

Solution Architectures Part I

What does an IT Architect need to be effective?



Learning Objectives

- ▣▣▣ At the end of this lecture, you should be able to understand:
 - ▣▣▣ The personal tools you need to be an IT Architect
 - ▣▣▣ The professional tools you need to be an IT Architect

- ▣▣▣ This will be illustrated through the three “Cs”:
 - ▣▣▣ Context
 - ▣▣▣ Common Sense
 - ▣▣▣ Communication

Characteristics of an IT Architect

- ■ ■ ■ Accountable for the integrity of the customer solution
- ■ ■ ■ Recognised lead technical authority
- ■ ■ ■ Able to architect full solution although focus will vary due to differing background and experience
- ■ ■ ■ Capable of delving into detail when required
- ■ ■ ■ Full life cycle, i.e. from concept, through development & roll-out to support/managed service
- ■ ■ ■ Strong leadership, technical & communication skills
- ■ ■ ■ Ability and willingness to mentor and coach

- ■ ■ ■ Overall, the architect is a specialist in reducing:
 - ■ ■ ■ Complexity
 - ■ ■ ■ Uncertainty
 - ■ ■ ■ Ambiguity
- ■ ■ ■ Creates workable concepts



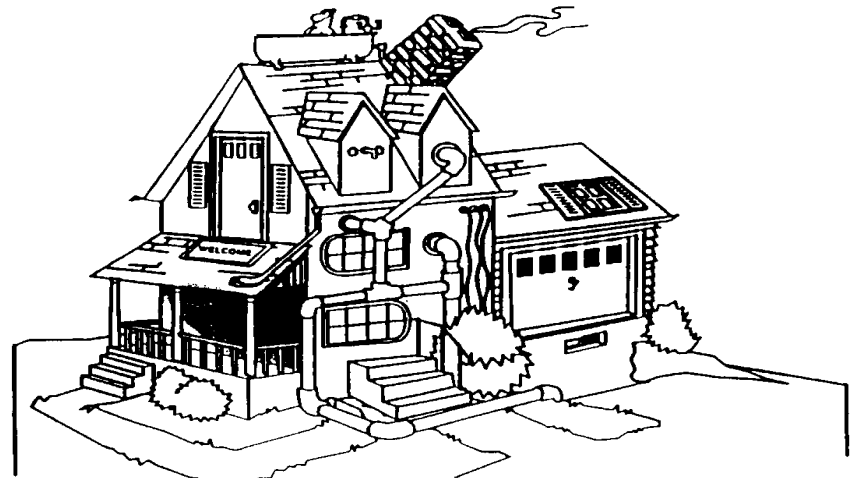
The three “Cs”

Context



The fundamental skill of an IT Architect is to understand the context they are working in

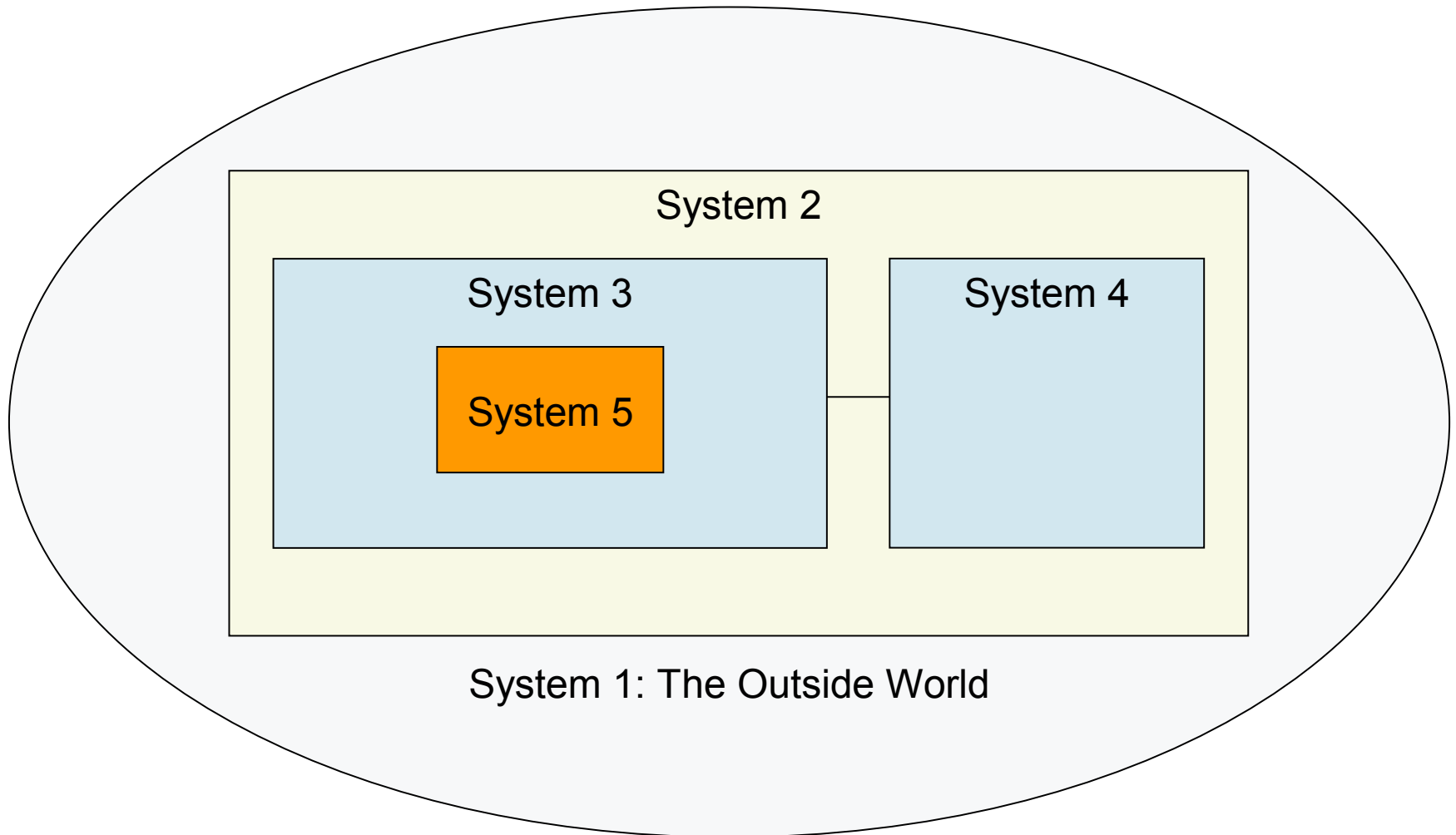
- ❑❑❑ Putting IT architecture in context itself - why do we need it?
- ❑❑❑ We will cover:
 - ❑❑❑ System context and boundaries
 - ❑❑❑ Architecture context - aspects of architecture
 - ❑❑❑ Project context



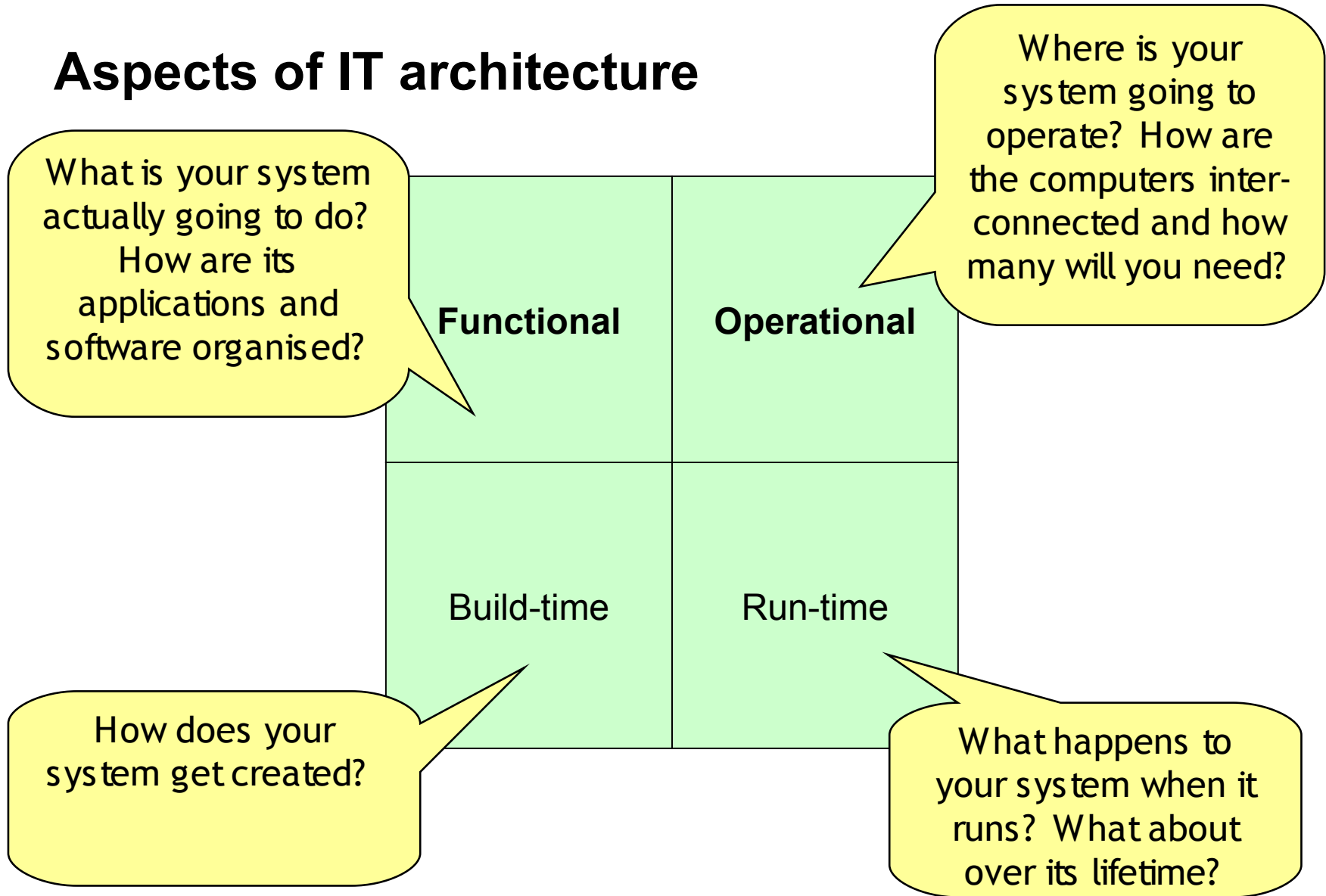
“IT architecture [has] proved to be successful in providing an **analytical and decision-making framework** for a sequence of new initiatives or **changes**, at the same time ensuring **design integrity** and stability”



Finding the system boundaries



Aspects of IT architecture



Architectural Thinking should lead to a complete systems architecture that serves multiple purposes.

- ❑ It breaks down the complexity of the IT system
- ❑ It analyzes the required functionality to identify required technical components
- ❑ It provides a basis for the specification of the physical computer systems
- ❑ It defines the structuring and strategy for connection of system elements
- ❑ It provides the rules of composition / decomposition of system elements
- ❑ It assists in the analysis of service level requirements to design a means of delivery
- ❑ It provides a decision trail which allows the system to evolve over time

Architectural decisions are made in context of the overall management of the project



Changing one side will always have an affect on the other two sides.

- ■ ■ You need to understand your project's context:
 - ■ ■ Be the project manager's best friend
 - ■ ■ Consider cost and budget implications
- ■ ■ How much will it cost?
 - ■ ■ Is it built to order?
 - ■ ■ Is it built to cost?

The three “Cs”

Common Sense



The Process of Architecting

❑ The Normative way

- ❑ Schools of architecture
- ❑ Innovation suppression
- ❑ Dogma

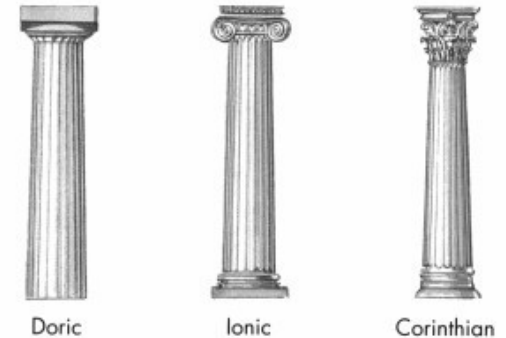
❑ The Rational way

- ❑ Procedure driven
- ❑ Logical

❑ The Argumentative way

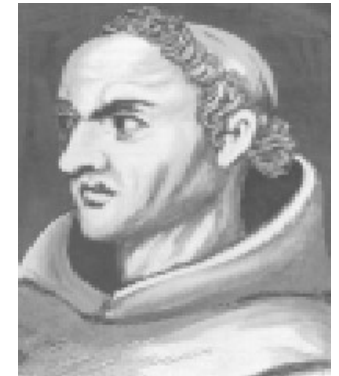
- ❑ Mechanistic
- ❑ Brainstorming
- ❑ Too many cooks

❑ The Heuristic way...



Just use common sense (heuristic reasoning)

- ▣ The knowledge of what is reasonable within a given context.
 - ▣ Includes insights, lessons learned and rules of thumb
- ▣ Heuristics can be prescriptive...
 - ▣ Keep It Simple, Stupid
- ▣ ... or descriptive
 - ▣ If anything can go wrong, it will.



▣ *The simplest solution is usually the correct one.*

Some favourites...

☐☐☐ Requirements

- ☐☐☐ Don't assume the original statement of the problem is necessarily the best, or even the right, one.
- ☐☐☐ Success is defined by the user, not the architect.

☐☐☐ Design

- ☐☐☐ You can't avoid redesign, or if first you don't succeed...
- ☐☐☐ No system can be optimum for all users (or all database accesses!)

Some more favourites...

Development

- Quality cannot be tested in, it has to be built in.
- Something good enough in a small system is unlikely to be good enough in a complex one.

Test

- Testing is a system in itself.
- Regardless of everything, the acceptance criteria determine what gets built.
- The sooner you find the problem, the cheaper it is to fix it.

The three “Cs”

Communication



Use a (system design) method

- ■ ■ The way a project communicates to achieve its goal

- ■ ■ Usually includes:
 - ■ ■ a language (notation)
 - ■ ■ a process model
- ■ ■ May also include:
 - ■ ■ work product descriptions
 - ■ ■ techniques
 - ■ ■ tools

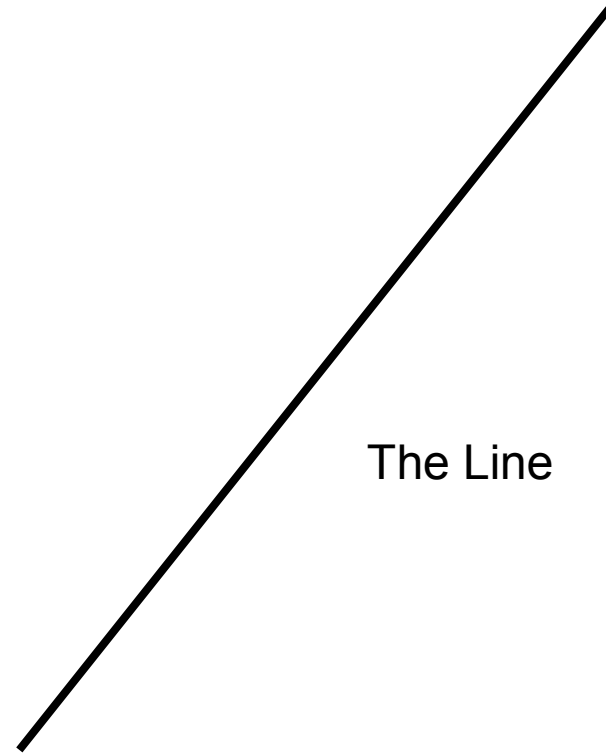
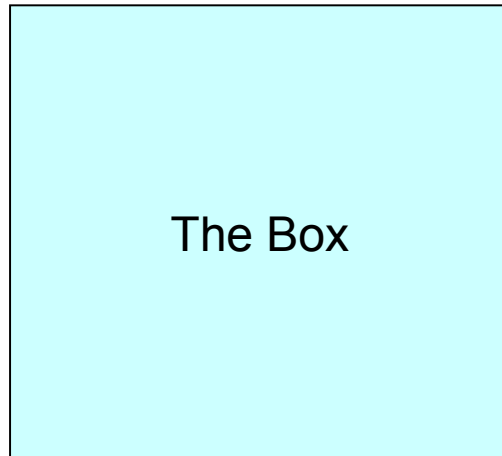


Functional	Operational
Build-time	Run-time

Why do we need a Method?

- ▣▣▣ Provides a mechanism to enable a common language among all practitioners delivering business solutions
- ▣▣▣ Fundamental component to accelerating the Global Services' shift to asset based services, providing a mechanism for practitioners to reuse knowledge and assets using a consistent, integrated approach
- ▣▣▣ Shifting from labour based to asset based services positions Global Services to compete more effectively in the marketplace by increasing productivity and minimizing cost, risk, and time to market

The weapons of the IT Architect...



As with all weapons, they need to be used carefully to stop you getting hurt.

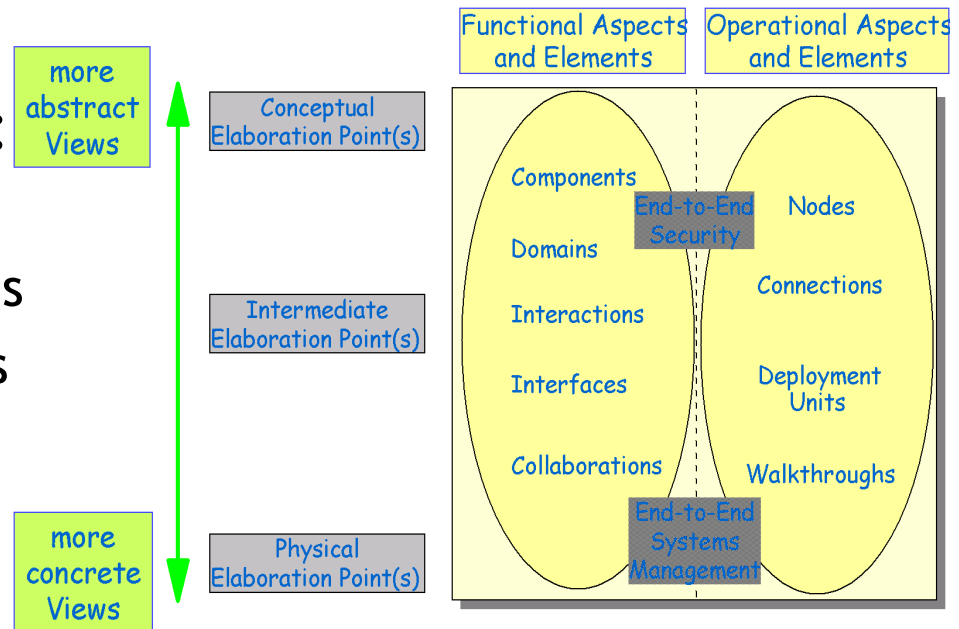
How do IBM IT Architects communicate their work?

❖ By using the Architecture Description Standard (ADS) !

❖ A common language (notation) for IBM IT Architects world-wide !

❖ ADS was developed to:

- ❖ facilitate reuse
- ❖ improve communications
- ❖ underpin IBM's methods



The ADS language underpins the Method...

Functional Aspects of Architecture -

Component

- a modular unit of (software) functionality, accessed through one or more interfaces (encapsulated)

Subsystem

- any subset of components in IT system

Collaboration

- collaboration between components realizes a use case scenario, consisting of a sequence of component operations

Interaction

- exchanges between two components, interface usage contract / protocol

Operational Aspects of Architecture -

Node

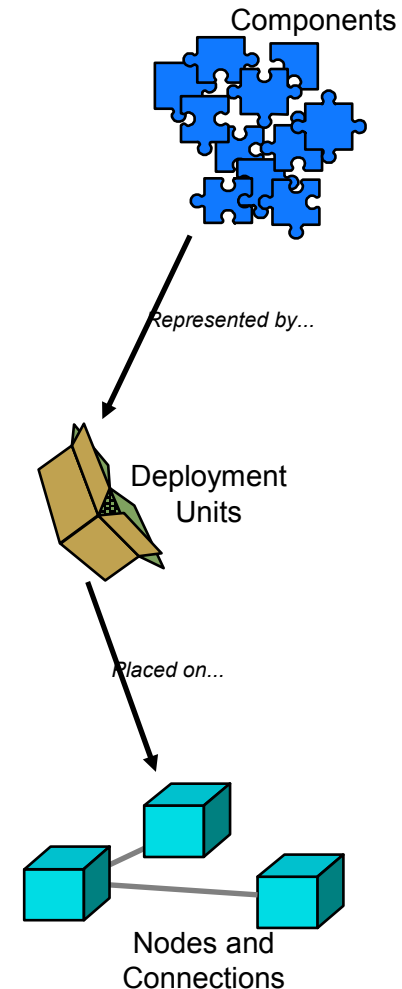
- a platform on which software executes

Connection

- the physical data path between nodes, LAN, WAN, dial-up etc.

Deployment Unit

- represents one or more components, placed together on a node and allows execution, presentation and persistence aspects to be separately placed

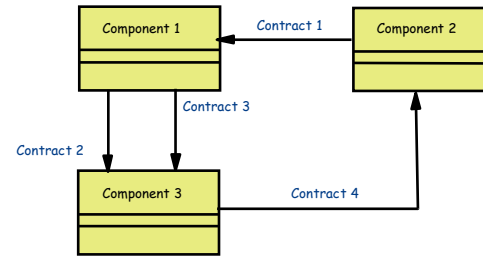


...and it provides key diagramming notations

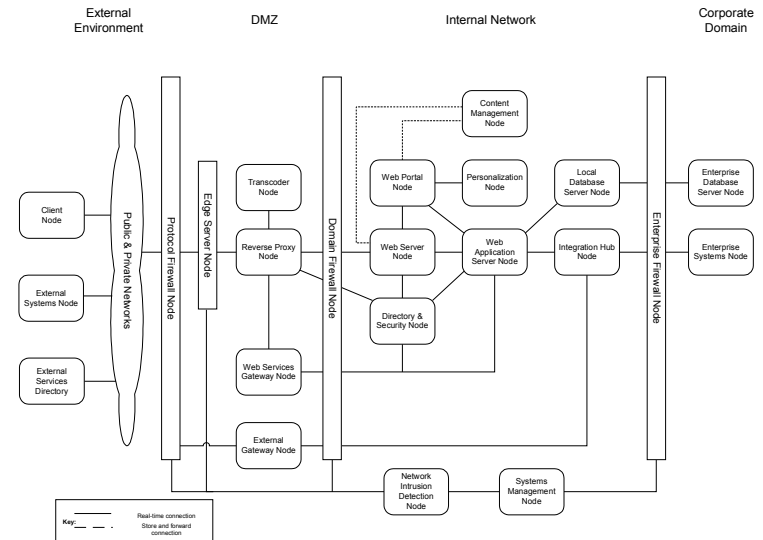
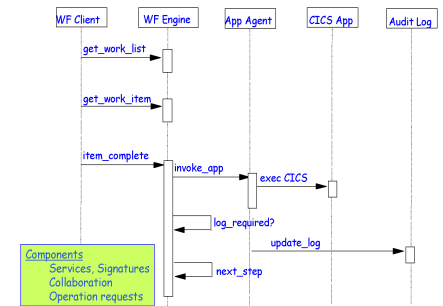
- ■ ■ UML for functional aspects
 - ■ ■ Component Relationship Diagrams
 - ■ ■ Class, Object Relationship
 - ■ ■ Use Cases
 - ■ ■ Deployment
 - ■ ■ Component Interaction Diagrams
 - ■ ■ Sequence Diagrams (Time oriented)
 - ■ ■ Object Collaboration (Message oriented)
 - ■ ■ State

- ■ ■ Various styles for operational aspects
 - ■ ■ Static Views of the Operational Model
 - ■ ■ Conceptual view
 - ■ ■ Specified view
 - ■ ■ Physical view
 - ■ ■ Dynamic views of the Operational Model
 - ■ ■ Walkthroughs

Component Diagrams, Relationships, Interactions



What are Component Interaction Diagrams



Communication skills are important for a successful IT Architect

Spoken Communication

- The IT Architect is often asked to present their solution
- Different audiences require different approaches
 - Business sponsor, project manager, developer...

Written Communication

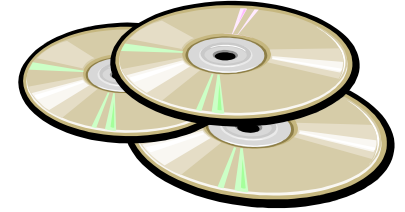
- An undocumented architecture is difficult to build and maintain; it leads to a lot of repeat discussions.
- Architectural documentation can be a time-saver, not a time-waster.
- Documentation is extracted from work products, but focuses on key messages and stakeholder needs

The kitbag

What every IT architect has on their laptop



The real tools



- ■ ■ The modern IT architect needs a number of tools to successfully communicate and manage their work
 - ■ ■ Requirements traceability database
 - ■ ■ Drawing tool
 - ■ ■ Word processor
 - ■ ■ Spreadsheet
 - ■ ■ Model-driven design and development platform

- ■ ■ Can you still can do it with a pencil and paper?

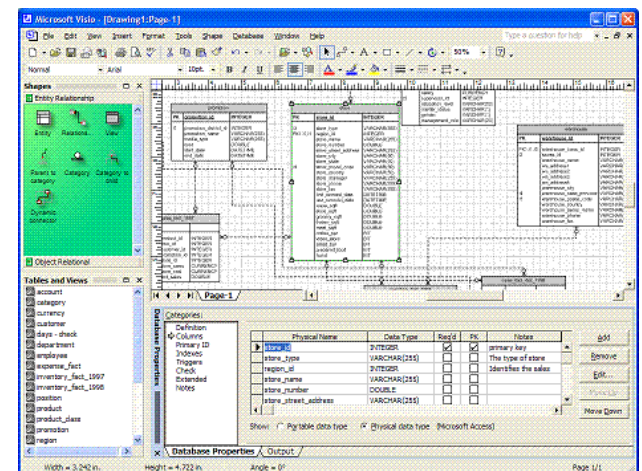
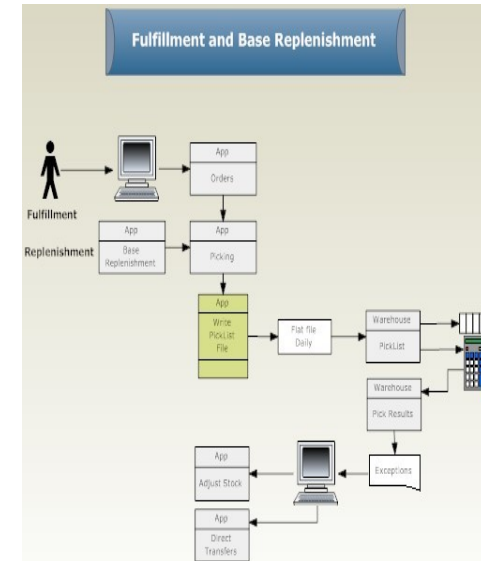
Considerations for a Requirements Traceability Database

- ■ ■ ■ A collaborative environment for the whole team
- ■ ■ ■ The ability to manage changing requirements
- ■ ■ ■ Multi-dimensional traceability
- ■ ■ ■ Scalability from small to large projects
- ■ ■ ■ Integration with design, development and testing tools
- ■ ■ ■ Examples :
 - ■ ■ ■ Spreadsheet, DOORS , RequisitePro

Requirements:	Priority	Compliance	WC Referenc	Assignee	Prime responsibility
RFT1: The Solution must support component reuse.	Mandatory	Yes	C.4.2.1.2	David Hanslip	No
RFT2: The Tenderer must advise, for each Component of...			C.4.2.1.6	David Hanslip	No
RFT3: The Solution should be able to integrate to the...	Highly Desirab	Yes	C.4.3.5.2	David Hanslip	No
RFT4: The Tenderer must advise any reporting that is...		Yes		David Hanslip	No
RFT5: The Tenderer must advise all tools from all...			C.4.4.1.1	David Oakley	Yes
RFT6: The SDE must support the following tasks...	Mandatory	Yes	C.4.4.1.2	David Hanslip	No
RFT7: The SDE must support the roles specified in the Base...	Mandatory	Yes	C.4.4.1.3	David Hanslip	No
RFT8: The SDE must support development of applications...	Mandatory	Yes	C.4.4.1.4	David Hanslip	No
RFT9: The SDE should be highly integrated across the...	Highly Desirab	Yes	C.4.4.1.5	David Hanslip	Yes
RFT10: The SDE should integrate to the systems delivery...	Mandatory	Yes	C.4.4.1.6	David Hanslip	No
RFT11: The SDE must support multi-team, multi project...		Yes	C.4.4.1.7	David Hanslip	No
RFT12: Advise any issues the Principal may have with team...	Mandatory		C.4.4.1.7	David Hanslip	No
RFT13: The Tenderer must advise the facilities provided for...			C.4.4.1.8	David Hanslip	No
RFT13.1: Diagram types supported	Mandatory		C.4.4.1.8	David Hanslip	No

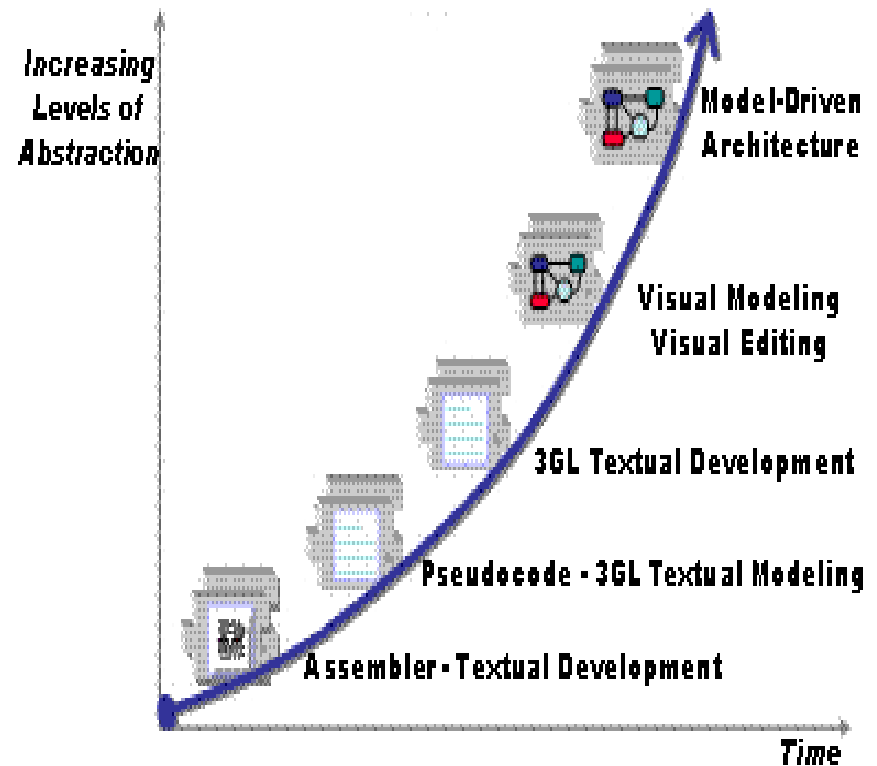
Considerations for a Drawing Tool

- ■ ■ ■ A collaborative environment for the whole team
- ■ ■ ■ The ability to draw both functional and operational designs
- ■ ■ ■ Draw diagrams with meaning - UML, ADS etc...
- ■ ■ ■ Easily incorporate drawings into documentation
- ■ ■ ■ Scalability from small to large projects
- ■ ■ ■ Integration with requirements, development and testing tools
- ■ ■ ■ Examples:
 - ■ ■ ■ Visio, SmartDraw, Rational Software Architect



Model-driven architecture (MDA) and development (MDD)

- MDA provides an approach for, and enables tools to be provided for:
 - Specifying a system independently of the platform that supports it.
 - Specifying platforms.
 - Choosing a particular platform for the system being developed.
 - Transforming the system specification into one for a particular platform.



Patterns

Resisting the urge to start from scratch



Why are Patterns important?

- ❑ A Pattern is a *reusable* generalization (or abstraction) that can be used as the starting point in future solutions.


- ❑ The benefits of Patterns are that they:
 - ❑ Provide a mechanism to capture knowledge and experience
 - ❑ Provide a common vocabulary among architects and designers
 - ❑ Facilitate reuse of approaches that have been successful elsewhere; thus, contributing towards the following aspects of a project by:
 - ❑ Reducing risk
 - ❑ Increasing quality
 - ❑ Improving delivery time

“One thing expert designers know not to do is to solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past. When they find a good solution, they use it again and again. Such experience is part of what makes them experts.”

Design Patterns, Gamma, Helm, Johnson & Vlissides 1995

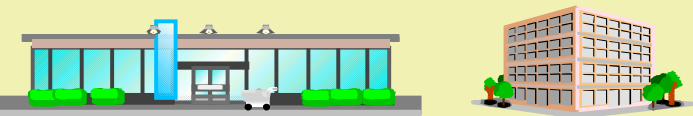
There are several main types of Patterns (from big to small).

Focus



Reference Architectures

Process-related: - CRM, SCM - Online Buying	Technical: - e-business - Wireless
--	---



Patterns for e-business

Used in high-level workshops (pre-contract) and in early stages of the project

Architectural Patterns

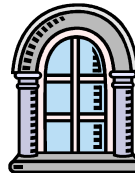
Used by architects and specialists (in early stages of the project)

- Layers
- Pipes and Filters

Design Patterns

Targeted at analysis and design

- Abstract Factory, Proxy, Facade



Analysis Patterns

Targeted at object modeling and database design

- Party, Organization, and Account

IBM Patterns for e-business

- ❑ **IBM Patterns for e-business** are a layered set of patterns to underpin an IT architecture
 - ❑ These patterns help you understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented using lower-level patterns.
- ❑ **Patterns e-business define the following assets:**
 - ❑ **Business patterns:** establish the primary business purpose of a solution; they identify the interaction between users, businesses, and data
 - ❑ **Integration patterns:** tie multiple Business patterns together
 - ❑ **Composite patterns:** represent commonly occurring combinations of Business patterns and Integration patterns (such as Electronic commerce and Portal)
 - ❑ **Application patterns:** refine Business patterns so that they can be implemented as computer-based solutions (in the form of application components and data)
 - ❑ **Runtime patterns:** define the logical middleware structure supporting an Application pattern (in form of middleware nodes and interfaces)
 - ❑ **Product mappings:** identify proven and tested software implementations for each Runtime pattern
 - ❑ **Best-practice guidelines:** document benefits and pitfalls during design, development, deployment, and management

How are Patterns for e-business used?

Seven Application Patterns for Business pattern "Self-Service"

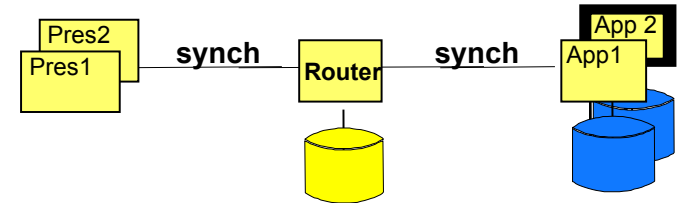
Blue Service

- Stand Alone Web-Up
- Directly Integrated Web-Up
- As-is Host
- Customized Presentation
- Router
- Decomposition
- Agent

Business Drivers	Stand Alone Web-Up	Directly Integrated Web-Up	As-is Host	Customized Presentation	Router	Decomposition	Agent
Time to market	✓	✓	✓	✓	✓	✓	✓
Improve the organizational structure	✓	✓	✓	✓	✓	✓	✓
Reduce the latency of business events	✓	✓	✓	✓	✓	✓	✓
Easy to adopt during Merger & Acquisitions	✓	✓	✓	✓	✓	✓	✓
Integration across multiple legacy databases and servers	✓	✓	✓	✓	✓	✓	✓
Global footprint and various Lines of Business (LOB)	✓	✓	✓	✓	✓	✓	✓
Simple reference and common code setting	✓	✓	✓	✓	✓	✓	✓
Mean capitalization	✓	✓	✓	✓	✓	✓	✓
IT Drivers	Stand Alone Web-Up	Directly Integrated Web-Up	As-is Host	Customized Presentation	Router	Decomposition	Agent
Minimize operation complexity	✓	✓	✓	✓	✓	✓	✓
Minimize total cost of ownership (TCO)	✓	✓	✓	✓	✓	✓	✓
Leverage existing skills	✓	✓	✓	✓	✓	✓	✓
Leverage legacy investment	✓	✓	✓	✓	✓	✓	✓
Reduce total application program	✓	✓	✓	✓	✓	✓	✓
Minimize hardware consumption	✓	✓	✓	✓	✓	✓	✓
Maintainability	✓	✓	✓	✓	✓	✓	✓
Scalability	✓	✓	✓	✓	✓	✓	✓

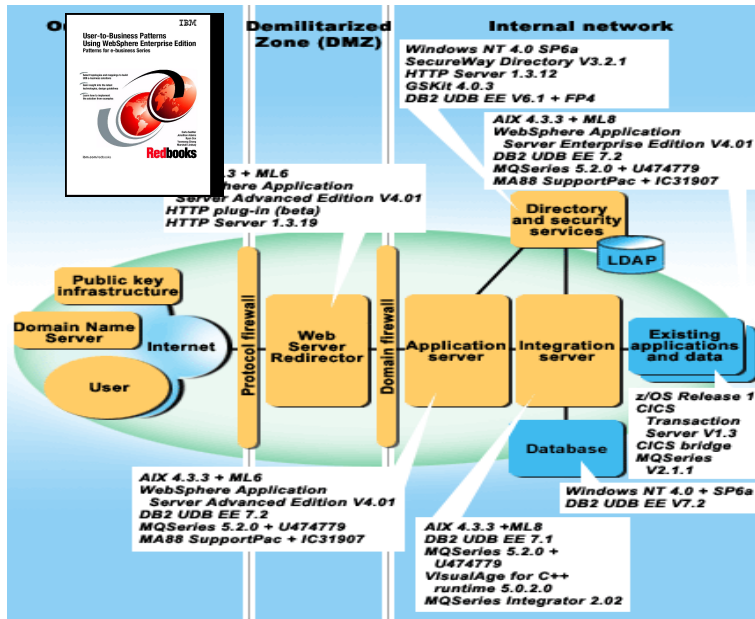
Select Application Pattern based on Business and IT Drivers

Application Pattern "Self-Service::Router"



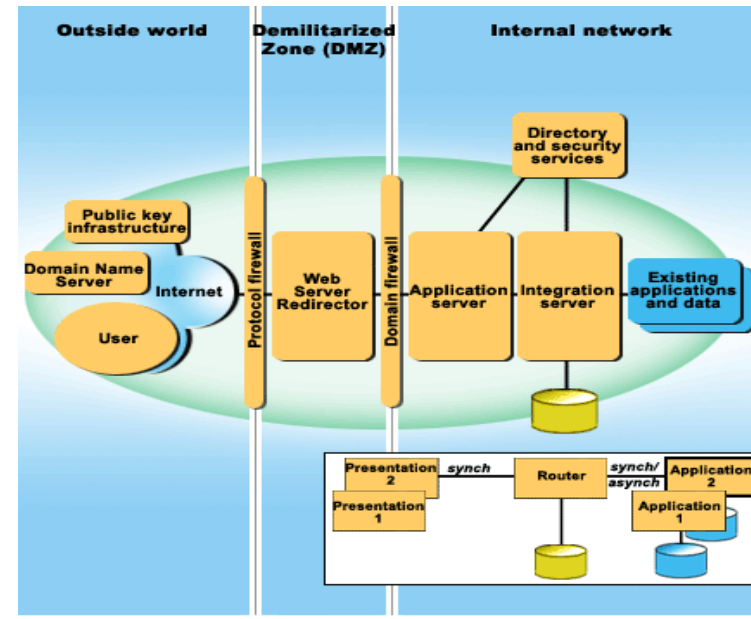
Select Runtime pattern based on NFRs

Product Mapping for "Self-Service::Router"



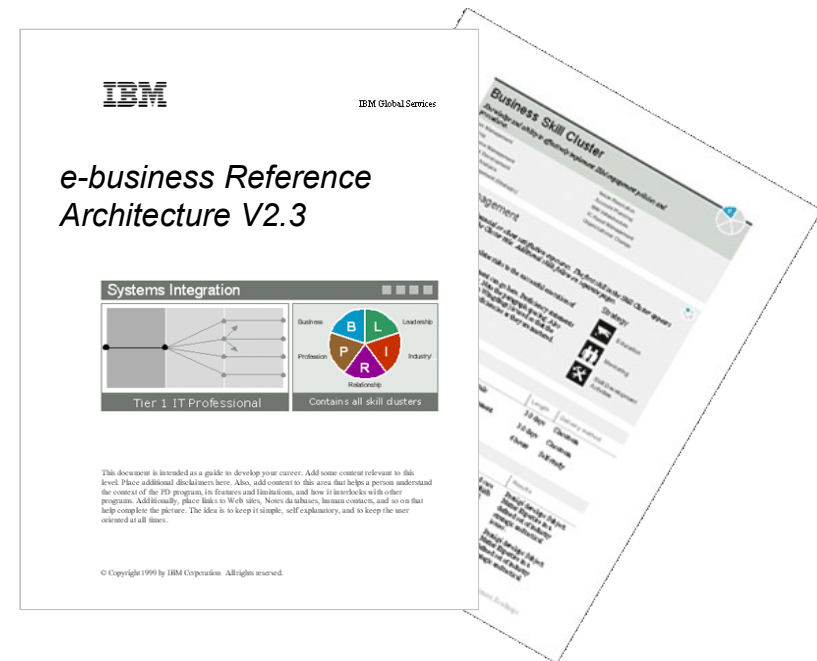
Select Product Mapping based on product decisions

Runtime Pattern "Self-Service::Router"



Introducing the e-business Reference Architecture

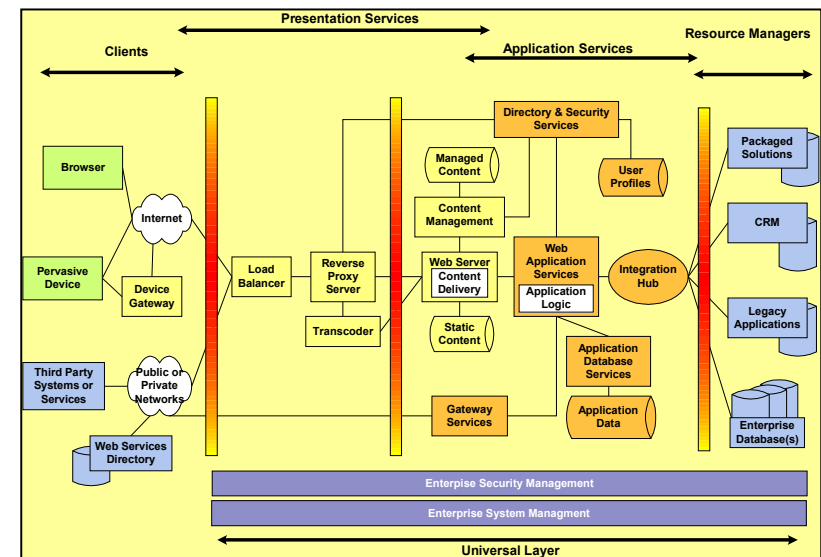
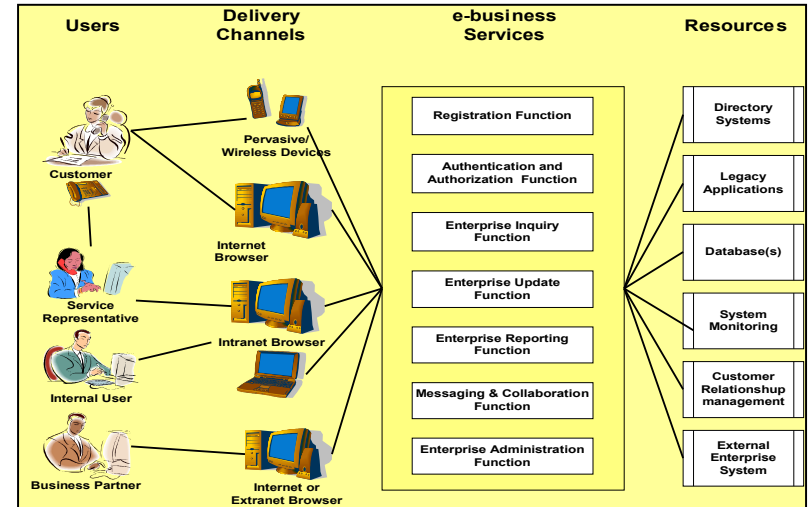
- ❖ The e-business Reference Architecture (ebRA) helps define the end-to-end architecture of the solution.
- ❖ Identifies the key architectural work products and defines these work products at the Conceptual and Specification levels of elaboration.
- ❖ Is typically used during Solution Outline and Macro Design stages of a project.



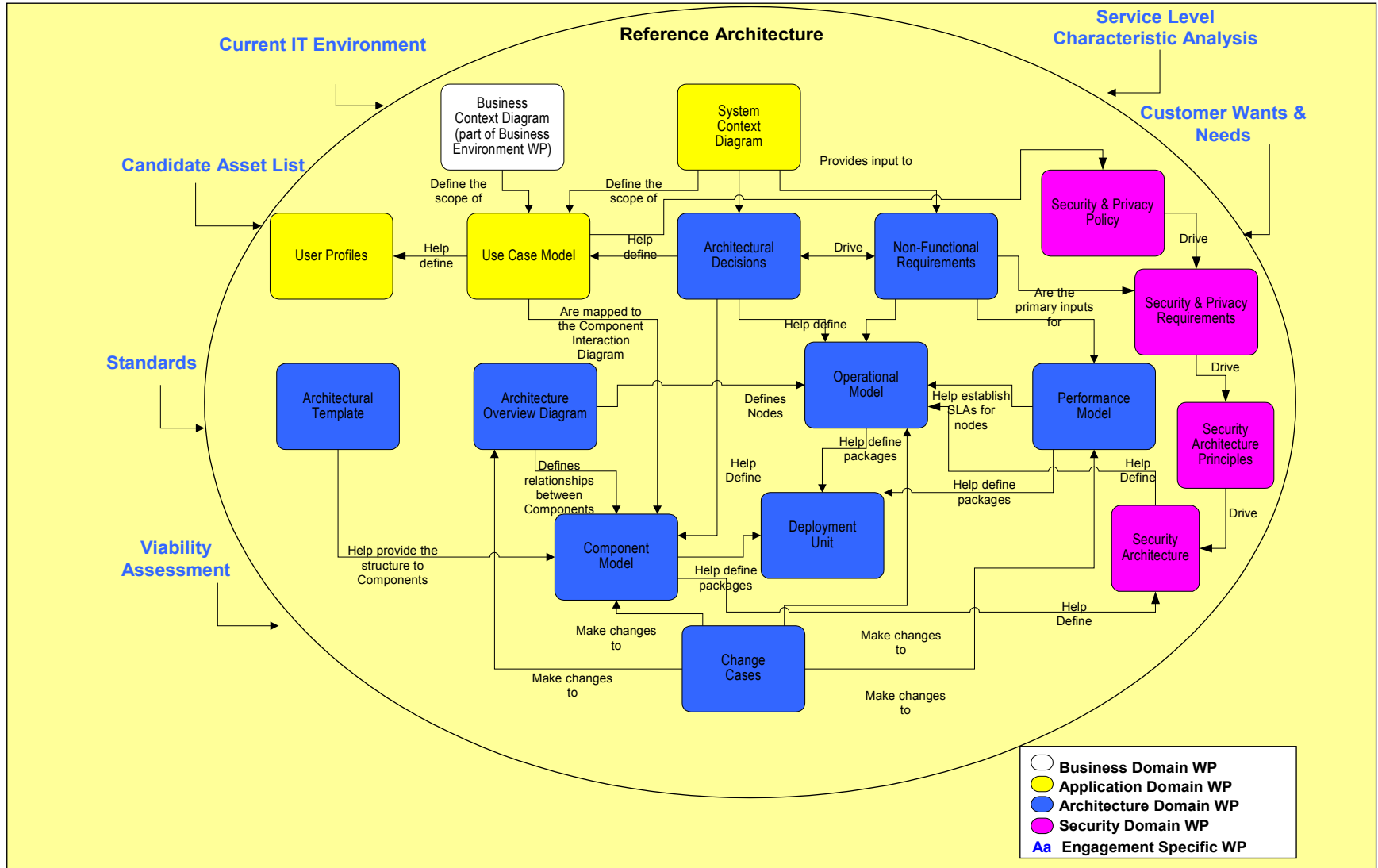
- ❖ The ebRA was developed by IBM as part of a Reference Architecture project called "Enterprise Solution Structure"

What's in the e-business Reference Architecture

- Contains 16 IBM Global Services Method based documents that are:
 - IS O 9000 Compliant
 - created using Microsoft Word, Microsoft PowerPoint and Microsoft Visio
- Includes a physical elaboration of an operational model using IBM Products
- Includes a technique paper: "Using the e-business Reference Architecture" and a glossary of terms



Work Products in the e-business Reference Architecture



Summary



Learning Points



- ❑❑❑ Remember the three ‘Cs’
 - ❑❑❑ Context, Common Sense and Communication
- ❑❑❑ Remember there are different ways of producing an IT Architecture and learn when to use the appropriate technique.
- ❑❑❑ Use methods, modelling languages and design tools to enable re-use of IT Architecture assets.
- ❑❑❑ Patterns are important. Don’t assume that your problem is “first-of-a-kind”. Use patterns to break down the complexity of your problem.