



Architectural team, Agile and other stuff

Walter Moscatelli
walter.moscatelli@eoc.ch



What is a software architect?

A software architect makes high-level design choices and frames technical standards. This might include tools, software coding standards, or platforms to be used. To be effective, a software architect needs broad (and deep) technical knowledge to make good decisions. However, technical knowledge isn't enough. They also have to have the soft skills to manage projects and people. Let's review the soft skills and the hard skills needed.

what a software architect may do in a project

1

Interact with clients, product managers, and developers to envision, model, and design the software solution. A software architect advocates clarity and transparency between the client and the team.

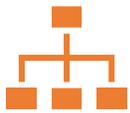
2

Perform regular code reviews to ensure the design quality and avoid overly complicated structures. These tasks usually involve hands-on work on prototype development, code contributions, or technological assessment.

3

Collaborate and mentor. A software architect's skills should enable them to help the development team and enhance their knowledge.

And then



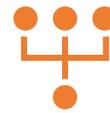
Determining what technical standards and tools are best for a project or the organization as a whole. This will involve self-directed research and evaluation.



Analyzing the goals of a project, breaking it down into discrete areas of functionality that can serve as the basis for applications or microservices, and designing the whole structure using UML.



Helping software teams understand the business requirements and criteria underlying their current project.



Assigning specific development tasks to individual developers or groups.



Performing QA assessments on the project codebase.



Writing code themselves that will make up part of the project.

Software architect: soft skills needed



Leadership - Overseeing the development of a project and coordinating teams of developers to meet design standards requires significant leadership. Software architects must be able to juggle the needs and demands of projects and teams.



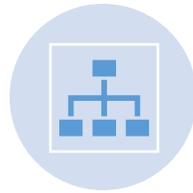
Problem-solving & conflict resolution - Managing and coordinating all of the elements that go into a successful application project requires strong problem-solving skills – both technical and human.



Communication - Communication is a key ingredient in any leadership position. To get the best of teams, software architects must clearly explain the mission, deadlines, and expectations.



Coaching & inspiration - If expectations aren't being met, leaders have to coach and inspire team members to achieve.



Organization - Since software architects set the roadmap for development, being organized is key. Often large-scale and intricate UML diagrams are necessary, which requires a systematic and organized way of thinking.



Prioritizing - Software architects need to quickly prioritize tasks and juggle team members' assignments throughout a product's development.



Detailed thinking - In any development project, there are a significant number of details that must be managed correctly. This requires extreme attention to detail to make sure the project code meets objectives.



Creative thinking - The software architect has to move teams forward to accomplish a build regardless of the obstacles. This takes the ability to think creatively to find alternate solutions or creative ways to solve problems.

what it is not

- A software architect **is not a product manager**. The former decides how the solution works internally, while the latter studies external factors such as market trends, demand for a particular solution, the need for improvements and competitor offers. An architect searches for existing technical possibilities.
- Likewise, he is **not even a project manager**. Although both share responsibility for the success of a solution, however, the former is responsible for code quality and technical risks, while the latter takes care of everything else, i.e. tasks associated with budget, scope and time.

Architectural team in EOC

Help teams build software architecture

Encourage team communication and maintain knowledge

Promote communication between operations and development

Together with business and developers, create the best solutions

Grow developers so they can design robust architectures

Explain and help to use patterns and recognize anti-patterns

Help to use correctly internal / external tools

Design and manage with operationals team CI /CD pipeline

Don't walk in front of me; I
may not follow. Don't walk
behind me; I may not lead.
Just walk beside me and be
my friend.

-albert camus



Error culture

Error culture is the handling of errors and their consequences, within societies, cultures and social systems. The term is often used as a synonym for an environment, where mistakes are allowed and seen as a way to improve. Error culture requires that errors happen and that they are also corrected



1. Thinking All Mistakes Are Equal

- Not all errors are the same. It is safe to say that slip-ups should not happen. We all agree on that. A company that allows such mentality does not work efficiently, wastes resources and will be forced out of the market sooner or later.
- However, some errors are unforeseeable, and for those, you should not punish your employees. Forbidding any kind of mistake to happen, nips every innovative idea in the bud by effectively limiting your employees' motivation to get creative. You want creative and innovative solutions, which makes risks and thereby mistakes inevitable.

2. Preferential Treatment

- Randomness will kill any credibility of maintaining an error culture. All mistakes, and especially mistakes that happen to people who are higher up on the career ladder, must be treated consistently in the same way as for the rest of the company.
- At ready, for instance, a co-founder missed an important deadline for a new product. We convened a meeting with all the employees involved and discussed what went wrong and why. Place emphasis on letting management own up to their mistakes! This creates “role models” for employees in promoting an uncomplicated approach to mistakes and ensures a credible and fair error culture.

3. Sweeping It Under The Rug

- When a mistake happens, talk about it! Make people tackle their mistakes instead of shaming them. Reward employees who dare to be open about it and can show they learned something.
- As a precondition for a healthy error culture, you need the right feedback culture. Formats like “**Bring gipfel**” — where an open discourse about everyone’s weekly mistakes and lowlights — should be encouraged. Creating space for everyone at the company to reflect will reduce the chance of repeating volatile errors.
- All employees will benefit from hearing others’ mistakes, learning from one another and growing closer as a team. Ultimately, a culture of trust is achieved, which is the be-all-end-all of an open feedback culture.



4. A Lack Of Trust

- As with personal relationships, trust in a company requires good communication. For a culture of error to thrive, there needs to be a culture of fairness, a culture of feedback and, finally, a culture of trust. If these are missing, then a company cannot establish a proper error culture.
- Open, honest and transparent communication is essential and can be additionally promoted by those in leadership roles. Narrow-mindedness and being resentful make this enormously difficult. The courage to experiment, innovate and be creative must be supported, and it should be clearly communicated that mistakes are allowed to happen in the process. Flexibility and the willingness to accept change are needed from both employees and management.

Don't be a developer bully

- We should always **put ourselves in another person's shoes** and attempt to remain professional and positive in our feedback to others.
- Be **professional**, don't attack the person
- **Words are Windows, or they're walls**

Gracing the very first page of the book 'Nonviolent Communication' by Marshall Rosenberg



Important
things to
consider

As a team we are noticing some work and team cohesiveness patterns forming around Pull Requests and Peer Reviews. We need to focus on the positive aspects and the goals of our team. **Our goals are to provide working product for our stakeholders and clients, not to put each other down for some trivial technical shortcoming.** If there is a shortcoming and you can provide a directional change as the reviewer, do it, just show the other developer the right way. If you are receiving this constructive criticism, just make the changes and move on.

We are Agile

.....

So we don't need Software architects

Extreme programming



All the contributors to an XP project sit together, members of one team.



There may be a manager, providing resources, handling external communication, coordinating activities. None of these roles is necessarily the exclusive property of just one individual: Everyone on an XP team contributes in any way that they can. The best teams have no specialists, only general contributors with special skills.



The best architectures, requirements, and designs emerge from self-organizing teams. (**Agile Manifesto**)

Responding to Change



We are all empowered to propose better ways of developing software.



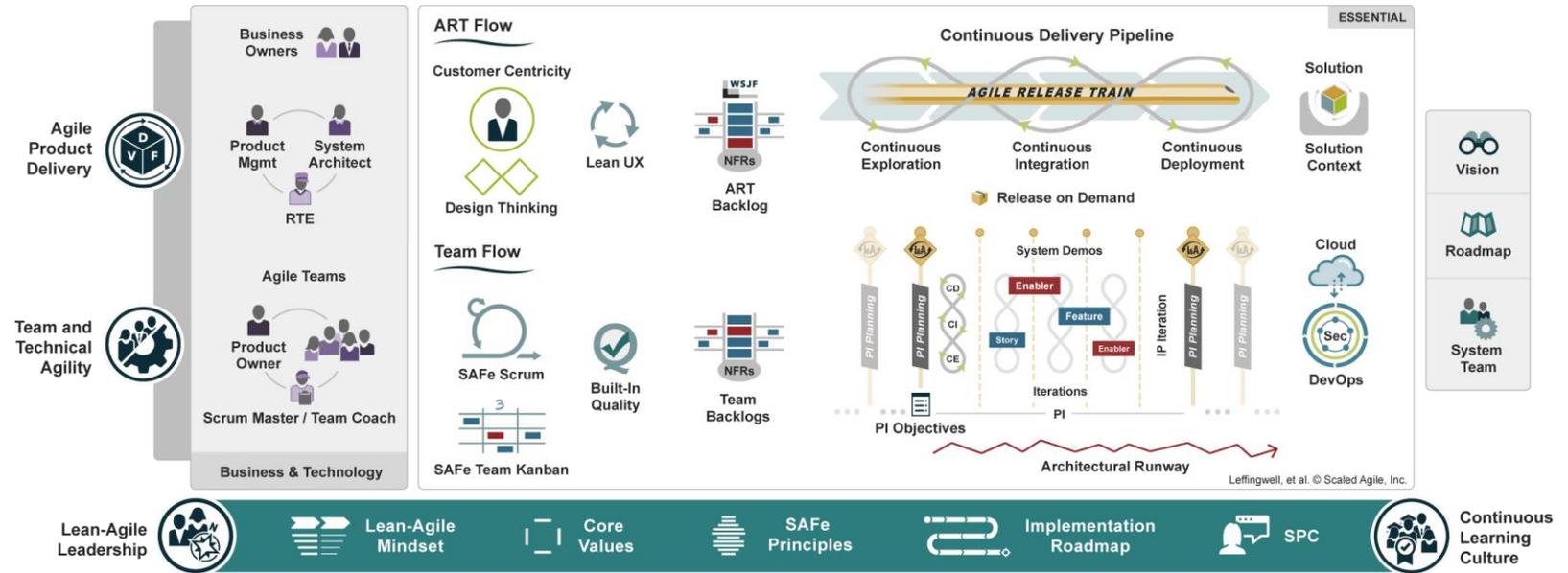
We follow change, we don't have a strategy

We don't need architect! Are you sure ?

SaFe -> Architectural Runway

Intentional Architecture Supports System Evolution

Emergent design



My vision



ARCHITECT AS FACILITATOR
TO SHARE **SOFTWARE**
ARCHITECTURE CULTURE



AT TEAM LEVEL AND
BETWEEN TEAMS



FOCUS ON BUSINESS-
OPTIMISED SOFTWARE
ARCHITECTURE



CREATE SOFTWARE
ARCHITECTURE
COMMUNITY / TEAM



TAKE TIME TO THINK
ABOUT GOOD
ARCHITECTURE



ENCOURAGE
COLLABORATIVE AND
POSITIVE **MINDSET**

Coffee Break

Mindset

HANDWRITTEN TYPEFACE





Architecture and Agile

Architectural team must :

- Help developers and their team take an active part in the architectural choices
- Create and update Architecture guidelines (api,security,network,etc.)
- Help the choice and growth of continuous integration and continuous delivery
- Manage Observability (it works on my computer)
- Force the team to wait to make architectural choices until it is mandatory(db,cache,micro perimeter)
- Document the choices made.

Big responsibility

Conway's law

[O]rganizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations.

— Melvin E. Conway, How Do Committees Invent?

“If you have four groups working on a compiler, you'll get a 4-pass compiler” Eric S. Raymond

Every day in EOC

“Every morning in EOC, a developer wakes up, he knows he has to develop more than the fastest Product Owner otherwise he will be fired. Every morning in EOC, a Product Owner wakes up. He knows he must writes stories faster than the slowest developer, or it will cry. It doesn't matter whether you're the Product Owner or a Developer when the sun comes up, you'd better be working to your maximum.”

The background consists of a dense, overlapping field of small, rectangular sticky notes in various colors: teal, light green, pink, and yellow. Each sticky note features a large, black, hand-drawn question mark. The notes are scattered across the entire frame, creating a textured, busy appearance.

Thank you

Questions ?

References

- **Manifesto for Agile Software Development**
<https://agilemanifesto.org/>
- **How Do Committees Invent?**
http://www.melconway.com/Home/Conways_Law.html
- **SAFe Scrum**
<https://scaledagileframework.com/safe-scrum/>
- **Christopher McDougall**
<https://www.goodreads.com/quotes/292417-every-morning-in-africa-a-gazelle-wakes-up-it-knows>