

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



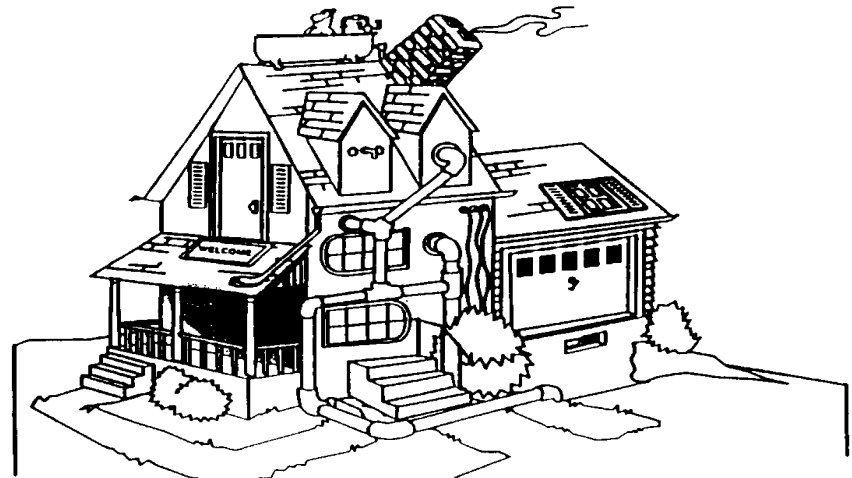
The three “Cs”

Context

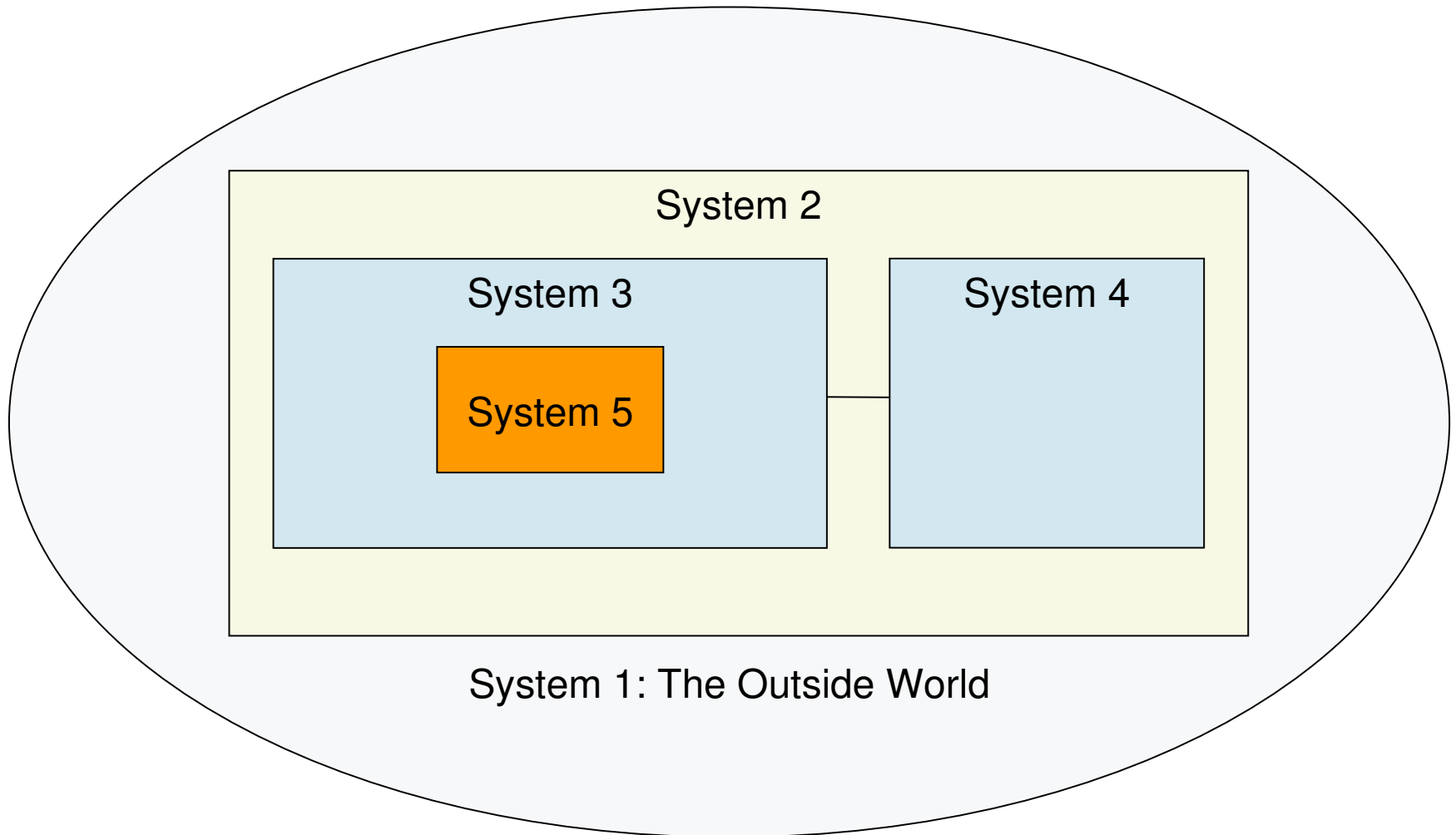


The fundamental skill of an IT Architect is to understand the context they are working in

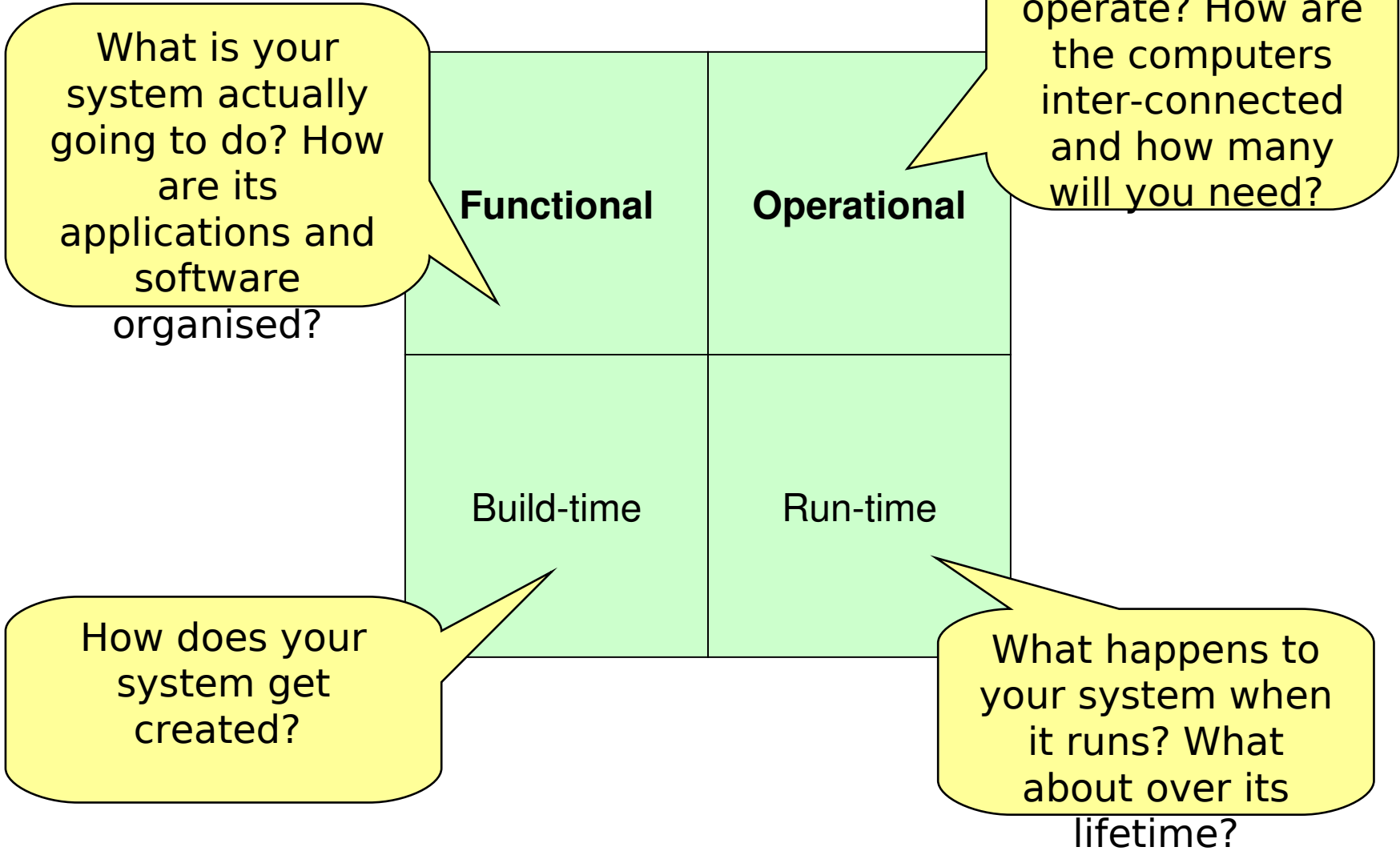
- ❑ Putting IT architecture in context itself – why do we need it?
- ❑ We will cover:
 - ❑ System context and boundaries
 - ❑ Architecture context – aspects of architecture
 - ❑ Project context



Finding the system boundaries



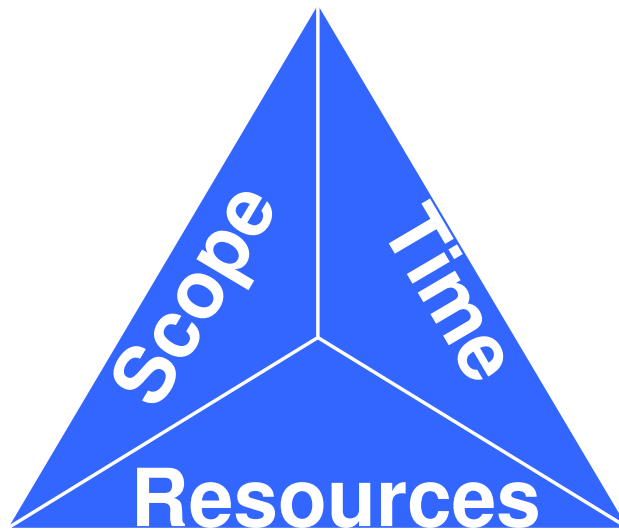
Aspects of IT architecture



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?

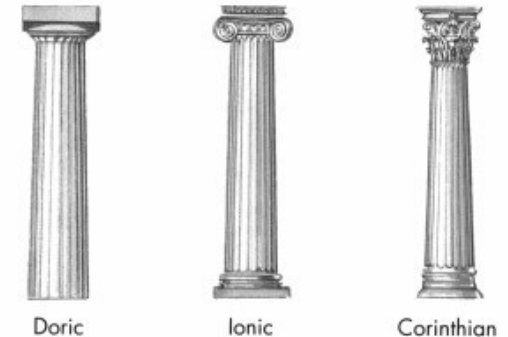
The three “Cs”

Common Sense



The Process of Architecting

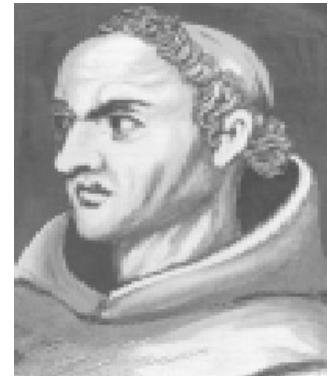
- ❑ The Normative way
 - ❑ Schools of architecture
 - ❑ Innovation suppression
 - ❑ Dogma
- ❑ The Rational way
 - ❑ Procedure driven
 - ❑ Logical
- ❑ The Argumentative way
 - ❑ Mechanistic
 - ❑ Brainstorming
 - ❑ Too many cooks
- ❑ The Heuristic way...



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

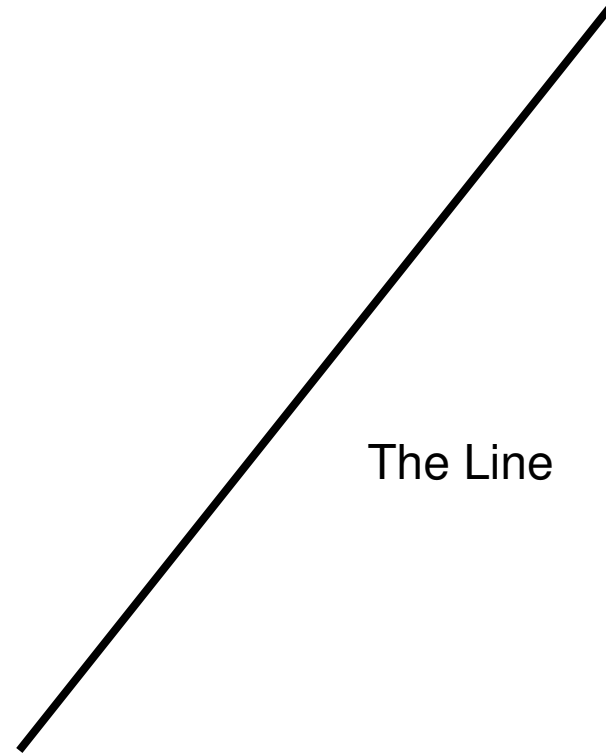
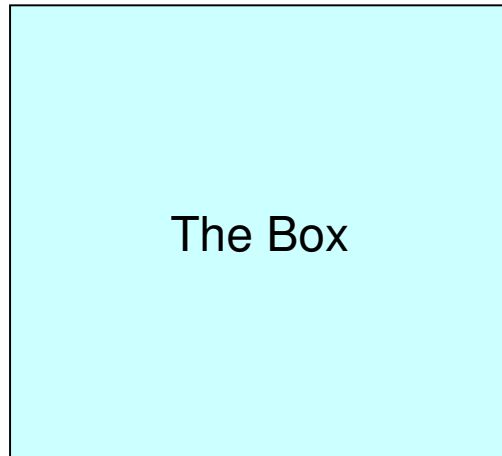


Functional	Operational
Build-time	Run-time

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.

Communication skills are important for a successful IT Architect

Spoken Communication

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

Written Communication

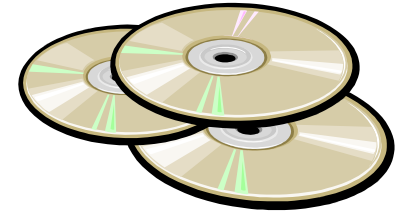
- Written Communication
 - An undocumented architecture is difficult to build and maintain; it leads to a lot of repeat discussions.
 - Architectural documentation can be a time-saver, not a time-waster.
 - Documentation is extracted from work products, but focuses on key messages and stakeholder needs

The kitbag

What every IT architect has
on their laptop



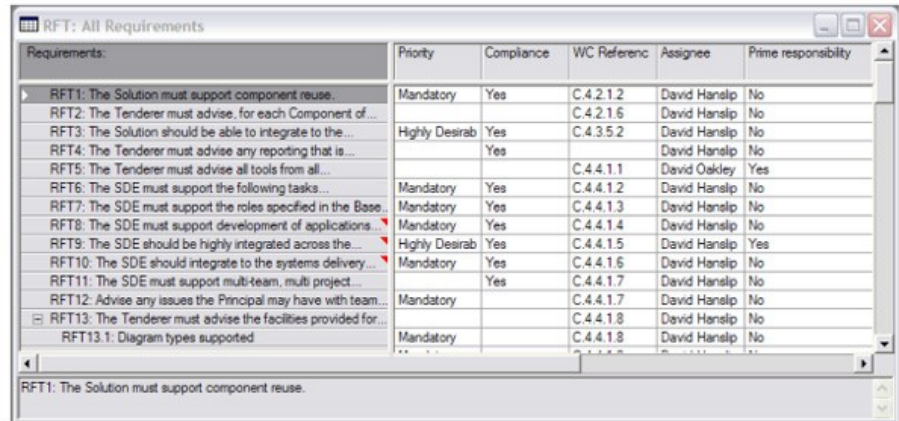
The real tools



- ⌘ 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

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

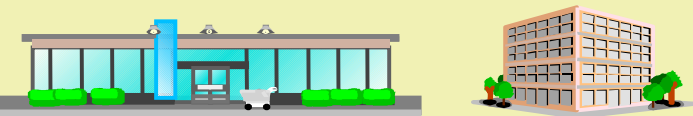
There are several main types of Patterns (from big to small).

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

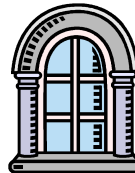
Used by architects and specialists (in early stages of the project)

- Layers
- Pipes and Filters

Design Patterns

Targeted at analysis and design

- Abstract Factory, Proxy, Facade



Analysis Patterns

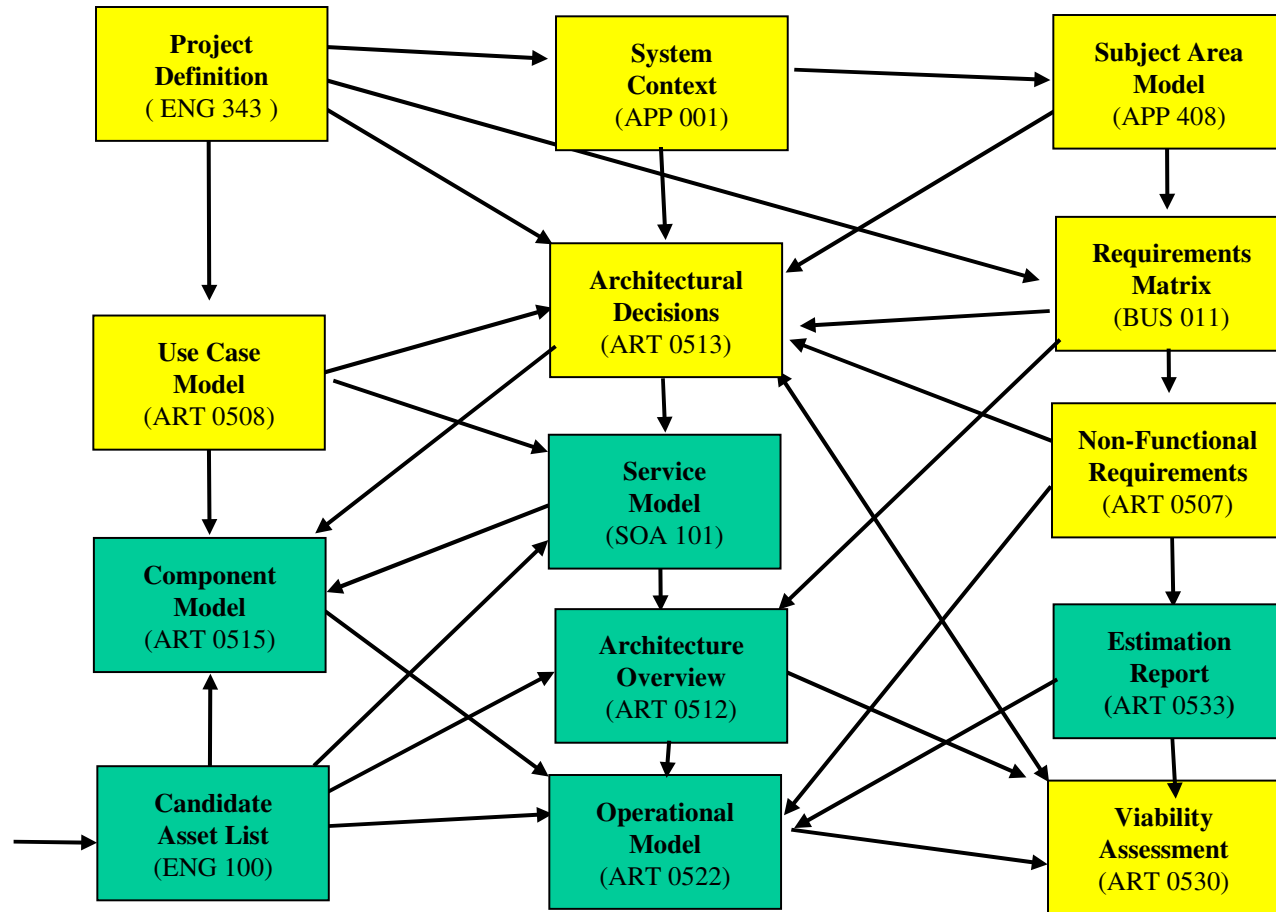
Targeted at object modeling and database design

- Party, Organization, and Account

Summary



Work Product Dependency Diagram



Assets
 Reference Architectures
 ITT Solution Brief, Patterns
 Redbooks, etc

Learning Points

- ❑ Remember the three 'Cs'
 - ❑ Context, Common Sense and Comm
- ❑ 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.

