جامعة الملك سعود
King Saud University
College of Computer & Information Sciences

Domain Analysis

Chapter 4 (4.1, 4.2,4.3)

Object-Oriented Software Engineering Practical Software Development using UML and Java by Timothy Lethbridge, Robert Laganiere

Note: all quotes are from this reference

Fall Semester 2021
1st Semester 1443 H

What is domain analysis?

Domain analysis is

"the process by which a software engineer learns background information. He or she has to learn sufficient information so as to be able to understand the problem and make good decisions during requirements analysis and other stages of the software engineering process."

What is domain analysis?

The word ‘domain’ means the “general field of business or technology in which the customers expect to be using the software”.

Examples of domains: airline reservations, medical diagnosis, financial analysis and scheduling meetings.

People who work in a domain and who have a deep knowledge of it (or part of it) are called *domain experts*.

Benefits of domain analysis

As a software engineer, you are not expected to become an expert in the domain; nevertheless, domain analysis can involve considerable work. The following benefits will make this work worthwhile:

- Faster development.
- Better system.
- Anticipation of extensions.

Contents of the domain analysis document

- Introduction
- Glossary
- General knowledge about the domain
- Customers and Users
- The environment
- Tasks and procedures currently performed
- Competing software

Activity

Outline the information you would need to gather in order to perform domain analysis for an ***airline reservation system.***

Starting point for software projects

	Requirements must be determined	Clients have produced requirements
New development	A	B
Evolution of existing system	C	D

Defining the problem and the scope

Once you have learned enough about the domain, you can begin to determine the requirements. The first step in this process is to work out an initial definition of the *problem* to be solved.

A problem can be expressed as a *difficulty* the users or customers are facing, or as an *opportunity* that will result in some benefit such as improved productivity or sales.

The solution to the problem will normally entail developing software, although you may decide that it is better to purchase software or to develop a non-software solution.

A good problem statement

A good problem statement is short and succinct – one or two sentences is best.

For example, if you were developing a new student registration system, you might express the problem as follows:

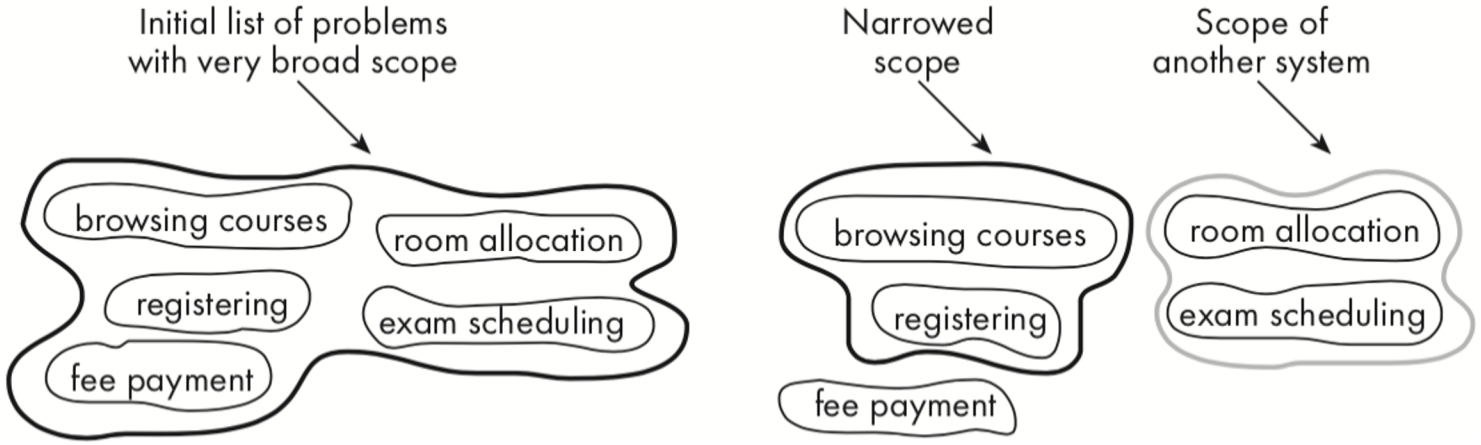
> "The system will allow students to register for courses, and change their registration, as simply and rapidly as possible. It will help students achieve their personal goals of obtaining their degree in the shortest reasonable time while taking courses that they find most interesting and fulfilling."

Problem Scope

- If the problem is broad, or contains a long list of sub-problems, then the system will have a broad scope, and hence be more complex. An important objective is to narrow the scope by defining a more precise problem.
- In the previous example, if we had stated: ‘the system will automate all the functions of the registrar’s office’, that leaves open the possibility of including such features as fee payment, printing class lists and allocating rooms to courses.

How to set the scope of the project

One way to set the scope is to list all the sub-problems you might imagine the system attacking. To narrow the scope, you can then exclude some of these sub-problems – perhaps they can be left for another project.



Narrowing the project's scope

Thank you!