Modeling in IBM
A practitioners view
In our model based development approach we have been trying to combine analysis, design and construction (to a very[?] far extend)
My world (most of it) is the basis for analysis, design and construction – *IT architecture*
What is your definition of IT architecture?

What is the difference between architecture and design?

Why is IT architecture important and why do projects need it?

What is the role of the IT Architect?

What diagrams and models do you currently use to represent architectures?
Agenda

Introduction

What is architecture?
Requirements aspect
Functional aspect
Operational aspect
Using patterns
Validation aspect
Architect career in IBM
Closing
My IBM Career

- 97-99: Developer
  - Lufthavne
  - TKE

- 98-…: Method Consultant

- 99-00: "Architect" - e-Commerce
  - Several proposals
  - ToyCity, GiftAhoy

- 98-…: Teacher
  - SI-/ GSMethod
  - OTU2
  - Commerce Suite
  - Component Modeling
  - Architectural Thinking
  - SOA Bootcamp
  - SOMA Workshop

- 01-…: Architect
  - 4PL vision
  - eCommerce at major shipping firm
  - Telco Service Platform
  - VIRK
  - Banking SOA Method Framework
  - Review: IKEA, MSL, PFA, YouSee, many IBM TDA’s, …
  - Proposal and pre-sale: Skat, Domstolsstyrelsen, TDC …
  - Leader of Nordic SOA Center of Excellence
  - Service delivery for Novartis
  - Application strategy @ Ministry of Finance
  - Several large proposals
  - Currently: Large complex system integration project at central government agency

- 03/12: Certified IT Architect
  - Application Architecture
  - Started the formal process in December 2002!
  - Submitted package ultimo Oktober 2003
  - Recertified in 2006 and 2009
What we deliver in my part of IBM (the “services” part) – an where I usually fit in…

Notice the gaps…
Introduction

What is architecture?

Requirements aspect

Functional aspect

Operational aspect

Using patterns

Validation aspect

Architect career in IBM

Closing
"a shift toward integrated solutions and quantifiable business value" mandates for business and IT alignment – **Enterprise Architecture is key** (but that is not my story today)

**What is architecture?**

- Strategy and vision
  - "Which city do we want to build?"

- Enterprise Architecture
  - "the city plan"

- System Architecture
  - functional aspects
  - operational aspects
  - "the infrastructure and single building design"
Definition: The architecture of an IT system is the structure or structures of the system, which comprise software and hardware elements, the externally manifested properties of those elements, and the relationships among them.

(I’m not considering “Enterprise Architecture”)
Architecture versus design

System Architecture is about creating the logical and physical level, technology independent and dependent structures that define the major static and dynamic elements of a solution using guidelines that influence their development. It sets the context for Solution Design.

System Design is about creating the internal behavior of the structural elements plus additional detail towards realizing the System Architecture.
Some popular misconceptions about architecture…

- Architecture is just paper
- Architecture and design are the same things
- Architecture and infrastructure are the same things
- <my favorite technology> is the architecture
- A good architecture is the work of a single architect
- Architecture is simply structure
- Architecture can be represented in a single blueprint
- System architecture precedes software architecture
- Architecture cannot be measured or validated
- Architecture is a science
- Architecture is an art
Definition: The IT Architect defines (architects) solutions to business problems through the reasoned application of information technology. Those solutions are manifested as architectures and can include systems, applications and process components.
What do these definitions mean in practical terms?

An IT Architect should be:
- A practitioner
- A consensus builder
- Results-oriented
- A generalist

An IT Architect should not be:
- A “top-level” software designer
- The project manager
- A technology expert
- A product expert
- A lone scientist
Architectural Thinking involves inputs, process, and outputs

- Analyzing the requirements (and resolving conflicts between them)
- Applying principles (possibly defined at the Enterprise Architecture level)
- Creating models at the right level of abstraction (that is, hiding details that are not relevant)
- Creating views that satisfy the concerns of different stakeholders
- Providing a decision trail
What is architecture?

Architectural Thinking involves analyzing requirements by addressing a number of concerns.

- What freedom do we have?
- What is to be solved?
- How good is it?
- What is available?

Constraints | Functional Requirements | Qualities | Assets & Technology

Architecture

- Procurement
- Design
- Implementation
- Testing
- Operations
Architectural Thinking involves creating views to satisfy the concerns of different stakeholders

Will the system be available?
Will the system let me create orders?
Will the system be maintainable?
Will the system be secure?
Our method ("Unified Method Framework") is the "repository" of best practice and experience representing a number of "basic" and "cross-cutting" viewpoints.
The dimensions of architecture
The System Description Standard (SDS) describes the fundamental concepts that can be used to describe systems.
The System Description Standard (SDS) provides a standard language with the *functional* and *operational* aspects being very important.

### Functional Aspect
- **IT Component**: A modular unit of IT functionality, which makes this functionality available through an interface
- **Interface**: A set of operations offered by a component
- **Collaboration**: Captures the exchange of messages between components in the context of a particular scenario
- **Interaction**: The messages exchanged between one or two components in the context of a collaboration, and the sequencing of these messages via their associated send/receive events

### Operational Aspect
- **IT Node**: A collection of hardware and software components fulfilling a specific responsibility with a certain quality of service within the overall system
- **Connection**: The required connectivity between IT nodes
- **Deployment Unit**: An abstraction of an IT component created to simplify the placement process
- **Location**: A geographical area or position

![Diagram showing Components, Deployment Units, and Nodes](image)
Introduction
What is architecture?

Requirements aspect
Functional aspect
Operational aspect
Using patterns
Validation aspect

Architect career in IBM
Closing
What are requirements? They are fundamental to system architecture

**Functional Requirements:**
- Are capabilities needed by users to fulfill their job
- Answers the question of "what" the customer needs (but often not "how" it is achieved)

**Qualities:**
- Define the expectations and characteristics that the system should support
- Might be runtime (for example, performance or availability) or non-runtime (for example, scalability or maintainability)

**Constraints:**
- Givens, those things that cannot be changed within the scope and lifetime of the project
- Other factors, such as mandated technologies, available skills, and budget

(Qualities and constraints are sometimes referred to as “non-functional requirements”)

**Requirements determine the architecture of a system in terms of:**
- Structuring and placement decisions
- Solution sizing and costing
- Product selection, deployment, and configuration framework

**Requirements are of a direct concern to an IT Architect:**
- Assists with the collection and definition of a system’s architectural qualities and constraints (non-functional requirements)
- Participates in the review and gathering of business and functional requirements
- Assesses feasibility of requirements in the context of the proposed technical solution
- Form the basis of traceability of business requirements to architectural designs, which is validated during the review process
What does the requirements space look like?

- Identifies business functions the system supports
- Primarily related to the system service levels
- Identifies potential enhancement based on both business and technology opportunities
A sample taxonomy for Qualities and Constraints (also known as Non-functional Requirements)

Non-functional Requirements

Qualities
- Runtime
- Non-Runtime

Constraints
- Business
- Technical

Accessibility

See more here: http://en.wikipedia.org/wiki/List_of_system_quality_attributes
Most important artifact (1): The System Context is essential to capturing the scope of the project

The System Context helps to:
- Clarify the environment in which the system has to operate
- Put bounds on the system
- Identify external interfaces (users or systems)
Most important artifact (1): The *Architecture Overview* setting the scene for the solution (to be slept with under the pillow)
Architecture Overview example: Retail Customer Access Points—The Retail Customer can choose from a variety of ways to interact with the company. The supporting infrastructure should be common whenever possible.
Architecture Overview example: Non-technical Audiences
Example – I lead a team of architects in a complex environment with unclear requirements and several stakeholders:

XXXWorld Service Platform – an architecture overview

XXX Integration Framework
Same example – Unclear requirements and several stakeholders and several very large “pre-selected” components with overlapping capabilities and being in development by vendor – another architecture overview
Introduction
What is architecture?
Requirements aspect

**Functional aspect**
Operational aspect
Using patterns
Validation aspect

**Architect career in IBM**
Closing
The functional aspect of IT architecture captures the system’s functional behaviour and is described using *components*, their responsibilities, dependencies, and interactions.

- A component is a modular unit of functionality.
- A component makes its functionality and state available through one or more interfaces.
- Examples of components are:
  - Business processing components (such as the Customer Processing component)
  - Business service components (such as the Account Manager component)
  - Technical components (middleware such as messaging or transaction software)
  - System software components (such as an operating system)
  - Hardware components (such as an encryption device)
- A component is not just a programming level concept such as an EJB or .NET component!
The functional aspect of IT architecture is primarily focused on...

its structure, as described in a Component Relationship Diagram, as well as...

...its behavior, as shown in a Component Interaction Diagram
Most important artifact (2): The Component Model which is established using the Component Modeling technique

- Partition into subsystems and components and assign responsibilities
- Review architectural patterns, reference architectures, and reusable assets
- Structure ensuring loose coupling, high cohesion, and so on

- Specify interfaces
- Specify operations and signatures
- Specify pre- and post-conditions

- Identify products and packages
- Define implementation approach
The Component Model is used as input into a number of activities

- Work Allocation
- Version Control
- Design Strategy
- Reuse
- Testing
- Project Management
- Product/Package Selection
The components arise from the functional requirements

**Use Case: Check Stock Availability**

- The actor requests a stock query.
- The system prompts for the actor to select a store name and enter an article or product name/number.
- The actor selects a store name and enters an article or product name. If a product name is entered, the user may also be required to specify further selection criteria (such as selecting color and so on) for that product.
- The system searches for the selected article or product in the named store and checks its availability.
- The system determines that the article or product is available.
- The system returns the availability information for the article or product.
- The use case ends successfully.
... and from data

1. Produce a logical data model showing business entities.

2. Identify core business entities.

3. Create components to ‘manage’ core business entities.
... and the operational aspect influences the Component Model from multiple sources like

Placement Considerations:
- Adding components that allow existing components to collaborate between nodes, e.g. asynchronous messaging
- Adding responsibilities to handle different presentation types, e.g. thin versus thick client
- Adding components that handle data and software distribution

Achieving Observable Qualities:
- Aggregating components, e.g. to achieve better performance
- Segregating components, e.g. if responsibilities have different runtime qualities

Availability:
- What if individual components must be failure-resilient?
- What if the system is used by subsidiaries in other time zones?

Scalability:
- What components are needed to support planned or unplanned growth?

Performance:
- How can long- versus short-running transactions be handled?
Introduction

What is architecture?

Requirements aspect

Functional aspect

Operational aspect

Using patterns

Validation aspect

Architect career in IBM

Closing
Systems must be deployed - either using new or onto existing infrastructure

The design of operational system aspect may be challenging to develop or understand
- Many competing concerns must be resolved:
  - Qualities: for example, Performance, Availability, Manageability, Security
  - Constraints: for example, Affordability, Standards compliance, Existing Infrastructure
- Multiple activities must be co-ordinated:
  - Detailed design: Network, Hardware, Systems management
  - Commercial: Planning, Pricing, Procurement

The functional aspect is not always sufficient for these "systems engineering" concerns - its prime purpose is "software engineering"
How systems are deployed is the primary concern of the Operational Aspect of IT System Architecture

The Operational Aspect:

- Is part of a dimension of an IT system’s architecture
- Represents how the application’s components are deployed across the geographical structure of the IT System
- Describes how the system’s Service Level Requirements (SLRs) are satisfied
- Serves as a blueprint for detailed infrastructure design, such as network design, platform design, systems management design

The key artifact is the Operational Model, which shows how we implement the IT system’s functional and non-functional requirements within the constraints of technology, skills and budget
Most important artifact (2): Operational Model helps ensure the system's non-functional requirements are delivered, within all constraints by providing the architectural context for detailed design across a myriad of themes.

- System Monitoring
- Systems Management
- Service Level Characteristic Analysis
- Capacity Planning
- Software & Data Distribution
- Availability & Performance Engineering
- Procurement
- Package & Product Selection
- Defining Organizational Responsibilities

How can we make it work?  
(Operational Model)

How do we configure it?  
How do we manage it?  
(Detailed Design)
There are many ways of deploying a single component into a simple system...

Option 1

Everything on the PC...

Option 2

How do we do this, for complex systems, in a structured manner?

Operational modelling!

Option 3

...Or not!

Option 4
To achieve this goal, we need a structured, formal language to describe the elements featured on the Operational Model

To achieve this goal, the Operational Model represents the system’s “infrastructure architecture”, using a variety of model elements, including:

- The geographic structure of the locations and their borders, over which the IT system will be deployed and operated
- The placement of the system’s nodes into these locations
- The deployment of the system’s components across these nodes, using deployment units
- The connections between nodes
- The organisation of the system’s elements into zones

As well as this description of the deployed system, the OM also documents:

- The usage of logical structures to deliver the non functional requirements
- The patterns which are use for logical and physical configuration
- The overall physical configuration of the technologies and products necessary to deliver the functional and non-functional requirements of the IT system
- Sizing and other hardware specifications for all the computers, storage devices and network technologies
We have to understand what it is we need to deploy, via the DEPLOYMENT UNIT Model

DUs give us a mechanism for
- Understanding the non-functional requirements placed on the system’s components
- Deciding where best to place the various aspects of the system’s components
- Tracing the system’s requirements to its design

<table>
<thead>
<tr>
<th>Nature of Access Point</th>
<th>Actor using Access Point</th>
<th>Location</th>
<th>Required Hours</th>
<th>Concurrent Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Txn Type</td>
<td>Usage ‘Window’</td>
<td>Number per user</td>
<td>95% response time</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Release Frequency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>Unit/record Size</td>
<td>Number records</td>
<td>Volatility</td>
<td>Currency</td>
</tr>
</tbody>
</table>
The Operational Model is developed across a “backdrop” of the system’s geographic landscape...
...based on an understanding of the locations and their relationships (borders) over which the DUs, nodes and connections will be placed.
... and the applications’ data DUs can now be placed, based on actor access requirements (“data sharing” and other factors).
Working through this rationale produces the final Application level OM.
Now we can “debate” non-functional requirements and other considerations, and hereby identify any additional node ...
At some point in time products may be selected

L1, Corporate Customer
L2, Private Customer
A_Browser
Internet

L3, Internet Services
(1 to n)

L4, Central Site Runtime Services (1)

L5, Corporate Services

L6, Other Internet Services

L7, Store

L8, Country Office

Given and obvious product choices

WAN

Select firewall for controlling access in the DMZ

Select software to host customer data (adjacent choice)

Operational aspect
Another example – huge (innovative) operational model designed for extreme flexibility and optimized resource usage.
Same example – an underlying concept of environments delivered as a composition of "segments" within “sectors” (depending on SLAs). All based on 4 very large virtualized p590 32way and a truck-load of Wintel servers.
Introduction
What is architecture?
Requirements aspect
Functional aspect
Operational aspect
Using patterns
Validation aspect
Architect career in IBM
Closing
What is a reusable asset?

Definition: Reusable assets provide a solution to a problem for a given context.*

- Assets may have variability points, which are defined as a section that can/should be customized by the user of the asset.
- Assets include rules and instructions for use or customization.
- Assets contain artifacts that are work products from the software development lifecycle.

* From OMG’s Reusable Asset Specification document
How are assets classified?

* From OMG's Reusable Asset Specification document
Examples of how assets are classified

- Reference Architectures
- Architectural Patterns / Styles
- Design Patterns
- Legacy Systems
- Frameworks, Packaged Applications
- Components, Component Libraries

Granularity
Variability
Articulation
A set of banking specific business models that represent good practice in banking

The IFW business models enable an enterprise to:
- Be more adaptive and to respond quickly to changing customer needs
- Focus on achieving competitive differentiation
- Identify and leverage best practice behaviors across the organization
Anti-pattern: Golden Hammer

- Anti-Pattern Name: Golden Hammer
- General Form: A software development team is very skilled in a particular solution (Golden Hammer) and every development effort is viewed as something that is best solved with it.
- Symptoms and Consequences:
  - Misapplication of a favored concept or tool; identical solutions for varying types of requirements and scope
  - Solutions have inferior performance, scalability, etc., compared to others
  - Requirements are not fully met in an attempt to leverage existing investments
  - Existing solution dictates architecture and design
  - Heavily dependent on a particular set of technologies
- Refactored Solution:
  - Expand the knowledge of developers through education, training, and book study groups to expose developers to new solutions
Introduction
What is architecture?
Requirements aspect
Functional aspect
Operational aspect
Using patterns
Validation aspect
Architect career in IBM
The architectural process

- Definition of the architectural process and how it fits in the IBM Unified Method Framework (UMF)
- The iterative approach of the UMF and how architectural work is structured to produce architectural work products
- Pitfalls to avoid
- Dangers inherent to the architectural process
- An organizational approach to the architectural process
- Assessments of the architecture
What is a Viability Assessment?

Most dictionaries define *viability* as the ability for something to grow or develop

When examining viability in terms of a development project or architecture, consider:

- **Is it suitable?**
  - Is it an appropriate solution in terms of customer requirements?
- **Is it doable?**
  - Is it in the "realm of the possible"?
- **Is it practical?**
  - Can it be done with the available means?

A Viability Assessment is used by the architect to:

- Verify that the solution meets customer requirements
- Ensure that the proposed solution is possible
- Assess client acceptance of the technical elements of the design
- Confirm development approach and environment
- Identify project risks
Rules of Thumb - Defect Removal, Testing and Function Points

RofT 23: The number of potential defects in a new development is given by raising the FP count to the power of 1.25. [14]

\[
\text{Potential Defects (New Developments)} = \text{FP Count} ^ {1.25}
\]

Typical US removal rates are around 85% efficiency, i.e. Delivered Defects = Potential defects * 0.15.

RofT 24: For a CMM level 3 organisation, the number of potential defects in a new development is given by raising the FP count to the power of 1.1 and the defect removal efficiency is about 95%. [14]

RofT 26: Each formal design inspection will remove 65% of the bugs present. [14]

RofT 27: Each formal code inspection will remove 60% of bugs present. [14]

RofT 28: Each software test will remove 30% of the bugs that are present. [14]
The testing ‘V’ model and System Engineering baseline reviews are a good starting point for planning validation & verification activities.
Agenda

Introduction
What is architecture?
Requirements aspect
Functional aspect
Operational aspect
Using patterns
Validation aspect

Architect career in IBM
Within IBM there are several IT Architect disciplines covering different aspects of architecture, representing best practices, advises on skill development and each is represented by a world-wide advisory body.
What does an IT Architect do;
We have can have the same context but different focus
Characteristics of an IT Architect

- Skills and experience producing architectures
- Appropriate technical skills and experience, including technical breadth
- Disciplined, method-driven execution
- Full life-cycle experience
- Leadership
- Strong personal and professional skills
  - The effective IT Architect brings a broad base of fundamental skills across many professional and technical disciplines to their work
    - A common set of fundamental skills

Architectural Skills
- Perform Technical Solution Assessments
- Develop client requirements and architectural decisions
- Lead in setting technical direction
- Use modeling techniques
- Apply IT standards in creation of solutions
- Architect solution for security
- Develop test strategies and plans
- Develop solutions architecture
- Apply methodologies
- Lead strategy, design, and implementation of solution
- Use existing work products
- Develop project output for future reuse

Project Management Skills
- Manage architectural elements of project plan
- Plan projects

Consulting skills
- Use consulting techniques
- Perform negotiations
- Manage client relationships

Leadership Skills
- Apply communication skills
- Lead individuals and teams

Industry knowledge
The IT Architect Career Model Focus is on Skills Development, and I can become Sam’s right hand….

Qualification levels build upon previous levels as part of the career development process
Agenda

Introduction
What is architecture?
Requirements aspect
Functional aspect
Operational aspect
Using patterns
Validation aspect
Architect career in IBM

Closing
The modeling and tooling problem

“Most architects use tools that are no better than those their kids use for doing their homework”
Hvorfor laver vi modeller?

For at
- Tænke
- Forstå
- Lære
- Beskrive
- Dokumentere
- Kommunikere
- Visualisere
- Abstrahere
- Håndtere kompleksitet ved at separere aspekter i forskellige “views” (separation of concerns)

Modeling is not an all-or-nothing proposition. Models can play a part in the software development process in many ways.

Gary Cernosek: “The Value of Modeling”
Basically my work is about requirements, solutions, processes and people

- Define/understand requirements
- Define architecture and technical solution
- Define method
- Define work breakdown
- Establish estimates
- Assemble team
- Lead technical delivery team
- Maintain requirements
- Maintain architecture
- Redefine architecture and technical solution
- Redefine method
- Redefine work breakdown
- Establish estimates
- ...
My work is about communication; communicating and discussing requirements, solutions and processes. I’m building bridges balancing opposing forces.
Do we ever reach "Model only"?

- Code only: "What’s a Model?"
- Code visualization: "The code is the model"
- Roundtrip engineering: "Code and model coexist"
- Model-centric: "The model is the code"
- Model only: "Let’s talk models"