that are comprised of a combination of internal, inwards and outwards tasks.
  - Reporting accuracy down to the minute is useless—the day or the half-day is often enough.
  - At first, architects will be somewhat puzzled as to which bucket something goes, so define simple guidelines:
Pragmatics – How to Implement? ...cont'd

- **Sample 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?
  - For example:
    - Defined API for authentication services: [100, 0, 0] - 2 days.
    - Explained the architecture to a potential vendor: [0, 10, 90] - 4 hrs.
    - Had a workshop with the database team: [10, 50, 40] - 8 hrs.
    - Attended a software engineering conference: [0, 100, 0] - 2 days.
    - Planned iteration 4 with PMO: [0, 0, 100] - 2 hrs.
Conclusion

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

• Major deviations should attract the attention of the architect or project managers and possibly some examination of the current focus.
Original Paper