



What's the Problem?

## Objectives

tecademy

- To explore the initial steps of the project lifecycle.

## Contents

tecademy

- The Role of the Business Analyst
- Problem Statement
- Operational Concept Document
- Glossary
- Stakeholders
- Business Context Diagram

## The Business Analyst

### ▶ The role of the Business Analyst (BA) is to

- ▶ understand the problem
- ▶ understand the requirements
- ▶ communicate effectively this understanding to others

### ▶ The BA's objective is to

- ▶ produce a complete, correct and consistent set of requirements...
  - ▶ represented by the use case model
  - ▶ summarised by the use case diagram
- ▶ ... supported by a model of the problem domain
  - ▶ represented by the class model
  - ▶ summarised by the class diagram

# The Problem Statement

## ▶ Problem Statement

- ▶ Describes the problem, not the solution
- ▶ As clear and concise as possible
- ▶ Produced by problem owner

## ▶ Operational Concept Document

- ▶ Provides a vision of the world where the problem has been solved
- ▶ Describes how the solution will look
- ▶ Describes how the solution might fit within the business environment
- ▶ Produced by 'Visionary'

People who are involved in software development are in the business of solving business problems for their customers. Before we can do our job as solutions providers, we must first know what the problem is. This might sound obvious, but many projects lack a clear definition of the problem and this makes it difficult to establish any meaningful criteria by which to judge the quality of the solution. A *Problem Statement* is therefore the first document to be produced, preferably by those who have the problem.

The Problem Statement should be short – no longer than one side of A4, otherwise people will be disinclined to read it. So, the shorter the better. It should be written in the language of the business and should avoid any discussion of a solution – just a clear, concise description of *what the problem is*. Contextual information and illustrations of the problem can be included to aid understanding.

Any information related to the solution – e.g. what machine it will run on, in what software environment, using which package – must be avoided in this document. Put it in another product – it's useful information, but it's not part of the problem – it describes the context in which the solution will be provided. An *Operational Concept Statement* can be compiled to describe how the solution might look. It should be produced by a 'Visionary' - someone who has projected themselves into a world where the solution exists – who can describe how the solution might fit and be used within the context of the business environment.

## The Glossary

- **glossary** [*glOssāri*] *n* 1. an alphabetical list of words peculiar to a subject area, or **domain**. 2. An excellent way of allowing everyone to share the same understanding of a word. 3. The final arbiter in any argument about what a word means.
  
- **glossary entry** [*glOssāri Ēntri*] *comp n* consists of the word, [its pronunciation], part of speech (e.g. *noun*, *verb*, *adjective*), precise definition and an example of usage. Multiple definitions are listed in descending order of usage. Words used in the definition appear in bold face if they appear elsewhere in the **glossary**. The entry ends with a list of synonyms, if any.

It is important that those entrusted with the job of solving problems inspire confidence in their customers. Customers want to feel that the solution providers *understand* the problem. The only effective way of convincing the customers that their problem is understood is by being able to explain to the customers *in their own language* what the problem is. So we must set about learning that language – and, as anyone who has studied a foreign language will know, one of the best ways of starting is to write down each new word we learn, with its meaning, in a *glossary*.

Glossaries are important. Many discussions (and arguments!) occur because of differing views of what a word means. The glossary ends the debate. Entries should be compiled and/or verified by domain experts, maintained regularly and made easily accessible to all.

Some glossary entries will point to the existence of important domain objects we will need to include in our models – these should be noted and compiled into a list of *candidate classes*, of which more later.

## Case Study - Exercise 1

- **Objective**

- To introduce the Case Study.

- **Turn to and complete exercise 1 in the exercise booklet.**

## Identify Stakeholders tecademy

▶ **Stakeholder** [*staykholder*] *n* anyone with an interest in the solution.  
Could include:

- ▶ **Users**
- ▶ **Architects**
- ▶ **Developers**
- ▶ **Managers**
- ▶ **Auditors**
- ▶ **Testers**
- ▶ **Quality Assurance**
- ▶ **Human Resources**
- ▶ **Trainers**
- ▶ **....**

©tecademy 8

Also important is that everybody with an interest in the problem and its solution be identified and involved at the outset. The list of Stakeholders is not confined to end-users and developers. Architects, Auditors, Quality Assurance are amongst those who need to be consulted to ensure the parameters within which the solution is to be constructed are understood fully. HR might be needed to make sure the project is adequately resourced with the right skills, and so on.

Joint Application Development (JAD), or some other formalised sessions can be used to bring these interests together in order to discuss the scale and scope of the project within the overall context of the business. JAD sessions are focused on driving out functional and non-functional requirements within a structured brainstorming environment led by an experienced facilitator. These sessions can be repeated for lower-level requirements as necessary by different groups concentrating on discrete parts of the problem.

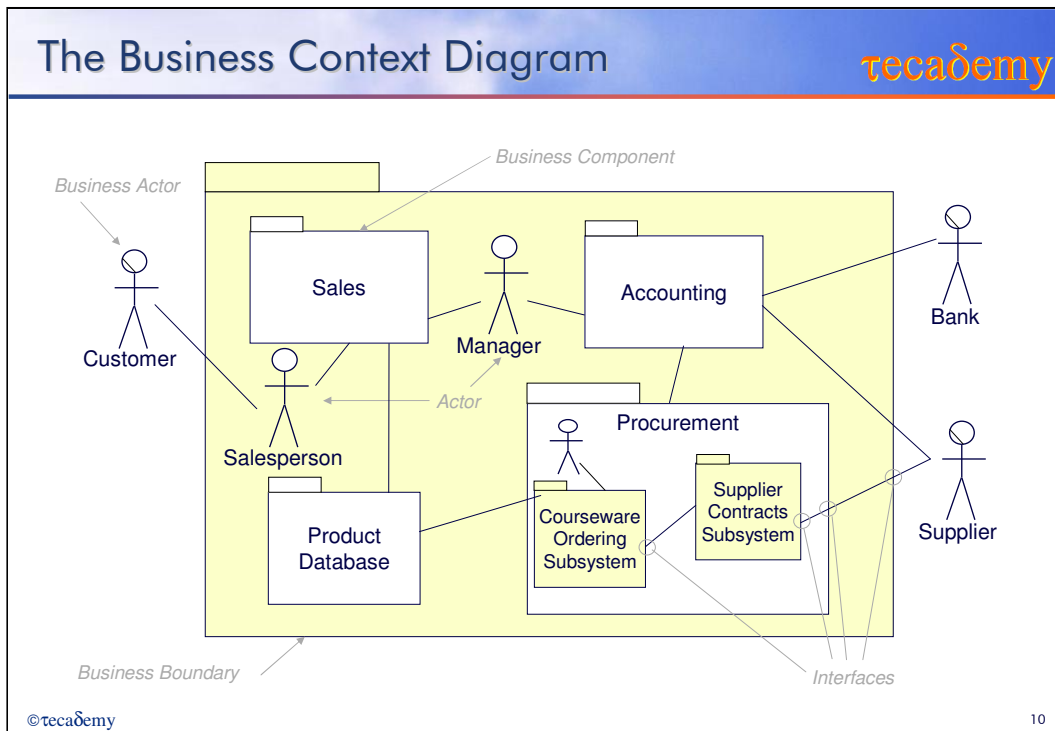
JAD session deliverables include the *Business Context Diagram*, *Use Case Diagrams* and *Use Case Descriptions*.

## The Business Context Diagram tecademy

- Shows the main architectural components of the solution within the context of the business and/or the outside world
- Allows identification of interfaces
  - between solution components,
  - between solution and the outside world
- Solution components can be
  - computer systems
  - job roles
  - departments

©tecademy 9

The Business Context Diagram is developed in consultation with the Stakeholders. It shows the main architectural components of the solution within the immediate context of the business and the world outside, together with the interfaces between them. Solution components can include Job Roles, Departments within the organisation, or computer systems. An example follows.


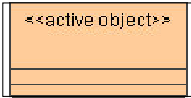

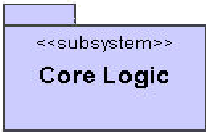




The Business Context Diagram is a useful device for exploring the position of the problem (and/or solution) and its relationships with other parts of the business, and that of the business with those external entities, or *Business Actors*, it deals with. Job roles within the business are modelled as *Actors*.

Lines represent interaction or dependency. The points where these lines cut across business or system boundaries represent *Interfaces* which can be investigated further and, if appropriate, modelled as prototypes.

Architectural Tool set
tecademy

**Commonly used UML tool set for architectural models**

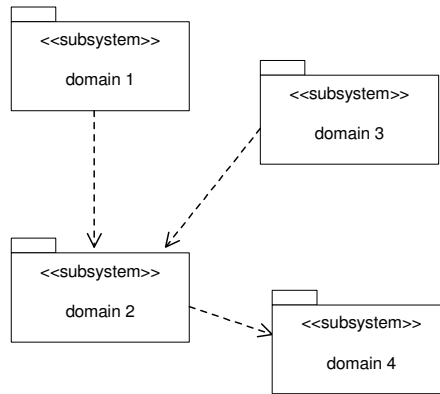
 <p><b>Data types</b></p> <p><b>Package</b></p> <p>Logical grouping of model elements</p>	 <p><b>Active Object</b></p> <p>A cluster of objects that function as one</p>	 <p><b>Component</b></p> <p>Physical grouping of model elements</p>
 <p><b>Core Logic</b></p> <p><b>Subsystem</b></p> <p>Organised set of run-time components or technology groupings</p>	 <p><b>Accounts</b></p> <p><b>Domain</b></p> <p>An independent subject matter that has its own set of concepts</p>	 <p><b>Rollover Server</b></p> <p><b>Node</b></p> <p>Physical unit of hardware</p>

©tecademy
11

Object Oriented Software Development is commonly termed an *Architecture Driven Approach* to software development. This is because the process of developing a business model should also drive the process of developing a full enterprise architecture model, encompassing the business, data, application and technical domains. Good models tend to have the same shape and structure of the real-world thing they represent. A business is the sum of its processes. It should, therefore, be possible to map all architectural components to the business processes they support. For that reason, the business model does, to some extent, decide the shape of the overall architecture.

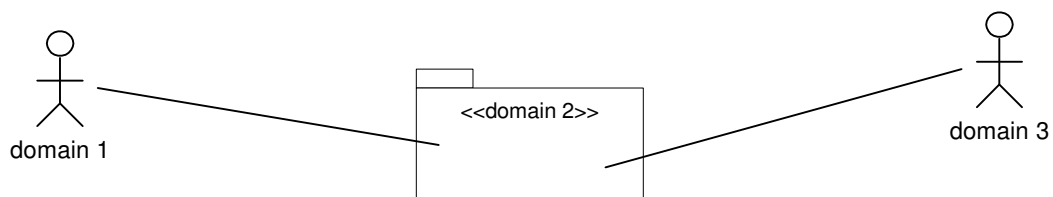
## Subsystem analysis

- A logical architectural view should be constructed first!
- Commonly known as a domain analysis view



During the initial stages of analysis, a team might begin by trying to analyse a problem in a single go. This usually fails, leading to much frustration. It is more productive (and makes the whole process more achievable) if the problem is broken down into smaller parts, each relating to a particular area within the the problem domain space. This process of breaking a large problem into smaller problem areas and then analysing each area separately is known as *domain partitioning* and *domain analysis*.

Each domain will become an actor to another domain.



## Summary

🎯 **The solution provider must understand:**

- the Problem
- The language
- how solution fits within business
- who the Stakeholders are

🎯 **Important documents are:**

- Problem Statement
- Operational Concept Document (the Vision)
- Glossary
- List of Stakeholders
- Business Context Diagram

## Case Study – Exercise 2

### Objective

- To build a Business Context Diagram

Turn to and complete exercise 2 in the exercise booklet.

Don't Forget – NO assumptions! If in doubt - ASK!