Thinking and Acting Like an Architect

Bill Seubert – IBM North America z Architect Leader

SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

Copyright (c) 2015 by SHARE Inc. Except where otherwise noted, this work is licensed under http://creativecommons.org/licenses/by-nc-sa/3.0/

Acknowledgements & sources

- “Becoming an Architect” – Lee H. Waldrep, Ph.D
- “Architectural Thought – The Design Process and the Expectant Eye” – Michael Brawne
- Presentation: “Architectural Thinking” – David Jackson, Peter Bouchard

Topics

- What's an architect?
- Architectural and "top-down" thinking
- The builder analogy
- Architecture and design
- Design methods and Frameworks
- IBM's Team Solution Design
- An architectural example that's near and dear to us all

"One day Alice came to a fork in the road and saw a Cheshire cat in a tree. 'Which road do I take?' she asked. 'Where do you want to go?' was his response. 'I don't know,' Alice answered. 'Then,' said the cat, 'it doesn't matter.'"

– Alice In Wonderland by Lewis Carroll

* - Credit: Enoch Sears - http://www.lifeofanarchitect.com/what-does-an-architect-do/

What's an architect?

1. a person who engages in the profession of architecture.
2. a person professionally engaged in the **design** of certain large constructions other than buildings and the like: *landscape architect; naval architect.*
3. the deviser, maker, or creator of anything: *the architects of the Constitution of the United States.*

Word origin - 1550s, from Middle French *architecte*, from Latin *architectus*, from Greek *arkhitekton* "**master builder, director of works**," from *arkhi-* "chief" (see archon) + *tekton* "builder, carpenter" (see texture). An Old English word for it was *heahcræftiga* "**high-crafter**."

“I’m an architect”

“**Nothing is higher than architect**”– George Castanza

A “system architect”

(a.k.a. “IT Architect”)

- One who is engaged in the practice of creating systems architectures, primarily in a **client centric** role, through use of a **methodical** process employing a combination of **artistic** and **engineering** approaches.

- *“...an individual engaged in the process of architecting, regardless of domain, job title, or employer; by definition and practice both. From time to time an architect may perform engineering and an engineer may perform architecting – **whatever it takes to get the job done**.”*
  - *Maier and Rechtin*

What does an architect *do?*

- Architects don't just draw pictures (on a whiteboard…)

  "*In designing buildings, architects* **communicate** *with and assist those who have* **needs** – **clients**, **users** *and the public as a whole* – **and those who will make** *the spaces that satisfy those needs – builders and contractors, plumbers and painters, carpenters, and air conditioning mechanics*" (Waldrep)

- A large part of the role is about **communication**
- Requirements ("needs") gathering is huge – again, communication!

Architects at IBM have several roles

- Solution Designer
- Methodologist
- Technology Advisor
- Project Leader
- Facilitator
- Business Advisor

Note the variety of responsibilities– they are NOT all technical in nature!

- IMPORTANT: These roles all reflect the need for **architectural thinking**

What would happen if you put a carpenter, a plumber, an electrician and an HVAC installer in a room and told them to build a house?

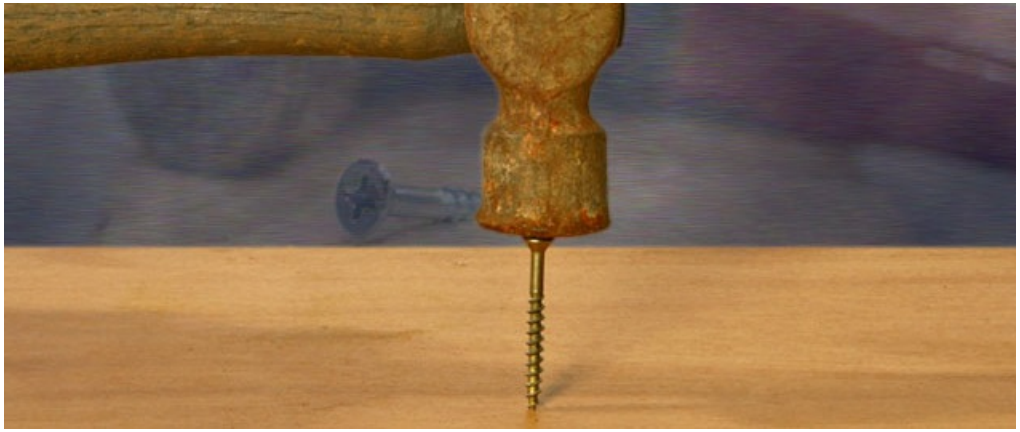
- This is often how we conduct I/T projects – a lot of very smart people who are very, very good at doing their jobs of installing operating systems, program products, networking, etc.
  - No one to cast the vision
  - No one to ensure the project fulfills the business or IT strategy
  - No one to establish standards and ensure they're followed
  - *Sometimes* there's a project manager
  - *Usually* someone is paying attention to what it costs
  - *Sometimes* the project gets done
  - Often it's over budget

  Sometimes it does what it's supposed to.

- How do we avoid such issues?
  - **An architect.**

  Well, at least it would help. IMO, a lot.

Can specialists architect and design good systems?



- Maybe.

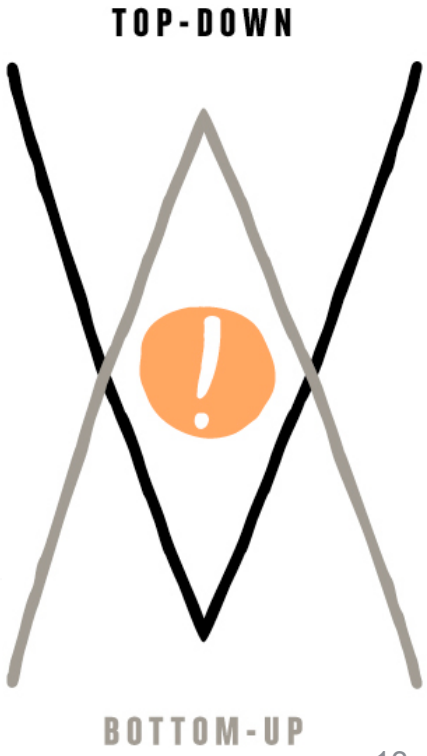
- "Builder-architected" systems
  - "form-first" architectural approach, with **technology-driven** systems rather than purpose driven systems (think: requirements, or lack thereof!).
  - Builder-conceived architecture rather than client-accepted purposes ("business drivers").
  - Uncertainty of end purpose is a *major risk*
  - Can become a "solution looking for a problem"

- Builders/specialists tend to think "bottom-up" and base designs on existing assets (hammer/nail) rather than *a top-down consideration of requirements that lead to a solution*

Architectural thinking and “top-down” vs “bottom-up”

- A “purpose orientation” drives architectural thinking
  - Systems architecting is driven by a client's purpose – the “top”
    - President Kennedy didn't say build me an Apollo 3 stage rocket and a Lunar excursion module.
  - Good architectures have a ***useful purpose***, affordable cost, and an acceptable period of time

- Good architects have a lot of experience and have probably made a lot of mistakes…
- As a result of the wisdom and mistakes, they have identified **patterns** that WORK
- Using patterns helps the architect think top-down



Patterns

- Architects are observers of behavior.
  - "...a place is given its character by certain **patterns** of events that keep on happening there…"
- Architects create space where these patterns of behavior can happen, flourish, and be generative – "be alive"
- From this activity a language of patterns emerges permitting endless possibilities of creation.

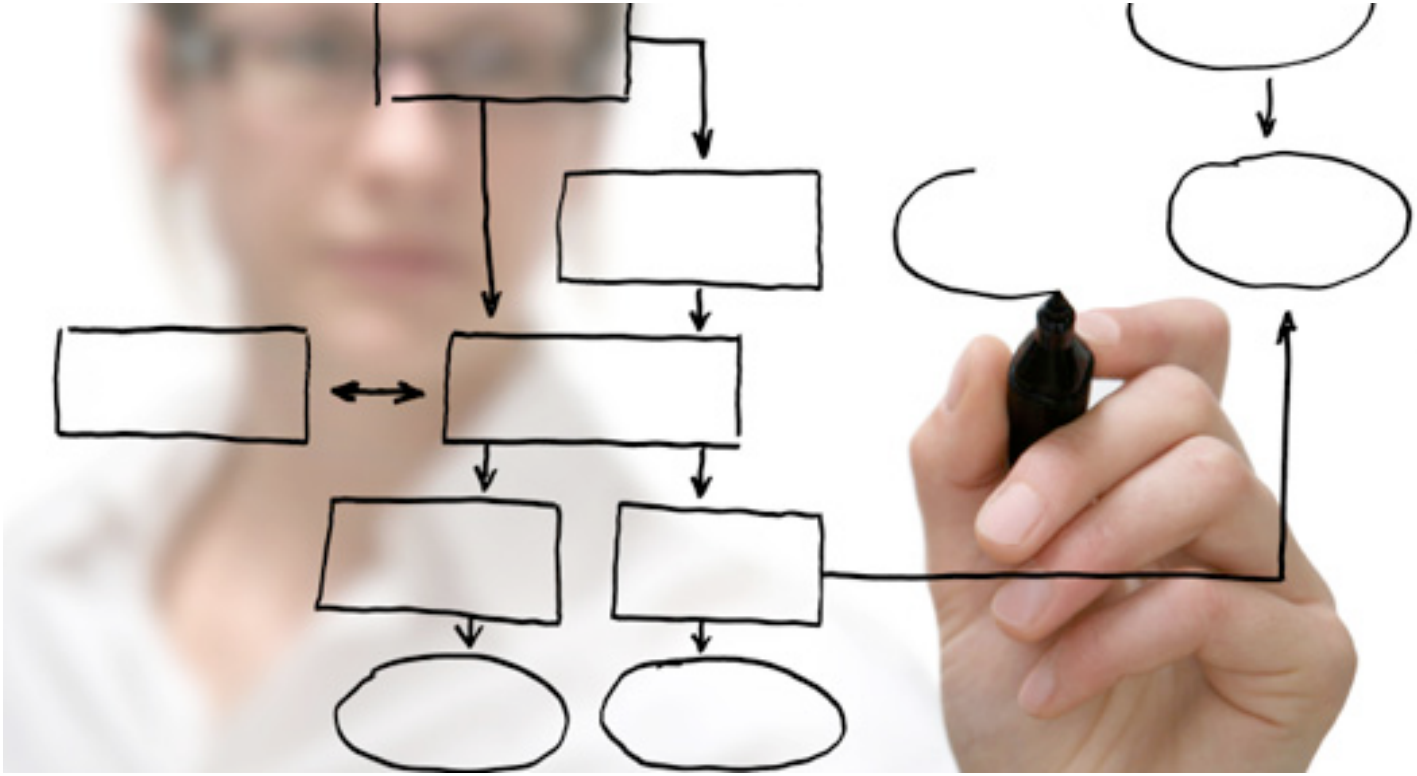
HGTV "Property Brothers" - "The Open Concept"

Taken from "Architectural Thinking" presentation – Jackson/Bouchard

Observing produces knowledge, but observing over time produces wisdom

- Information, knowledge and wisdom*:
  - **Information** is a sequence of symbols that can be interpreted as a message.
  - **Knowledge** is a unique set of facts and skills acquired by a person through experience or education.
  - **Wisdom** reflects understanding of "universal truths" or basic laws or patterns; it is knowledge that is based on values, meaning systems, and understanding that clarity is not always possible and that unpredictability and uncertainty are part of life.
- It's tough for an architect to function without all 3 of these
  - Wisdom comes with time and experience.
  - Seldom are architects new to IT…no wisdom/ experience in what does and doesn't work

* - Taken from: http://staroversky.com/blog/information-knowledge-and-wisdom-what-is-the-difference



How?

Solution designs, Frameworks & methodologies

Architects produce architectures and solution designs

- An architectural **framework** is often used:
  - a tool for:
    - Designing a broad range of a architectures
    - Assisting the evaluation of different architectures
    - Selecting and building the right architecture for an organization
  - It embodies best practice and acknowledged wisdom
  - It presents a set of services, standards, design concepts, components and configurations
  - It guides the development of specific architectures

- Examples: The Open Group Architecture Framework (TOGAF), Zachman, C4ISR (DoD)

Architecture and Design

- **Architectures drive solution designs**
  - The architect will create a higher-level abstraction, standards and guidelines that dictate how various solutions can be designed in their enterprise – *the architecture*
  - The solution design process follows the architecture

Architecture and 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, IBM Fellow (and father of UML)

- Tom Graves, on the topic:
  - Architecture and design are closely related; the main difference between them is really about which way we face.
  - **Architecture faces towards strategy, structure and purpose, towards the abstract.**
  - **Design faces towards implementation and practice, towards the concrete.**
  - Most designers and architects will do both types of work; but most will describe themselves as either a 'designer' or an 'architect' according to which way they most often face.
  - Architecture without design does nothing: it can too easily remain stuck in an 'ivory-tower' world, seeking ever finer and more idealized abstractions.
  - **Design without architecture tends toward point-solutions that are optimized solely for a single task and context**, often developed only for the **current** techniques and technologies, and often with high levels of hidden 'technical debt'.

Creating an architecture involves “looking to the future”

- As an architect, one must not only look at solving the current problem but also to creating a solution architecture that will “age gracefully” and will be somewhat “future-proof”
  - So, architecture, to some extent, requires a bit of **extrapolation into the future**.
    - Will an architecture be useful into the future?
    - Are there interfaces that allow for expansion?
    - What will the “eventual outcome” be for the system being designed?
    - Is the architecture built to be “future-proof”

- Can you think of an architecture that matches that description?

  (Did you catch the hint?)

The heritage of a successful architecture

SHARE

IBM

IBM® Mainframe50

Celebrating 50 years of groundbreaking innovations

Industry Milestones

1964 **SABRE** – Airline Reservation System

1965 **Time Sharing** – Reduced Cost of Computing

1966 **Medicare** – Social Security Administration

1967 **IBM 9020** – Air Traffic Control System

1969 **Breaking Space Barriers** – The Apollo 11 Mission

1969 **Magnetic Stripe Technology** – Credit Card Revolution

1970 **Relational Databases**

1972 **SAP** – Real Time Inventory Tracking

1972 **VM Virtualization**

1976 **NYSE DOT** – Stock Trading

1976 **Business Intelligence** with SAS

1979 **UPC** – Retail Transformation

1980 **CSC's Hogan** – First Integrated US Banking System

1981 **PROFS** – Pioneered Corporate e-mail

1983 **4730 Personal Banking Machine** – A One-Stop ATM

1994 **CompStat** – Data-Driven Approach to Policing (NYPD)

1996 **DEEP THUNDER** – Micro-Weather Modeling System

1999 **Open Source** with Linux®

2000s **Mobile Banking**

2008 **Nationwide Smart Grid** – World's First National Smart Utility Grid

2009 **Stream Computing** – Critical Insights on the Fly

2010 **Hybrid Computing** with IBM zEnterprise®

2013 **Ford: Connected Cars** – Real Time Traffic Info

Technology Milestones

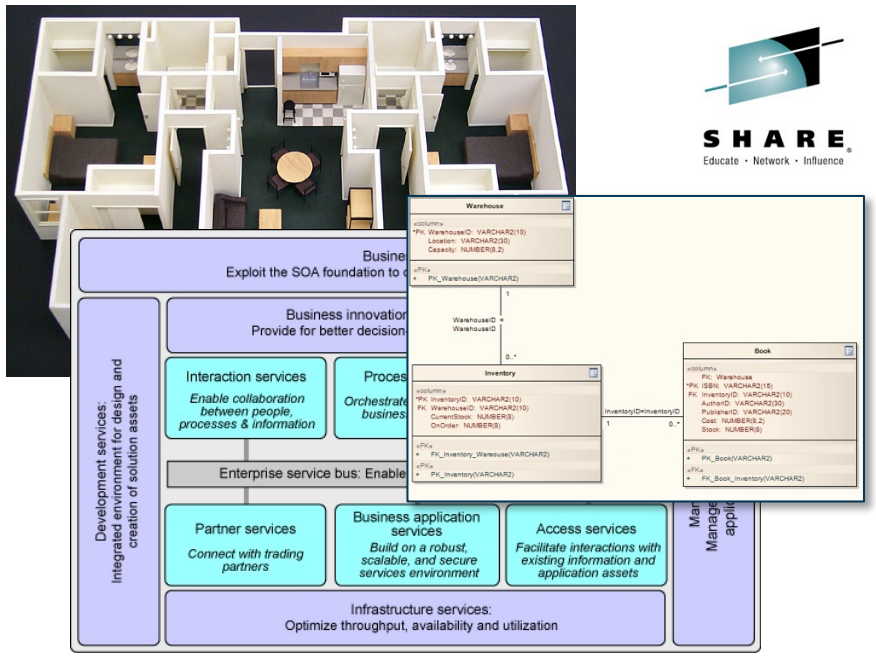
© Copyright IBM Corporation 2014. IBM, the IBM logo, and zEnterprise are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.
Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Architectural Models and Modeling

- What are models?
  - The product of the architect is “a vision” - intangible
  - Modeling creates a *steady reduction of abstraction* from vision to reality.
  - Eventually the models become construction drawings and itemized budgets, and pass into the hands of the builders.
- Why model?
  - Match the desirability of purposes with the practical feasibility of a system to fulfill those purposes.

Using models to reflect the reduction of abstraction reflects “top-down” architectural thinking that is necessary to be an architect!

Kinds of models?



- Models of form
  - Physically identifiable elements of, and interfaces to what will be constructed.
  - POCs
  - Block diagrams
- Behavioral (functional) models
  - Describe specific patterns of behavior by the system
  - Use Cases
  - Data and event flow – allow threads to be collapsed into more compact models
- Performance models
  - Describes or predicts how effectively an aspect of the architecture satisfies some function
  - These are “ilities” or nonfunctional requirements
- Data Models
  - What data does the system retain and what relationships among the data does it develop and maintain ?
  - Entity-Relationship diagrams for relational databases

Modeling, to the extreme

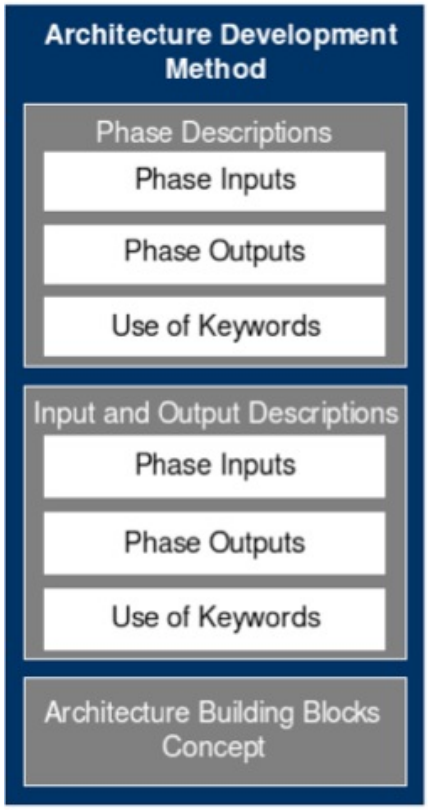
- Gillian: “I never imagined that!”
- Roy: “Next time try sculpture.”

Turn thought into action by using a methodology

- Part of the definition of an architect is someone who **uses a methodical approach.**
- The use, transformation, and creation of method tools is seen as a sign of maturity in an architect.
- As a craft person becomes more mature in their practice, they grow in their facility to use the tools of their craft in different ways.
- But, one's use of methods should not be “mechanical”

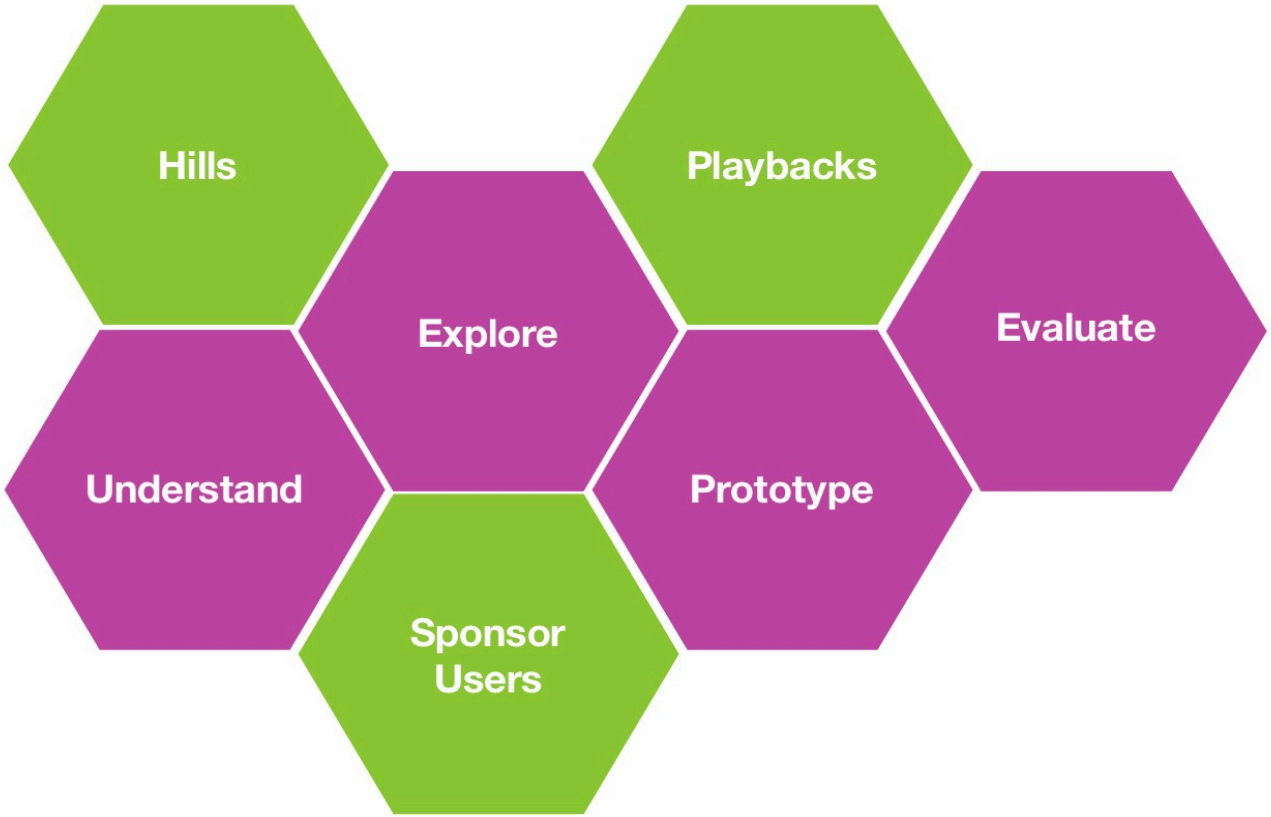
- **Methods provide structure**
  - Methods provide structure to our thought processes to give us an idea of “the next thing to do”
  - “Muscle memory” will eventually take over.
  - But, you really have not left method behind, it still informs what you do.

The Architecture Development Method of TOGAF

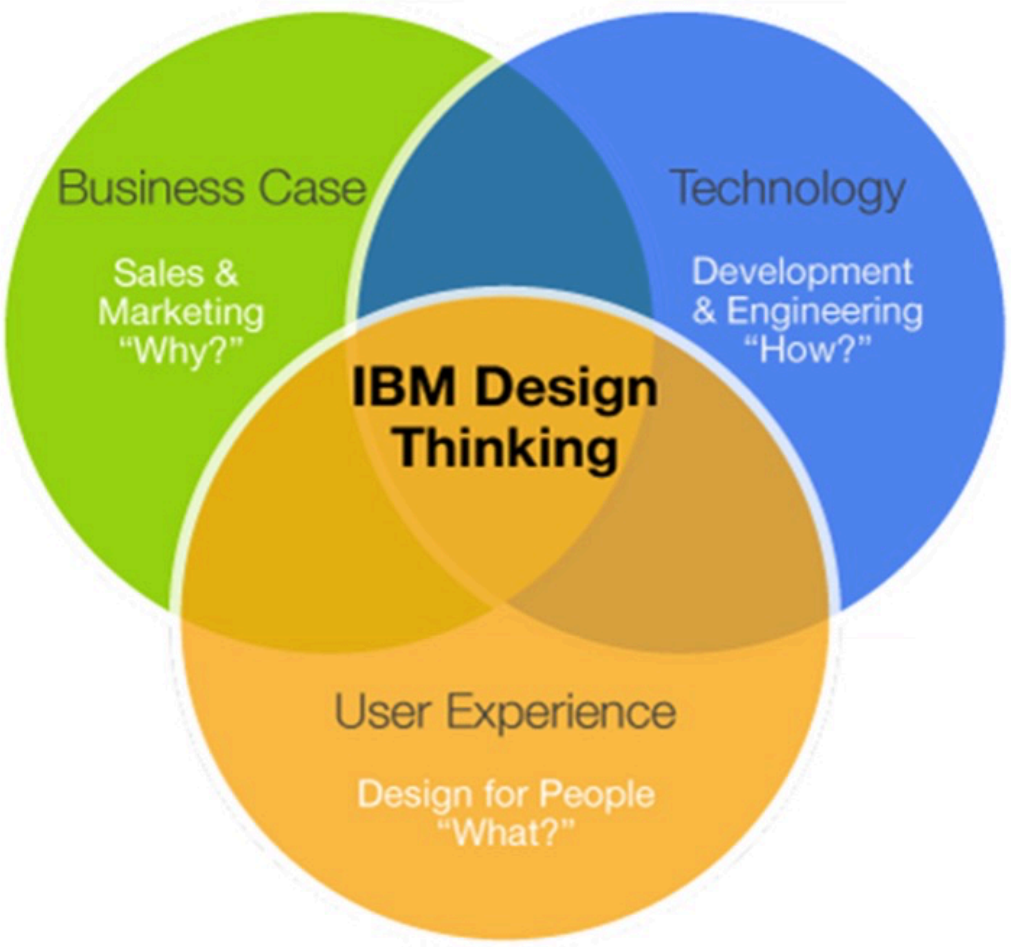


An iterative process for developing architectures

IBM Design Thinking



IBM Design Thinking



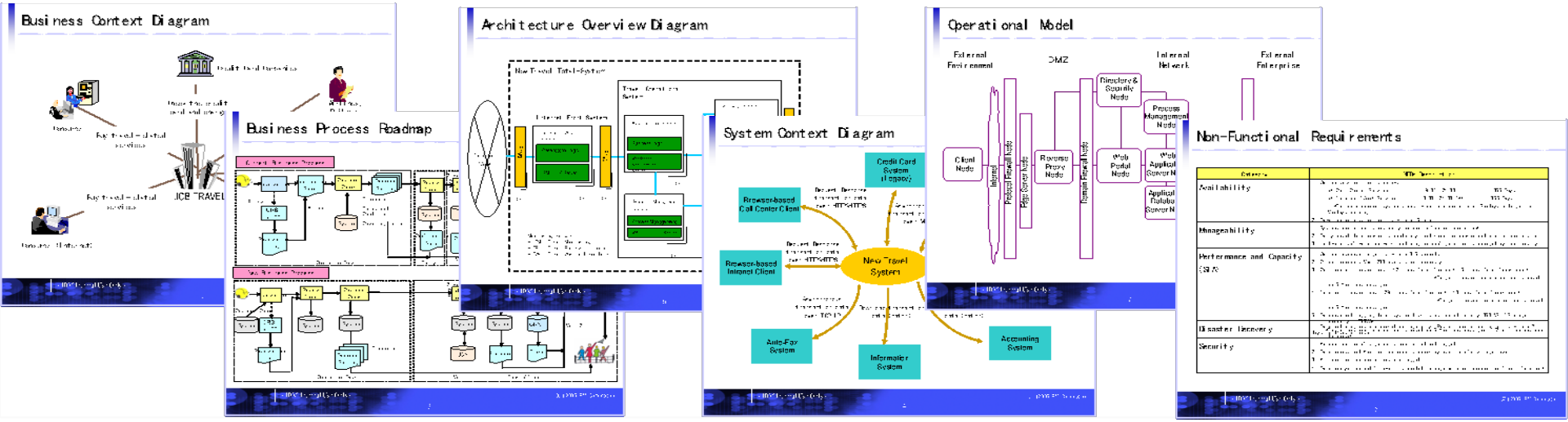
IBM's Team Solution Design methodology

A single pre-sales solution design method by all brands and service organizations.

Team Solution Design includes standard document types, tasks, assets and guidance that provide IBM's disciplined, standard approach to pre-sales design best practices through delivery, across all brands and service organizations.

- Increase our ability to **design, propose, and build solutions**
  - **Reduced risk** with **predictable results and higher quality**
- Enable effective **communications between sales and delivery**

Team Solution Design is accepted by The Open Group as a standard design methodology and IBM's architect certification is interlocked with Open Group.

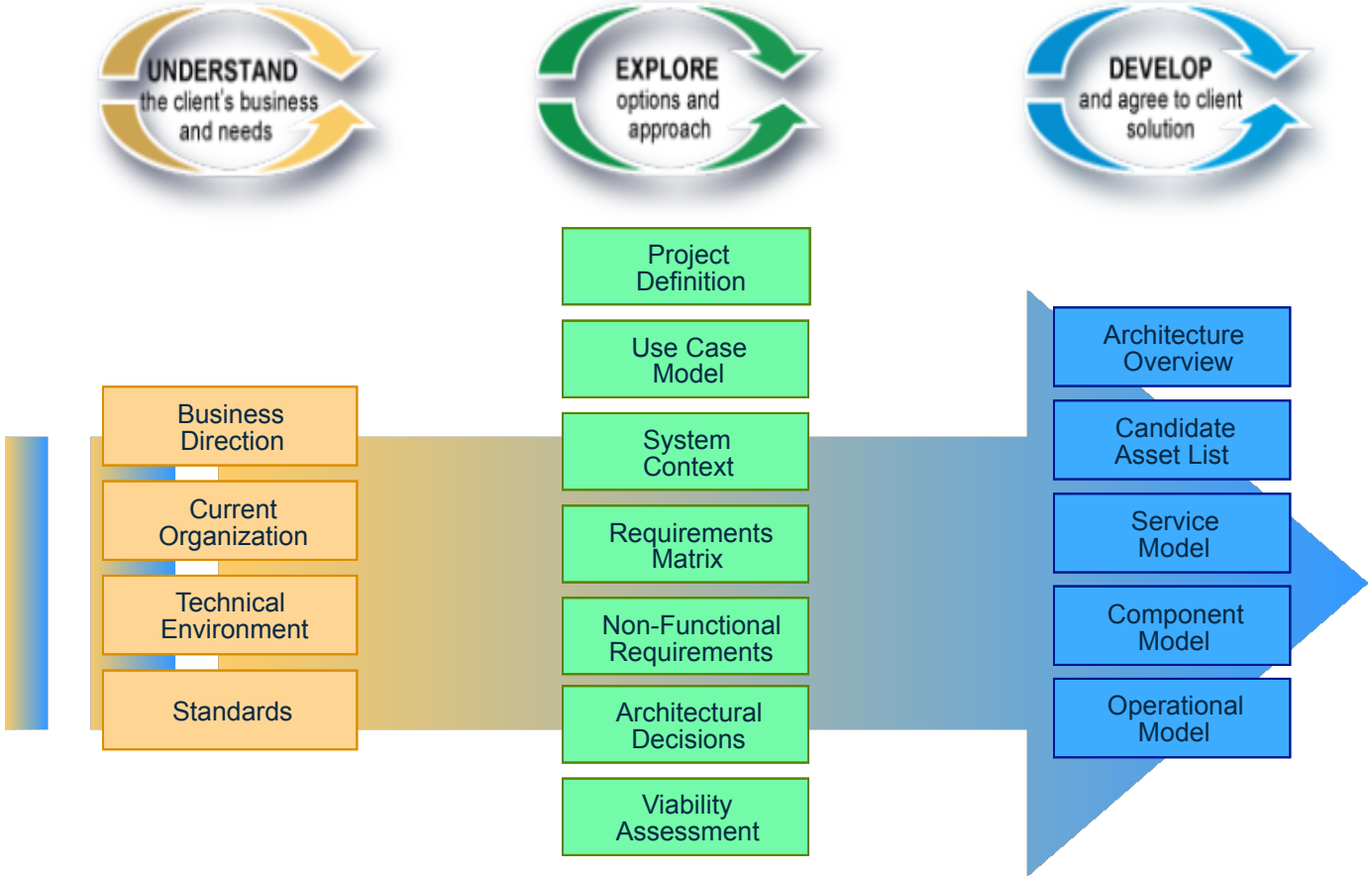


Phases and steps in Team Solution Design

- Plan
  - UNDERSTAND Client's Business and Needs
    - Understand Business Environment and Objectives
    - Describe Current Organization
    - Describe Current IT Environment and Plans
    - Identify Opportunity
    - Opportunity Validated
- Pre-sale Solution Design
  - EXPLORE Options and Approach
    - Define Project
    - Describe System Context
    - Identify Non-Functional Requirements
    - Identify and Outline Requirements
    - Identify High Level Data Sources
    - Document Architectural Decisions
    - Conduct Viability Assessment
    - Opportunity Qualified
  - DEVELOP and Agree to Client Solution
    - Develop Architecture Overview
    - Survey Candidate Assets
    - Define Key Services
    - Develop High Level Component Model
    - Develop High Level Operational Model
    - Identify Delivery Approach
    - Develop Solution Estimates
    - Refine Viability Assessment
    - Evaluate Integrated Solution
    - Propose Solution, Resolve Concerns
    - Solution Agreed To
- Support Implementation and Confirm Value
  - IMPLEMENT Client Solution
    - Transition to Implementation
    - Monitor Pilot and Early Implementation
    - Harvest Assets
    - Project Implemented
  - CONFIRM Client Value and Experience
    - Evaluate Success
    - Explore New Client Issues
    - Value Confirmed

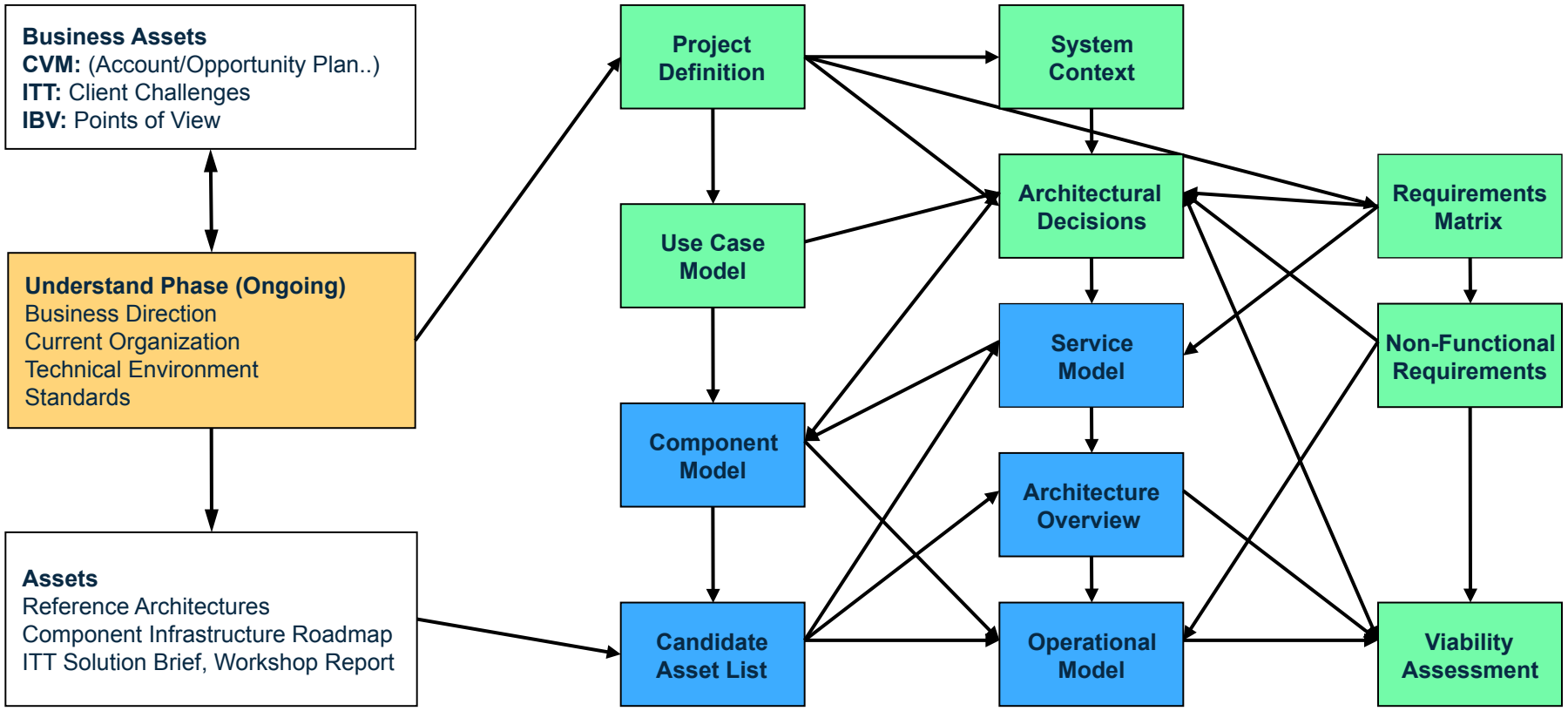
- Each task has certain roles and input/output work products associated with it

Key Team Solution Design Work Products by Phase



Not all of the work-products will be needed on every project.
Work-products will be dependant on the solution/service type.

Team Solution Design Work Product Dependencies



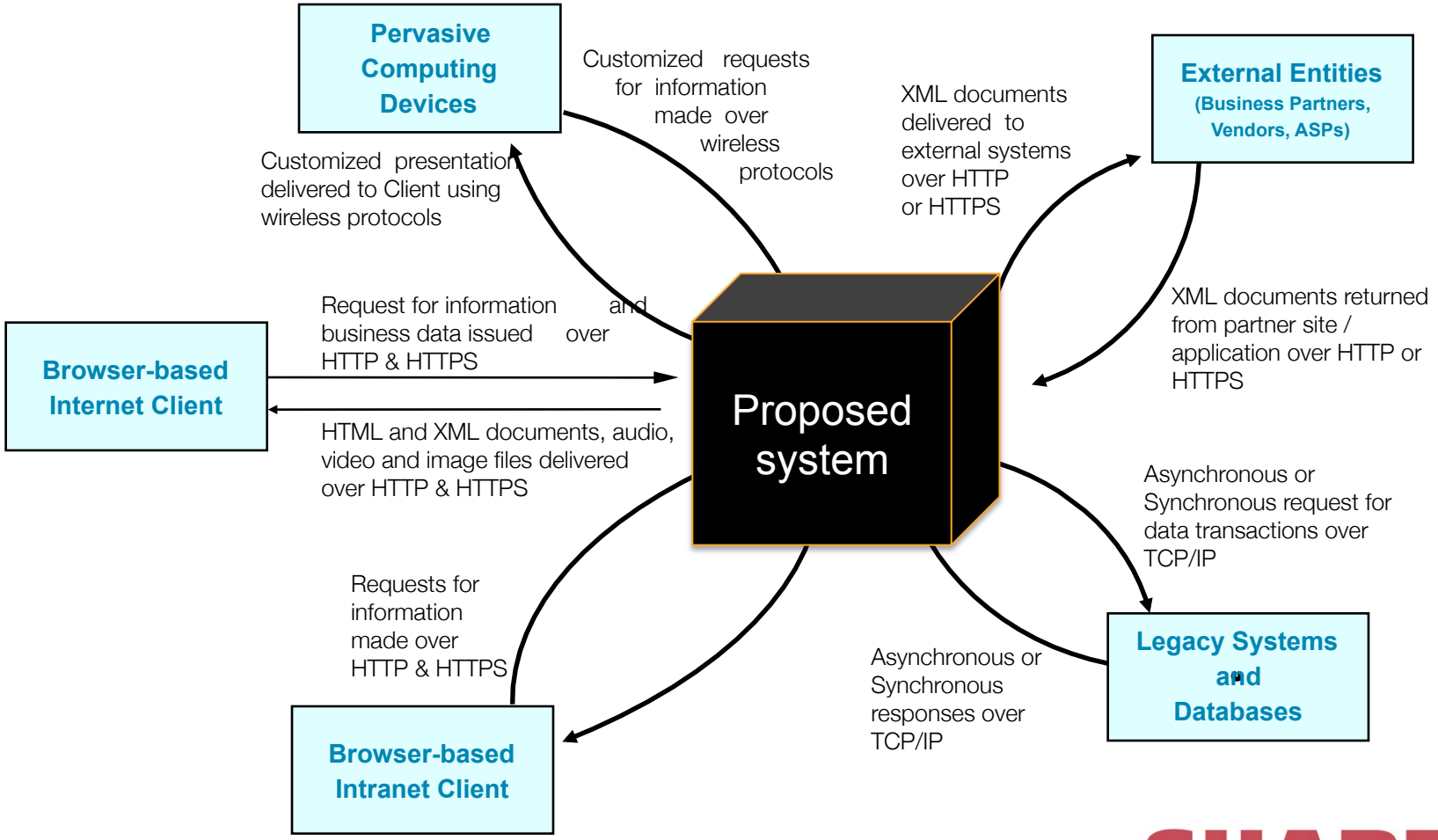
NOTES: A work product is a tool that can be used to define and describe the items needed as input or created as output of one or more tasks. Colors indicate where the work products first get created. They continue to be updated throughout the Client project.

Example: Project Definition

Contents

1. Introduction 4
2. Project Objectives 5
   - 2.1 Business Needs and Environment 5
   - 2.2 Scope and Plan Constraints 6
   - 2.3 Solution background 6
   - 2.4 Requirements 7
3. Target solution and Overall Approach 10
   - 3.1 Overview of the Target Solution 10
   - 3.2 Components 10
     - 3.2.1 Management 11
     - 3.2.2 Compute Farm 13
     - 3.2.3 Data Gateway 15
     - 3.2.4 Data Farm 16
   - 3.3 Technical Environment and Key Options 17
4. Project Scope 18
   - 4.1 Breadth 18
   - 4.2 Major Deliverables 18

Example: System Context



Example: Requirements Matrix (simple)

Functional Requirement ID	*Functional Requirement Name*	*Functional Requirement*	*Requirement Supported by Software / Component*	*Comments*
FR001	Backup/Recovery	Facilities and procedures created and defined to support backup and recovery of data at the plant location		
FR002	Archive data	Provide processes to periodically remove and archive on-line data		
FR003	Electronic notification	Ability to receive electronic notification/report of orders on a daily basis w/detail available		
FR004	Track goods	Ability to classify and track goods, materials/supplies by code, by property number, by cost center, by employee number, by location, by date of purchase, by warranty of equipment, by insurance code		Business requirement driven by initiative to reduce inventory.
FR004.1	Ad hoc reporting	Provides robust ad-hoc reporting facility and tools	cmpTool	This IT requirement needed to support FR004.
FR004.2	Automatic scheduling	Periodic reports, distribution of data, and system backups can be automatically scheduled and managed	cmpTool	This IT requirement needed to support FR004.
FR004.3	Inventory levels	Includes ability to set min/max for inventory items	cmpTool	This IT requirement needed to support FR004.

Example: Non-Functional Requirements

Question	Response	Explanation
Number of Loans Processed	20,000	Revenue model estimates 10,000 loans processed in the year 1. All these loans are processed in the 4 month peak period
Total number of anticipated users of the system	120	The number of anticipated users of the system is a fraction of the overall population of end users. How an organization would promote the usage of an e-business applications would determine what percentage of the overall population of end users would use the system.
Anticipated number of Concurrent Users	50	This value was based upon 20000 applications over a 4 month time = 5000 applications per month. 5000 application over 30 days = 166 applications per day. Assuming a scenario in which all applications are created in 3-4 hours window, the anticipated number of concurrent users is approximately 50
Expected Availability	12 hrs / 6 days per week.	Availability is best described in terms of XX% Uptime instead of 24X7. 24X7 could mean 100% availability and such requirements result in extremely expensive infrastructure.
Expected Response Time	30 sec	The response time could vary substantially between static and dynamic pages. In the case of dynamic pages involving backend data processing the response time would depend on response time provided by the backend systems.

Example: Architectural Decisions

	Architectural Decision
Subject Area	Enterprise Connectivity
Architectural Decision	Means to communicate between with backend systems and databases
Problem Statement	The e-business application should integrate with several backend systems and databases. The question is how to best communicate with these systems.
Assumptions	Interactions with back-end systems will be a mix of synchronous and asynchronous communications.
Motivation	Because of the degree of application integration needed, a decision must be made on enterprise connectivity to avoid inconsistent and unmanageable connectivity.
Alternatives	1. Point-to-Point Connectivity This alternative connects the application server connect directly and individually to each necessary enterprise system, using the protocol best suited for that system. For example, connectivity to the mainframe could use Common Programming Interface for Communications (CPI-C). Alternatively, messages could be exchanged individually with each system using a messaging product such as Message Queuing Series (MQSeries) or Electronic Data Interchange (EDI). A very basic option would be to exchange data via File Transfer Protocol (FTP). 2. Integration Hub This alternative centralizes all enterprise communications at a common intermediary called an Integration Hub. Business-level messages are given to the hub for delivery. The hub routes the message to the correct enterprise system, translating the message if necessary into a format acceptable by the enterprise system and communicating with the enterprise system in an appropriate protocol. This hub can be used to support synchronous and asynchronous integration. 3. Business Workflow This alternative adds onto the Integration Hub by adding business process functions to its capabilities. A message would trigger a set of further messages to different destinations based on a pre-defined workflow for that business process.

Architecture of the IBM 360

- Architectural decisions are driven by **design principles**, standards, etc.
- The IBM 360 architecture is a marvelous case study in following design principles Some examples:

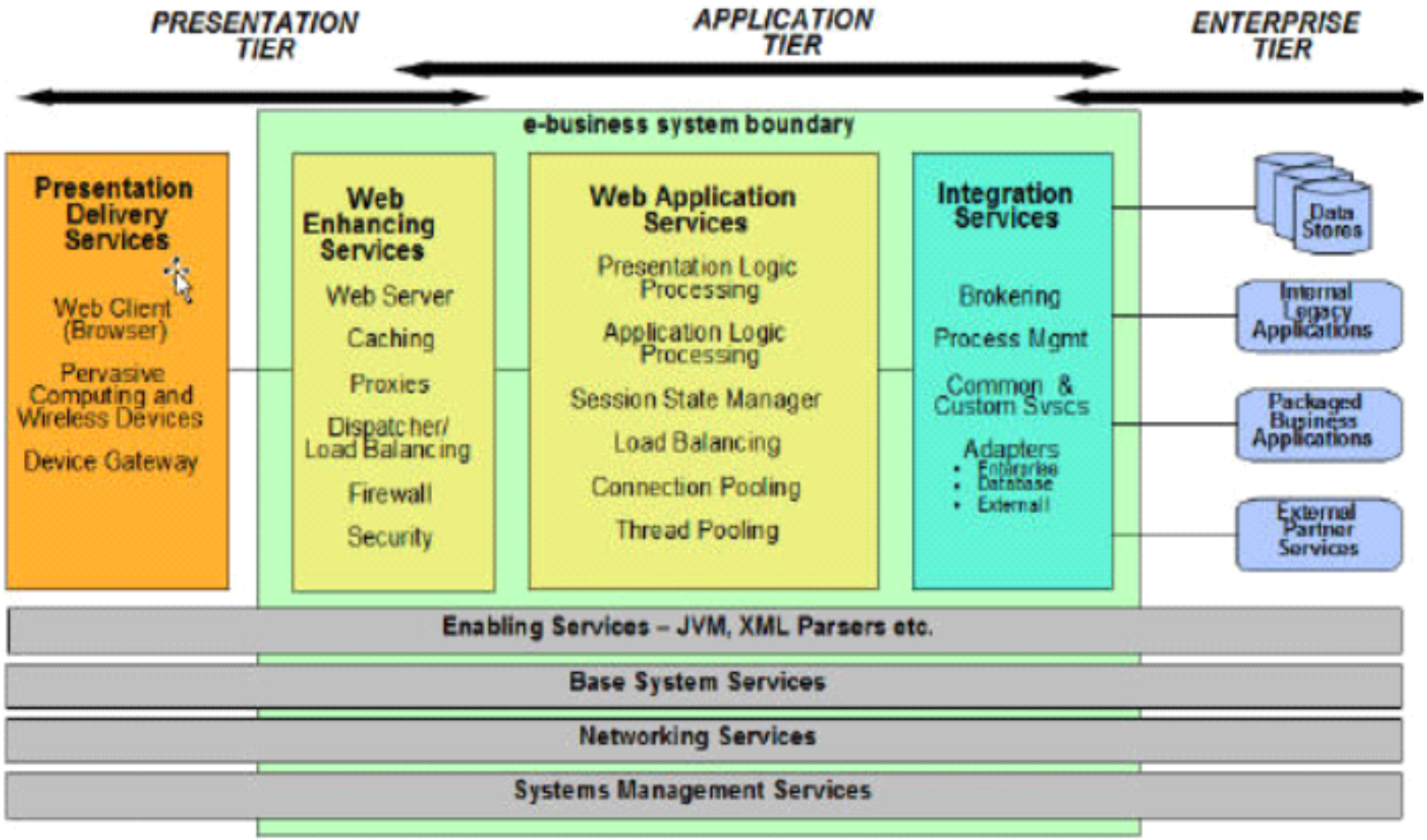
1. An approach to storage which permits and exploits very large capacities, hierarchies of speeds, read-only storage for microprogram control, flexible storage protection, and simple program relocation.

2. An input/output system offering new degrees of concurrent operation, compatible channel operation, data rates approaching 5,000,000 characters/second, integrated design of hardware and software, a new low-cost, multiple-channel package sharing main-frame hardware, new provisions for device status information, and a standard channel interface between central processing unit and input/output devices.

3. A truly general-purpose machine organization offering new supervisory facilities, powerful logical processing operations, and a wide variety of data formats.

4. Strict upward and downward machine-language compatibility over a line of six models having a performance range factor of 50.

Example: Architecture Overview



How does this apply to IBM z today?

- Obviously the aforementioned 360 architecture and its development over the years
- Enterprise architectures that include the mainframe
- Solution designs that encompass portions of your z/OS or Linux on z implementations
  - Mobile
  - Analytics
  - Services/SOA
- Be ready to provide IBM z “participation” in enterprise system designs
- Know how to talk to your enterprise architects

Let's summarize…

- Architects should **think like architects** – top-down
- Architects should be **methodological**
  - But the architectural method-of-choice need not be a mechanical thing
- Successful solution **designs are based on architecture**
- A good guideline:
  - Existing environment – business and I/T
  - Requirements – business and I/T, functional and non
  - Consider the alternatives
  - Assess the risk
  - Use models to represent various facets of the design
- The architect must be involved *through the lifecycle* of the solution

The QR code

Thank you!

More on the IBM 360 architecture

- The IBM Systems Journal ran an article on IBM 360 design in 1964:
  - http://www.eecs.berkeley.edu/~culler/courses/cs252-s05/papers/amdahl.pdf (Gene Amdahl was one of the authors)
  - http://www.cs.tufts.edu/~nr/cs257/archive/alfred-spector/spector87ibm.pdf - great review of 360 as an architecture case study

Team Solution Design
Additional details charts

Work Product: Project Definition

Purpose

- Formalize the understanding of the project
- Serve as a basis for planning and controlling project activities
- Document who, what, where, why, when and how?
- Updated throughout the duration of the project

Description

- The project definition is a text document that addresses the key questions and issues necessary to define and manage a project.
- Examples of content include:
  - Project purpose and business problems to be to addressed
  - Project timeline and approach
  - Primary, high level functional aspects of the system
  - Information on sponsors, participants and responsibilities

Work Product: System Context

Purpose

- To clarify and confirm the environment in which the system has to operate.
- To document information flows between the solution to be installed and external entities.
- Provide a basis for establishing scope of the target solution and external dependencies.

Description

- The System Context is usually represented in a diagram showing the entire proposed system as a single “black box” with relationships to users and external devices and systems.
- Documents important characteristics of the system context such as users, external interfaces and systems, inputs and outputs, and external devices.
- Identifies boundary definitions and the information and control flows with external systems.

Work Product: Requirements Matrix

Purpose

- To capture Client requirements and to evaluate the initial functional "fit" of alternatives.
  - This artifact documents important Client functional requirements of the proposed system in a text form.
  - It provides an initial means to compare multiple alternatives against each other and to ensure that the products and technology selected are suited to their purpose.
  - It is used to identify initial functional gaps or special software enhancements needed to fulfil the Client's desired system capabilities.

Description

- The Requirements Matrix is a document detailing the functionality and technical requirements desired by the Client in the solution.
- It is either a simple table or a spreadsheet with multiple tabs, each tab representing a separate business area or other subset of the system requirements.

Work Product: Non-Functional Requirements

Purpose

- To describe the quality attributes of the system and the constraints which the design options will be required to satisfy in order to deliver the business goals, objectives or capabilities.
- To provide a quantitative basis for assessing the sizing, cost and viability of the proposed system.
- Establish key considerations for understanding service level agreements for operational management of the solution.

Description

- Non-functional requirements are established in areas such as performance and capacity, availability, usability, security and privacy, maintainability, manageability and flexibility.
- Specific examples might include number of concurrent users, update response time, end-user availability, etc.

Work Product: Architectural Decisions

Purpose:

- Provide a rigorous approach for key architectural decisions which require considerable analysis and may have enduring influence on future designs.
- Ensure there is a single, authoritative source for communicating key decisions made about the architecture.

Description:

- Each significant architectural decision is documented in a table. Examples of information include:
  - What is being decided or the issue that is being addressed.
  - What options were considered
  - What decision was made, including justification and implications

Use established **architectural principles** to drive the decisions

Work Product: Architecture Overview

Purpose

- Provide a high-level shared vision of the architecture of the solution and its scope.
- Provide the sponsor and stakeholders a conceptual understanding of the intended architecture.
- Support evaluation of alternative architectural options.
- Enable early recognition and validation of the architectural approach.

Description

- This artifact illustrates the basic ideas of the proposed architecture, serving as means of confirming architectural understanding between IBM and the Client.
- Architecture Overview Diagrams may be created for different audiences and at different levels of detail.
- This description of the architecture is intended to be brief and understandable rather than comprehensive or accurate in all details (like Component or Operational Models).