Tom Nedwek
Senior Business Architect, Pegasystems

Architects In An Agile World
HOW’S THAT STATUS QUO TREATIN’ YOU?

How Effective Is Enterprise Architecture?

- Fulfilling architecture’s promise: 20%
- Failed initiatives: 30%
- Struggling for success: 10%
- Delivering good journeyman architectural work: 40%

From “Building a Sustainable Architecture Practice”, Scott, 2014
How Effective Is Business Architecture?

- 60%: Struggling for success
- 20%: Failed initiatives
- 15%: Fulfilling architecture's promise
- 5%: Delivering good journeyman architectural work

From “Building a Sustainable Architecture Practice”, Scott, 2014
WHAT’S HAMPERING ALL THESE ARCHITECTS?

“I Am The Architect!”
The Architect’s AntiManifesto

WE ALREADY KNOW THE BEST WAY TO DEVELOP SOFTWARE. WE DON’T NEED TO DO IT BECAUSE WE’RE HERE TO TELL EVERYONE ELSE HOW TO DO IT.

BECAUSE WE’RE SO SMART, WE VALUE

• TECHNOLOGY OVER THE BUSINESS
• EFFICIENCY OVER EFFECTIVENESS
• CONTROL OVER COLLABORATION
• CREATING MODELS OVER DELIVERING VALUE
• BEING RIGHT OVER BEING SUCCESSFUL

Principles Behind the Architect’s AntiManifesto

Change is ok, if it’s our way
Culture and context are irrelevant
Technology is the center of our universe
We want everything to be easy for us
But Seriously ... We Have Our Reasons
The architect is the person who makes all the important decisions because

A single mind is needed to ensure a system’s conceptual integrity
The team members are not sufficiently skilled to make these decisions
These decisions need to be made early on so that everyone has a plan to follow

“Here – Follow This!”

Governance
Principles

Current State Viewpoint
Reference Architecture Framework
Target State Vision

Solution Architecture
Don’t Forget: Your Future-State Diagram

Is just a hypothesis
Needs to be tested
Needs to evolve
Will not survive contact with a dev team

“They Didn’t Do What I Told Them To Do!”

This is not their problem. It’s yours.

Before blaming a dev team, ask yourself

• Are you a part of that dev team?
• Are you slowing down their efforts?
• How often are you communicating with them?
• Are you focusing on results? Or standards?
WHAT’S ARCHITECTURE GOT TO DO WITH IT?

Architecture Is Not IEEE’s Definition

*Architecture is the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution*

ISO/IEC 42010

- “Fundamental concepts or properties”?
- Fundamental to whom?
Architecture is Not RUP’s Definition

Architecture is the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces.

Agility and Discipline Made Easy: Practices from OpenUP and RUP

❓ “Highest level concept of a system”?

❓ Highest level to whom?

Architecture Is Not Upfront Design

Architecture is the set of design decisions that need to be made early on

⚠️ These are actually the things that you wish you could get right early on

⚠️ But early on, you don’t have the information to do so
So, What Is Architecture?

Shared design understanding + Decisions about things that are hard to change = The important stuff. Whatever that is.

Architecture: Shared Design Understanding

- Architecture is the understanding of a system’s design shared by its expert practitioners
- For software architecture, this includes only those components and interfaces that are understood by all expert developers
- Architecture, therefore, is a social construct
  - Doesn’t depend solely on the software
  - Also depends on what group consensus deems important
Architecture: What’s Hard to Change

✓ Architecture is a set of decisions made about aspects of a system (and its development) that are hard (or perceived to be hard) to change

✓ “Hard to change” often closely correlates with “what we wish we could get right early on”

Who Are Architects?

Architects build a shared design understanding among a system’s expert developers

Architects find ways to make hard-to-change things easier to change

In short, they worry about the important stuff
SEEN A SUCCESSFUL, AGILE ARCHITECT IN THE MIRROR LATELY?

Three Jobs of an Agile Architect

1. Serve the business
2. Help IT serve the business
3. Educate, teach, & inspire
Job #1: Serve The Business

Architecture is not important because it’s morally right

Architecture is important because it has economic value

Architecture’s Economic Value

Architecture is about internal value

Internal value is invisible to the customer

What is visible: the software’s ability to change
Software’s Ability to Change

Poorly architected software becomes more difficult to change over time

Well-architected software can actually become easier to change over time

Addressing Irreversibility

Shared design understanding + Decisions about things that are hard to change = The important stuff. Whatever that is.
Addressing Irreversibility

- Irreversibility is a core driver of complexity
- Finding ways to make those things easier to change is actually getting rid of architecture
- Anything can be made easy to change, but ...
  - Making anything easy to change makes the overall system more complex
  - Making everything easy to change makes the system very complex

Job #2: Help IT Serve The Business

If your architects are more worried about making the machines happy than about providing business value, you are talking to the wrong people.

Mike Gehard
Job #2: Help IT Serve The Business

Architects may not have influence over how org charts are drawn.

They can, however, have significant influence over how work is done within that structure.

Conway’s Law

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.
Centralized architecture groups tend to produce architectures based on service busses, monolithic server-side applications, etc.

Decentralized architectural efforts and responsibility tend to produce decentralized architectures based on patterns like microservices.

Nedwek’s Corollary

Any enterprise will favor an organizational design whose structure supports the enterprise's communication structure.
**Functional Groups**

A command-and-control enterprise will

- Pool people into functional departments
- Assign them to projects (for the life of the project)
- Have them report to a functional manager

This functional grouping discourages optimal architectural involvement

---

**OK, It Actually Looks Like This:**

Extra credit: Correctly identify the architect and developers in this picture
Alternative to Functional Groups: Spotify

A matrixed organization, but with a different type of matrix

Traditional matrix is weighted toward hierarchy
- Functional departments
- Assignment to projects
- Reporting to functional manager

This one’s weighted toward delivery
- Vertical dimension: the “what”
- Horizontal dimension: the “how”

Squad

Has a long-term mission
Focuses on one part of the product
Similar to a Scrum team
Contains all the skills required to design, develop, test, and release a solution to production
Has a product owner
**Tribe**

Collection of squads that work in related areas
Typically co-located
Typically smaller than 100 people
Hold regular gatherings
  • What they are working on
  • What they’ve delivered
  • What others can learn

**Chapter**

Small family of people
  • Having similar skills
  • Working in the same general competency area
Local to a tribe
Meets regularly to discuss their
  • Area of expertise
  • Specific challenges
**Guild**

Organic and wide-reaching “community of interest”

Group of people who want to share
- Knowledge
- Tools / Code
- Practices

Cuts across the whole org

---

**Architecture in Chapter-and-Guild**

Disciplines still have a way to organize

These are self-selecting – and often self-managing – Communities of Interest

This is **not** the traditional COE model
Architects At Spotify

Each system has a system owner (or a pair of system owners)
  • Go-to person for technical or architectural issues related to that system
  • Coordinator and guide
  • Typically a squad member or chapter lead

There is also a chief architect role
  • Coordinates work on architectural issues that cut across systems
  • Reviews new systems for alignment to architectural vision
  • Understands that feedback is only a suggestion

-Squads own the designs of the systems they build

Job #3: Educate, Teach, and Inspire
Cultivating A Shared Design Understanding

Shared design understanding + Decisions about things that are hard to change = The important stuff. Whatever that is.

How: Modes of Architectural Leadership

Observer

Scout

Coach

Team Member
Architect as Observer

*Laissez faire* involvement
Want to empower teams to self-organize
• Hands-off ... let the team decide

Empowering people means more than simply taking your hands off the wheel
• Teams need more than freedom
• They need skills and context
• More on that later ...

Architect as Scout

Architect is nominated to find an initial approach
• Goal: avoid becoming a dictator
• Just scout ahead and report back
• The teams own the design

Problems
• Architect becomes bottleneck
• Team members see problems with architecture but didn’t question it
• “I thought you knew something I didn’t”

This is a leadership problem, not a technical problem
Architect as Coach

People are most successful when they have both:

• Technical competence: how
  - Do they know how to do the job?
  - Do they have the skills to make decisions?

• Organizational clarity: why
  - What assumptions are built into the architecture?
  - What are the motivations behind the architectural approach?
  - What is the intent of the architecture?

Having both of those allows us to

• Stop moving information to those with the authority
• Start moving authority to those with the information

Providing Organizational Clarity

Partner to arrive at solutions

Focus on intent

Develop a framework for conflict resolution

• Take people & personalities out of the equation
• Concentrate on solving the problem
• Objectively discuss pros/cons/tradeoffs

Embrace disagreement
Architect as Team Member

The **team** should
- Have the skills to produce the right solution
- Understand the architecture
  - Design
  - Intent, reasons, and assumptions behind it
- Have the ability and authority to make design decisions
- Be able to resolve conflict and navigate engineering tradeoffs
- Feel ownership, be motivated and engaged

The **architect** should
- Lead by helping the team reach that state

How? Become A Servant Leader

**Listen**
- To the business
- To solution delivery teams
- To operations teams

**Share power**
- Collaborate, don’t control
- Suggest, don’t stipulate
- Coach, don’t command

**Help people grow**
- Guide architectural thought
- Unleash energy & intelligence of others
- Seek out candidate architects
Summary

- The architectural status quo doesn’t work
- We architects need to change
- Architecture is the important stuff ... both
- Cultivating a shared design understanding
- Making decisions about things that are presumed to be hard to change
- An architect has three jobs
  1. Serve the business
  2. Help IT serve the business
  3. Educate, teach, and inspire

More Agile2016 Presentations In This Vein

- Emergent Architecture - Just Enough Just In Time
  - Michael Vincent

- Agile Architect - Turning Followers Into Leaders
  - Chris Edwards

- Growing Your Servant Leadership
  - Johanna Rothman
Finally

To borrow from Alan Trefler …

*Our job as architects is not to reach across the aisle to the development community*

*Our job is to work actively to get rid of the aisle*