The Process of Software Architecting

Peter Eeles

Executive IT Architect
IBM UK
peter.eeles@uk.ibm.com
Agenda

Introduction

- Architecture, Architect, Architecting
- Method fundamentals
- Documenting a software architecture
- Reusable architecture assets
- A day in the life
- Summary
Coming Soon!

THE PROCESS OF
SOFTWARE ARCHITECTING

PETER EELES
PETER Cripps
Inspiration

- “If I have seen further it is only by standing on the shoulders of Giants”
- Sir Isaac Newton, letter to Robert Hooke, 15th February 1676

www.booch.com/architecture
www.handbookofsoftwarearchitecture.com
Agenda

- Introduction
- Architecture, Architect, Architecting
  - Method fundamentals
  - Documenting a software architecture
  - Reusable architecture assets
  - A day in the life
- Summary
Core Concepts
Architecture

- Architecture is the fundamental **organization** of a **system** embodied in its **components**, their **relationships** to each other, and to the **environment**, and the **principles** guiding its design and evolution. [IEEE 1471]

- The software architecture of a program or computing system is the **structure** or structures of the system, which comprise software **elements**, the externally visible properties of those elements, and the **relationships** among them. [Bass]

- [Architecture is] the organizational **structure** and associated **behavior** of a system. An architecture can be **recursively decomposed** into **parts** that interact through interfaces, **relationships** that connect parts, and **constraints** for assembling parts. Parts that interact through interfaces include classes, components and subsystems. [UML 1.5]
Architecture versus Design

*All architecture is design but not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change.*

- Grady Booch
An architecture has a particular scope
The benefits of architecting

- Architecting helps manage complexity
- Architecting ensures architectural integrity
- Architecting provides a basis for reuse
- Architecting addresses system qualities
- Architecting drives consensus
- Architecting reduces maintenance costs
- Architecting supports impact analysis
- Architecting supports the planning process
Agenda

- Introduction
- Architecture, Architect, Architecting
- Method fundamentals
  - Documenting a software architecture
  - Reusable architecture assets
  - A day in the life
- Summary
Key Method Concepts

Process

Phase \( \xrightarrow{\text{divided into}} \) Iteration \( \xrightarrow{\text{considers}} \) Activity

Method Content

Task \( \xrightarrow{\text{performed by}} \) Role

uses and produces Work Product

responsible for references

A Waterfall Process
An Iterative Process
Agile

- 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.

- Scrum is a management and control process that cuts through complexity to focus on building software to meet business needs. Scrum is superimposed on top of and wraps existing engineering practices, development methodologies and standards. [Schwaber]
Agenda

- Introduction
- Architecture, Architect, Architecting
- Method fundamentals

Documenting a software architecture

- Reusable architecture assets
- A day in the life
- Summary
Views, Diagrams and Models
## Basic Views and Cross-Cutting Views

<table>
<thead>
<tr>
<th>Stakeholder Group</th>
<th>Requirements View</th>
<th>Functional View</th>
<th>Deployment View</th>
<th>Validation View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>Performance requirements</td>
<td>Component coupling</td>
<td>Component location, Hardware specification, Distribution topology</td>
<td>Performance validation elements</td>
</tr>
<tr>
<td>Group B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Performance View**: Performance requirements
- **Security View**: Security requirements
- **Component coupling**
- **Component location**
- **Hardware specification**
- **Distribution topology**
- **Firewalls**
- **Security policies**
- **User authentication**
- **User authorization**
- **Security validation elements**
Views, Models and Levels of Realization

<table>
<thead>
<tr>
<th>Level</th>
<th>View</th>
<th>Functional View</th>
<th>Deployment View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Architecture</td>
<td>Functional Model</td>
<td>Data Model</td>
<td>Deployment Model</td>
</tr>
<tr>
<td>Physical Architecture</td>
<td>Functional Model</td>
<td>Data Model</td>
<td>Deployment Model</td>
</tr>
</tbody>
</table>
Agenda

- Introduction
- Architecture, Architect, Architecting
- Method fundamentals
- Documenting a software architecture

Reusuable architecture assets
- A day in the life
- Summary
A Metamodel of Architecture Assets
Agenda

- Introduction
- Architecture, Architect, Architecting
- Method fundamentals
- Documenting a software architecture
- Reusable architecture assets

A day in the life

- Summary
Inputs

- Business Entity Model
- Business Process Model
- Business Rules
- Existing IT Environment
- Vision

[Diagram of entity relationships]
Types of Requirements

- Functional requirements
  - *Describe the behaviors (functions or services) of the [IT] system that support user goals, tasks or activities.* [Malan]

- Non-functional requirements
  - *Non-functional requirements include constraints and qualities.* [Malan]
  - **Constraint**
    - *A constraint is a restriction on the degree of freedom we have in providing a solution.* [Leffingwell]
  - **Quality**
    - *[System] qualities are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system.* [Malan]
Define Requirements

1. Collect Stakeholder Requests
2. Capture Common Vocabulary
3. Define System Context
4. Outline Functional Requirements
5. Outline Non-Functional Requirements
6. Prioritize Requirements
7. Detail Functional Requirements
8. Detail Non-Functional Requirements
9. Update Software Architecture Document
10. Review Requirements with Stakeholders
Task: Collect Stakeholder Requests

- Pitfall: Treating Requests as Requirements
- Pitfall: The Shopping Cart Mentality
- Pitfall: The Questions are too Technical
- Pitfall: Requests Are Too General
- Pitfall: Requests Are Not Measurable
- Pitfall: Talking with the Wrong People
- Pitfall: All Requests Are Equal
Task: Define System Context

- Customer
- Sales Clerk
- Tour Organizer
- System Administrator

System:
- Payment Engine
- Reservation System

YourTour
Task: Outline Functional Requirements
Task: Outline Non-Functional Requirements

- Usability Requirements
- Reliability Requirements
- Performance Requirements
- Supportability Requirements
- Constraints
  - Business Constraints
  - Architecture Constraints
  - Development Constraints
  - Physical Constraints

“Brownfield sites are those in which redevelopment or reuse of the site is complicated by existing contaminants. Greenfield sites are clean, previously undeveloped land”. [Hopkins]
## Task: Prioritize Requirements

<table>
<thead>
<tr>
<th></th>
<th>Accessibility</th>
<th>Availability</th>
<th>Communication</th>
<th>Integration</th>
<th>Online Help</th>
<th>Platform Support</th>
<th>Scalability</th>
<th>Schedule</th>
<th>Security</th>
<th>Speed</th>
<th>Standards Compliance</th>
<th>Testability</th>
<th>Third-Party Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Content</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Book Tour</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browse Tours</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Booking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Manage Profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Tour</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage Tour Participant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register Tour Participant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specify Tour</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Any*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
Create Logical Architecture

1. Survey Architecture Assets
2. Define Architecture Overview
   - Outline Functional Elements
     - Detail Functional Elements
   - Outline Deployment Elements
     - Detail Deployment Elements
3. Document Architecture Decisions
   - Verify Architecture
   - Build Architecture Proof-of-Concept
4. Validate Architecture
5. Update Software Architecture Document
6. Review Architecture with Stakeholders
From Requirements to Solution

Requirements

Functional Requirements

Non-Functional Requirements

Solution

Functional Model

Deployment Model
Approaches

- **Attribute Driven Design (ADD) Method**
  - Developed at the Software Engineering Institute
  - Quality attributes drive the derivation of the architecture
  - Underpinned by architectural tactics and patterns

- **Siemens’ 4 Views (S4V) method**
  - Developed at Siemens Corporate Research
  - Starts with a global analysis of the factors that influence the architecture
  - Iteratively addresses challenges across four views (conceptual, execution, module and code architecture)

- **The Rational Unified Process (RUP)**
  - Developed at Rational Software (now IBM Rational)
  - Driven by architecturally-significant requirements
  - Each iteration considers the key architectural elements of the solution, before realizing the requirements using these solution elements
How Much Logical Architecture?

- Minimizing Logical Architecture
  - The logical architecture is simply a means of getting to a physical architecture as quickly as possible
  - In some cases, no logical architecture may be required at all
    - E.g. The requirements for the system are similar to those of an existing system
    - E.g. We are using a packaged application or integrating with an existing system

- Logical Architecture as an Investment
  - A valuable asset if a technology change is anticipated at some point in the future
Task: Define Architecture Overview
Task: Outline Functional Elements

- Component identification
  - Business Entity Model
    - Clustering of related entities
  - Functional requirements
    - Boundary, control and entity components
  - Non-functional requirements
    - Constraints
    - Components that address specific technical challenges (e.g. security)
- Business rules
  - Business rules component(s)
- Architecture decisions
  - Use of particular assets (e.g. packages, patterns)
Task: Outline Functional Elements

- **Tour Booker**
  - Tour Booker Interface

- **Payment Engine**
  - Payment Engine Interface

- **Reservation System**
  - Reservation System Interface

- **Book Tour Controller**
  - get tours()
  - select tour()
  - add tour participants()
  - add tour participant options()
  - book tour()

- **Tour Manager**

- **Tour Booking Manager**
Task: Outline Functional Elements

1: get tours
2: select tour
3: add tour participants
4: book tour

1.1: get tours
2.1: create tour booking
3.1: add tour participants
4.1: book tour

1.1.1: get tours
2.1.1: create tour booking
3.1.1: add tour participants
4.1.1: book tour

1.1.1.1: add tour participant options
4.1.1.1: book tour

1.1.1.1.1: add tour participant options
4.1.1.1.1: book tour

1.1.1.1.1.1: add tour participant options
4.1.1.1.1.1: book tour

1.1.1.1.2: provide payment details
4.1.1.2: process payment

1.1.1.1.2.1: provide payment details
4.1.1.2.1: process payment

1.1.1.1.2.2: make reservations
4.1.1.2.2: process payment

1.1.1.1.2.2.1: make reservations
4.1.1.2.2.1: process payment
## Assigning NFRs to Components

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Component</th>
<th>Operation</th>
<th>Budgeted Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking a tour must take less than 10 seconds, from the time the request is submitted, to the confirmation of the booking being presented to the user</td>
<td>Tour Booking Manager</td>
<td>provide payment details()</td>
<td>1 second</td>
</tr>
<tr>
<td></td>
<td>Payment Engine Interface</td>
<td>process payment()</td>
<td>3 seconds</td>
</tr>
<tr>
<td></td>
<td>Reservation System Interface</td>
<td>make reservations()</td>
<td>6 seconds</td>
</tr>
</tbody>
</table>
Task: Outline Deployment Elements
Task: Outline Deployment Elements
Task: Outline Deployment Elements
Summary

- The process of architecting …
  - Spans software engineering disciplines
  - Applies across the project lifecycle
  - Draws upon proven experience (practices, standards and other assets)
  - Is built upon solid engineering principles
Thank You