

PORTUGAL

FEB 10 & 11  
2011



# Emerging Architecture: an agile approach for hyper-productive architects

Fernando Nunes  
Software Engineer, RUPEAL



BACKLOG  
development and consulting

Organized by:

Sponsored by

**Microsoft**<sup>®</sup>



# Agenda

- Non-Scrum Architecture
- Architect as a Scrum Team member, how to work in an agile way
- Build an Architecture, Intentional yet Emergent that “embraces change”
- Best Practices to achieve agile architecture

# What is Architecture?

- “In most successful software projects, the expert developers working on that project have a shared understanding of the system design. **This shared understanding is called *architecture***. This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers”.

Martin Fowler, “Who needs and architect?”, IEEE, 2003



# What is Architecture?

- “A formal description of a system, or a detailed plan of the system at component level to guide its implementation”.
- “The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time”.

TOGAF – The Open Group Architecture Framework

# Easy to change makes the entire system very complex

- “Software is not limited by physics, like buildings are. It is limited by imagination, by design, by organization. In short, it is limited by properties of people, not by properties of the world. *We have met the enemy, and he is us*”.

Martin Fowler, “Who needs and architect?”, IEEE, 2003

# Non-Scrum Architecture

- Big design up front
- Many times it became an Ivory Tower
- Nobody moves forward until a consensus has been reached
- One data truth
- Sounds great in theory, but leads to poor decisions and solutions far away from real needs

# Who is the Architect

- Technical lead
- Has valuable experiences to share
- Mentor, guide and counselor for the team
- Has business vision
- Responsible for technical risks
- Supports communication between stakeholders
- Helps define the structure of the project

# Non-coding Architect

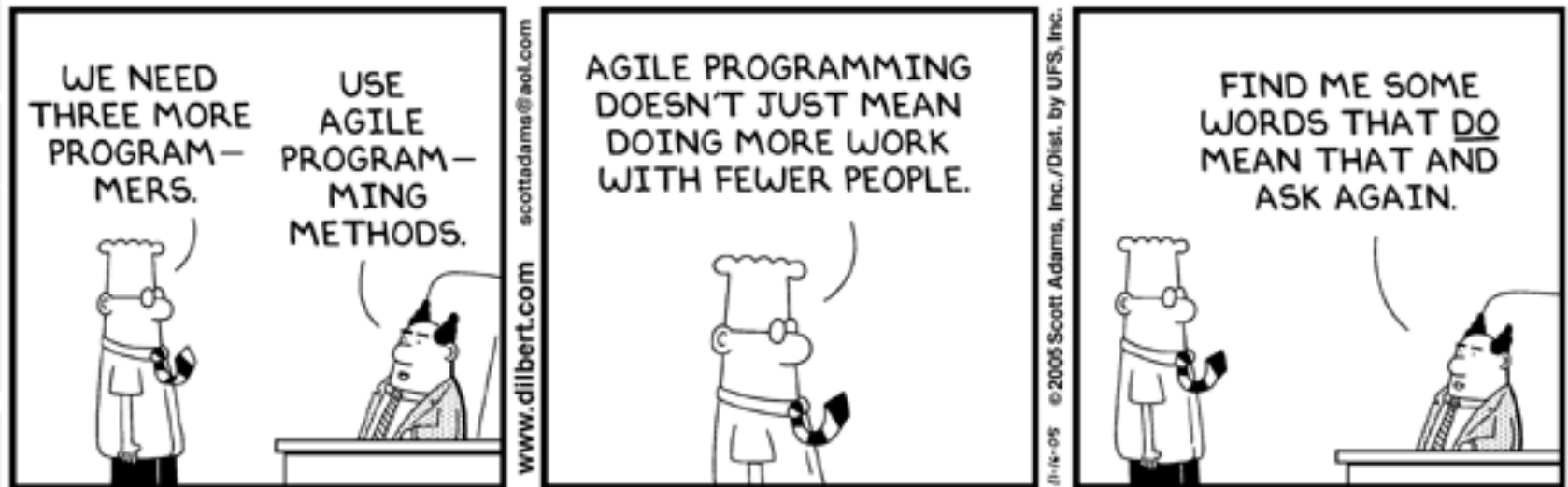
- A few are called “Ivory tower architects”.
- Not welcome to Scrum projects.
- They may know how hard is to implement, but they don't feel it everyday.
- Some of them look on Scrum as a change to again do some the programming.



# Why projects fail?

- Why fail on budget, time, functional requirement or customer expectations?
  - System integration conflicts.
  - Technology coupling.
  - Badly designed interactions.
- Anticipate all the potential issues of integration is also unrealistic, costly and time-consuming, that's why is important an emergent approach.

# Agile Programming



© Scott Adams, Inc./Dist. by UFS, Inc.

# Architect in agile projects

- The architect is **not** replaced or removed.
- Must adjust to fit in with the project structure and approach.
- Becomes an advisor and a facilitator, and not leaves the power to dictate a big and fancy technical solution.

# What is an Agile Architect?

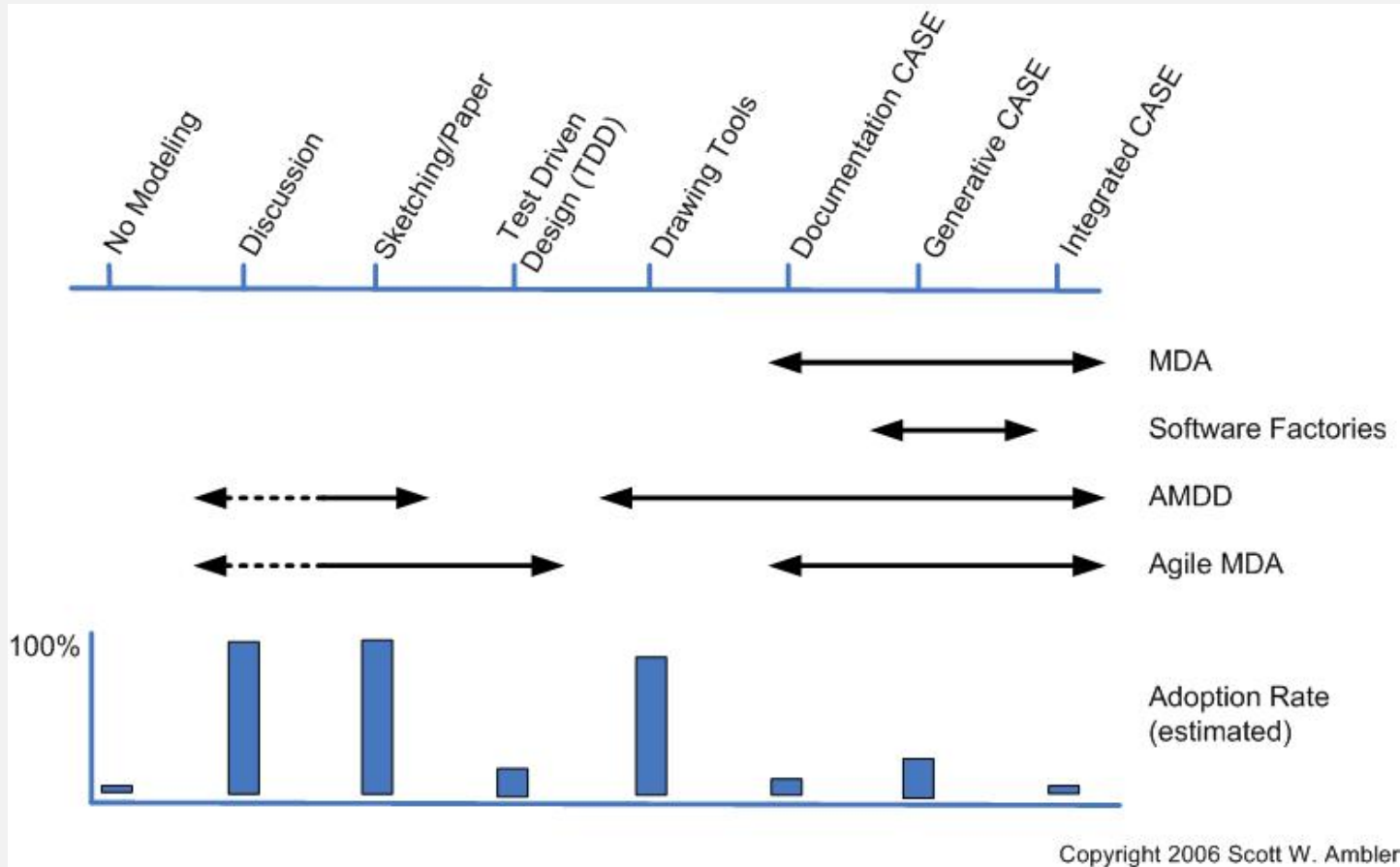
- Be humble. Remember that you don't know everything.
- Beware of building Ivory Towers for “minion” developers.
- Don't document just because... you should.
- Roll up their sleeves and get deep into the code.
- Educates the Product Owner about architectural implications

# Agile Modeling

- For agile teams, Architecture is a way to organize and communicate about work and not to define the work.
- All agile teams use models. They just don't often do a lot of modeling up front.
- Agile teams will create models on the fly as needed to describe, design and define a candidate design for a user story or task.
- In today's globally dispersed environments, whiteboards won't always suffice. We need to share in a way that can cross geographies and time zones.



# Tools for modeling



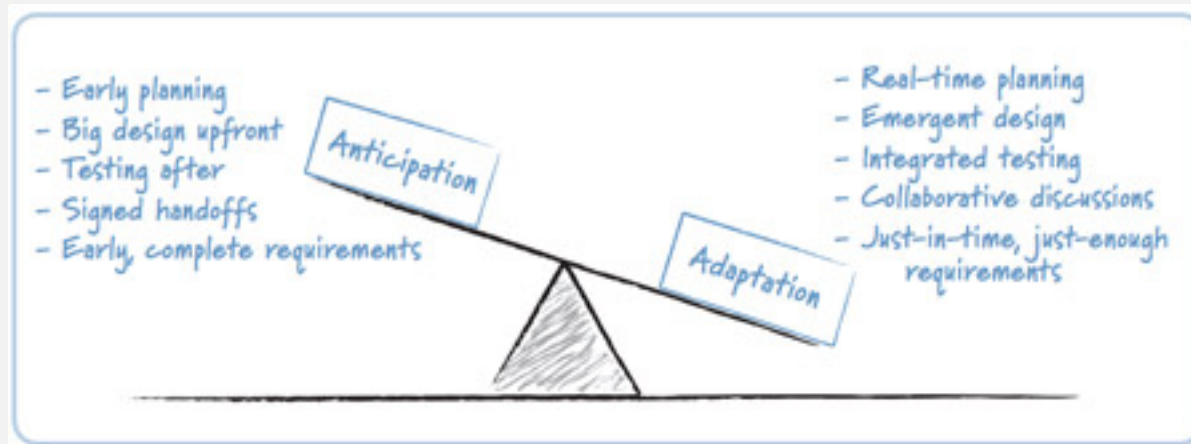
# ...document only what is necessary

- The documentation should be enough to allow a support team to take over without a learning curve.



# Architecture Intentional yet Emergent

- No (big) up-front analysis or design phase.
- The design is done incrementally.
- Achieve the balance between anticipation and adaptation



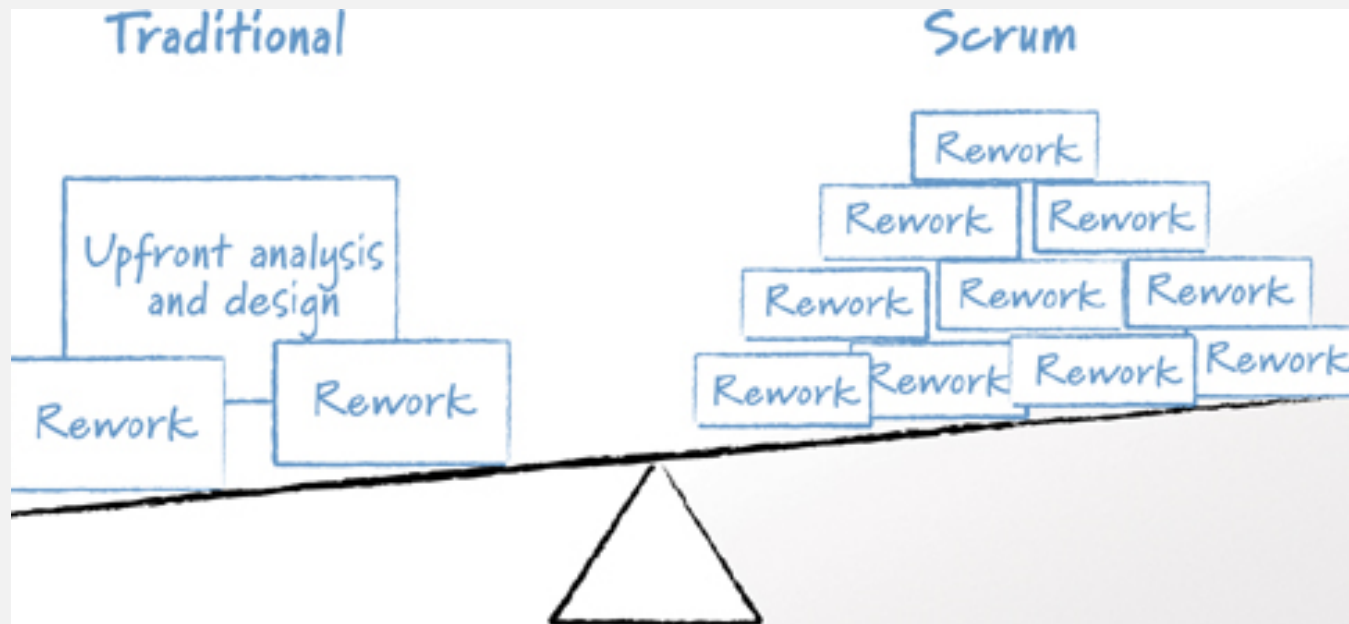
# Life without a Big Design

- Estimating, planning and committing its already hard, but it becomes a bit more.
- It's harder to partition the work.
- Uncomfortable not to have design done.
- Rework will be inevitable.

# Traditional versus Rework and rework

- Traditional has an up-front design and few reworks.
- A Scrum team pursues technical excellence, always keeping the code well factored, with as simple a design as possible, and with a suite of automated tests for early detection of regression problems

# Traditional versus Scrum



# Guiding the Design

- The team should encourage the Product Owner to select items that will maximize learning and drive out technical uncertainty or risk.
- Emerges because there is no up-front design phase.
- Is intentional because product backlog items are deliberately chosen to push the design in different directions and times.

# Prioritize not only for ROI

- Facilitate a discussion between the team and the product owner on how much influence technical factors should have on how the product owner prioritizes de product backlog.



# Eliminate Uncertainties

- Identify the top five technical uncertainties on the project. Move slightly up in priority the product backlog items that could create the **learning** necessary to **eliminate uncertainties**



# Try before you buy

- Consult other teams to get them to pitch “their vision” of a given solution.
- Use a “*tiny implementation of the system that performs a small end-to-end*” (Alistair Cockburn).
- Once that is in place, start adding more end-to-end functionality incrementally.



# Think About Performance Early



- Must the times the performance issues, fault tolerance, uptime and maintenance are stepped over by the product owners.
- Introduce early this issue on the project's lifecycle.
- Let the tests announce that a change led to a degradation in performance.

# Think About Performance Early

- Focus on the most recent changes when performance issues arrived.
- Understand the theoretical achievable performance.
- My favorite question is “How many requests per second, the system must respond?”.

# Listen to Developers



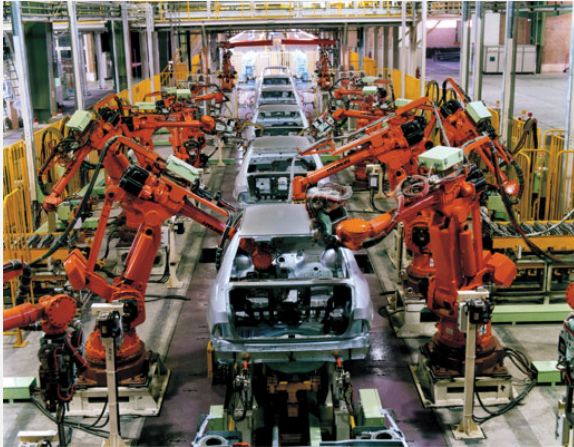
- If they say there's a problem with the design, there probably is.
- Focus on the team. Architecture is not about your ego nor your CV.
- Promote that tools, operating systems are for the most part left to the decision of the team.
- It is far more effective to motivate people to do the right thing than it is to force them to do so.

# Eat your own dog food

- Do not sit in an ivory tower dictating the way forward.
- You'll get a new perspective by being forced to implement your own design.
- Lead by example and be able to fulfill any role on the team.
- Avoid suggesting the latest technology if you have no experience with it.
- Specification's alone have no inherent value.
- Remember the goal is to deploy **working software**.



# Use Continuous Integration



- Ensure the build environment is easy to adopt and use.
- *“It works on my machine”* should be a thing of the past.
- Make your builds available for feedback ASAP.
- Augment continuous integration with **unit/functional testing**.



# Recognize your decision-making

- Make sure you've communicated your decisions and the reasons for it to all those affected directly and indirectly.
- Ensure you have team buy-in, remember that they will actually only enforce decisions they believe in.
- Delegate decisions to domain experts. The best people to model requirements are stakeholders because they're the ones who are the domain experts, not you.
- Good decisions come from experience and experience comes from bad decisions.



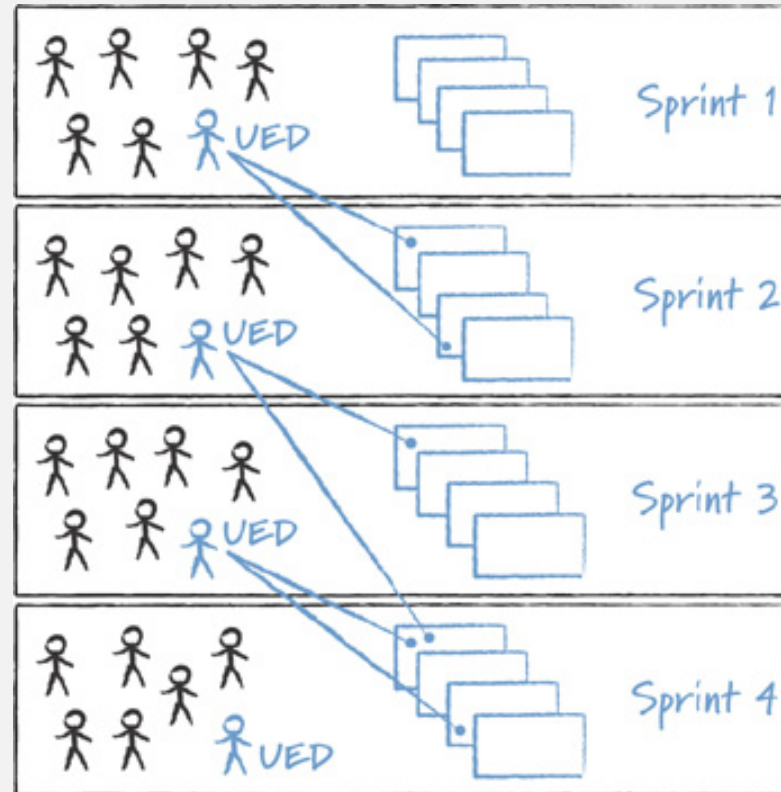
# Realize the interface is the system

- Too many good products are hidden behind poor user interfaces.
- Users don't care about your chosen technology platform.
- User-interface experts should be involved at an early stage.
- UX should be an integral part of the decision-making process.



Boeing 747 Cockpit

# Overlapping User Experience Design



# You can't have it all



- It is virtually impossible to design an architecture that meet expectation all the time.
- You need to be able to convey actual reality.
- Trying to fulfill each and every requirement creates an unstable architecture that excels at nothing.

# Things that work!

- Try before you buy
- Prioritize not only for ROI
- Eliminate Uncertainties
- Use continuous integration
- Listen to developers
- Eat your own dog food
- Realize the interface is the system
- You can't have it all



“Service Manager Beta 2 Has Shipped!!!”

# Who is Responsible for Architecture

- Everyone on the team is responsible for architecture.
- Increases everyone's understanding and acceptance of the architecture because they worked on it together as a team.
- Increases the chance that developers are willing to change aspects of the architecture.

# Conclusions

- Every system has an architecture, the trick is figuring out what you have and what you want to do.
- Educate the Product Owner to prioritize to reduce the risk.
- Trust and listen to developer.
- Architectural modeling and discovery tooling is useful regardless of development methodology.
- Embrace change, but don't overbuild.
- Everyone on the team is responsible for architecture.

# Conclusions - Key Objectives

- Deliver working solutions
- Maximise stakeholder value
- Find solutions which meet the goals of all stakeholders
- Enable the next effort
- Manage change and complexity

# Conclusions - Achieving Objectives

- Value People
  - Everyone has their own area of expertise.
- Communicate!
  - Best way of conveying information is by a face-to-face conversation.
- Less is More
  - Everything you create, document, model, code, test or any other artefact has a cost.
  - Minimise complexity to deliver stakeholder value.

# Conclusions - Achieving Objectives

- Embrace Change: **Plan It, Manage It!**
  - if your architecture cannot handle change then is fragile, not agile.
- Choose the Right Solution for the Enterprise
  - The best solution for the enterprise might not be exactly what a particular stakeholder wants
- Deliver Quality
  - Focus on quality work, but measure quality by the strength and usefulness of the deliverables
- Model and Document in an Agile Fashion
  - Apply the right artefacts and model in small increments

# References

- Scott Ambler: Agile Modeling, Agile Documentation.
- Andrew Johnston: Agile Architect Documentation
- Kevin Francis: TechEd 2009, NZ.
- James Cooper: Architecture in an Agile World.
- Succeeding with Agile, Mike Cohn

# Acknowledgments

Audience, thank you all.

António Cruz,  
Software Architect, PT/SAPO

# Please, give feedback, thank you



## feedback form – session

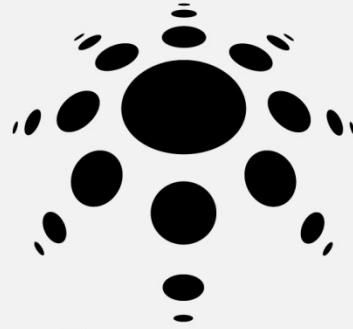


### Thursday 10

<input type="radio"/> Keynote – Mitch Lacey	
<input type="radio"/> Vasco	<input type="radio"/> Ademar
lunch	
<input type="radio"/> Vasco	<input type="radio"/> Jurgen
<input type="radio"/> Keynote - Lasse Koskela	
<input type="radio"/> João	<input type="radio"/> Tiago

### Friday 11

<input type="radio"/> Keynote – Jurgen Appelo	
<input type="radio"/> Geoff	<input checked="" type="radio"/> Fernando
lunch	
<input type="radio"/> Paul	<input type="radio"/> Lasse
<input type="radio"/> Keynote - José Angelo Pinto	
<input type="radio"/> Alexandre	<input type="radio"/> Ana Paula



# Scrum Gathering

**PORTUGAL**

FEB 10 & 11  
2011