Architecture-Centric Engineering
Architecture is a Central Artifact
But Architecture is NOT a Stand-Alone Artifact!

Requirements

addresses

Software Architecture

prescribes, abstracts from

Implementation
Project Management
Organizational Management
Lifecycle Management

Evolution Trigger
Development

Project Management

- Requirements Engineering
- Build & Test
- Component Engineering
- Implementation
- Integration

Interfaces
Architecture and Agility
Agile Manifesto

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more

[ http://agilemanifesto.org/ ]
Agile Development Processes in Practice

- Scrum
  - Scrum, but...
- Kanban
- Xtreme Programming
- Lean Development
- ...

© Fraunhofer IESE
Different „Types“ of Interpreting Agility in Practice

- Space defined by Agile Manifesto
  - Dogmatic Agility
  - Perceived Agility
- Space defined by agile methods (e.g. Scrum)
Positive Observations

- Fast results
- Early customer feedback
- Value oriented development
- Change considered as unavoidable fact in software development
Characteristics of Successful Agile Projects

- Small team of skilled developers
- Developers directly talk to customers
- Architectural decisions are taken based on experience
- Coding starts very early
- Running system is delivered, discussed, and improved
Negative Observations

- Agile as excuse for..
  - Ad hoc organization
  - Development without plan

- Highly dependent on (excellent) people
- Major refactorings reduce development speed
- High maintenance cost in subsequent lifecycle

- Does not scale to large-scale projects without adaptations
- Does often not lead to maintainable systems
- Does not allow changing developers
- Does not lead to uniform solutions
Common Anti-Patterns

- Planning with only one iteration in mind
- Customer value overrated, long term business value neglected
- Code considered as the only documentation
- Every requirement should be completely changeable
- Volatile organizational structures are considered agile
- Wrong productivity assumptions
- Self organizing team = No process

Code considered as the only documentation

“Code is documentation”
- There are other stakeholders than developers
- Separation of concerns hardly possible
- Missing abstraction
- Missing documentation of rationales
- Basis for long-term maintenance and evolution?
Business Value over Customer Value

- Agile companies often only look at customer value
- They should look at business value (for their own company), too

**Business Value =**
Customer Value +
Future Ability to deliver Customer Value

(parallel customers, low effort, high-speed delivery, ...)

- Thus, there needs to be a counter-part to „feature-oriented only“ POs
Planning with only one iteration in mind

Short-term architectural solutions: Planning only with next sprint in mind

Global architectural changes within every sprint

Expensive global refactoring (up to 50% refactoring per sprint)
Architecture Work in Agile Environment

- “As little as possible, as much as needed”

- Planning
  - Use available knowledge
  - Risk-based approach → mitigate risks

- Structuring
  - Enable definition of work packages
  - Guide development

- Actively deciding what is decided anyway
  - “You cannot prevent architecture”
  - “You can only prevent an inappropriate architecture

- Always adjusted to organization and project situation
Architecture Best Practices for Agile Development

- Identify Architecture Significant Stories
- Document Quality Drivers as Scenarios
- Code Review to check Compliance
- Determine Planning Horizon for Architecture
- Include Team Expertise in Architectural Decision
- Lightweight Scenario Discussion
- Continuous Integration / Delivery
- Agile Development
- Prototyping
- Planning of Development based on Architecture
- Technical Counterpart to Product Owner
Exemplary Architecture Planning in Agile Development

- **Backlog**: High level architecture
- **Epic**: Detailed solution architecture
- **Story**: Architectural delta for specific sprint

**Coarse Planning**
- Team / Architect

**Guided Planning**
- Development Team

**Derivation**
Influence Factors for Adequacy of Agile Development

- Complexity of the product (size of the team)
- Complexity of the product’s environment
- Adequacy of Agile Development
- Expected life-span of the system and intensity of maintenance
- Similarity of the product to previous ones
- Developers experience and expertise
Things to Remember about Agile Development

- You have to care about the quality attributes of your system!

- You always make architectural decisions…
  - … during architecture design or implementation

- Your architectural decisions get manifested in your implementation

- Don’t rely too much on refactoring
  - It can be very effort-intensive
  - Not all architectural decisions can be refactored
  - It might compromise your architecture

→ Plan upfront, at least to a certain extent
Discussion

- What characterizes a good architect?
1. Software Architect

**Top 100 rank:** 1  
**Sector:** Information Technology

**What they do:** Like architects who design buildings, they create the blueprints for software engineers to follow -- and pitch in with programming too. Plus, architects are often called on to work with customers and product managers, and they serve as a link between a company's tech and business staffs.

**What's to like:** The job is creatively challenging, and engineers with good people skills are liberated from their screens. Salaries are generally higher than for programmers, and a typical day has more variety.  
"Some days I'll focus on product strategy, and other days I'll be coding down in the guts of the system," says David Chaiken, 46, of Yahoo in Sunnyvale, Calif., whose current projects include helping the web giant customize content for its 600 million users. Even though programming jobs are moving overseas, the face-to-face aspect of this position helps cement local demand.

**What's not to like:** You are often outside the management chain of command, making it hard to get things done.

**Requirements:** Bachelor's degree, and either a master's or considerable work experience to demonstrate your ability to design software and work collaboratively.
What Architects Should be...

- Software designer
- Domain expert
- Standards expert
- Technology expert
- Software architect
- Software Engineering economist
- Leader and manager
What Architects do...

- Lead
- Communicate with stakeholders
- Develop project strategy
- Design systems
- Software architect
- Prototype spike solutions
- Evaluate technologies
An Architect’s Skills...

Engineering skills

Organizational skills

Software architect

Interpersonal skills
An Architect’s Skills…

... and, most important, *communication skills!*
Architect as a Mediator and Communicator

- Business Managers
- Software Architect
- Developers

- Business Level
  - Language
  - Value
  - Risks

- Architecture
  - Language

- Technology (-specific) Level

- Credibility
Types of Architects

- Project architect
- Application architect
- System architect
- Enterprise architect
- Software architect
- ... architect

→ Determines scope, role, responsibility, and relationships
→ Depends on the organization, goals, products, ...
→ Terms vary!!!
An Architect’s Goals…

Meet time, budget and quality

→ Happy project owner!

Design adequate solutions for the requirements

→ Happy customers and users!

Design testable, producible, and shippable software (variants)

→ Happy internal stakeholders!

Break-down complexity in manageable, integratable frames open for creative solutions

→ Happy engineers