

Computer Science Project

3COM0063

Also Specialist Modules 3COM0065, 3COM0066, 3COM0067, 3COM0068,
3COM0069, 3COM0070, 3COM0080, 3COM0120

Lecture 2

AIMS

This session aims to get you to think more about:

- What makes a good project
- What makes a good set of objectives
- How to extend a basic idea

Slides available on-line

The slides from this lecture are at:

http://homepages.feis.herts.ac.uk/~cs4_proj/

Choose Page Scaling: Multiple pages per sheet

Recap

We told you earlier:

- Project as a series of increasingly challenging objectives
- Process and Assessment
- Don't look for an easy project
- Need to Reflect

Choosing a Project

- Free choice of topic and the issues you set out to address (within reason)
- Suggestions available (but limited spaces)
- Even if you choose a project from our list, you define the advanced objectives for your project and how you are going to meet them
- So you set the objectives for your project, which is DANGEROUS

So what makes a good project?

- practical problem to solve / issue to address
- realistic idea for solving the problem /
addressing the issue
- understanding when to stop
- a set of milestones for judging progress
- a plan for presenting the results
- understanding of how to evaluate your work
- motivation to carry the project through
- ability to achieve the goals you have set

Success, mediocrity and failure

- good practical work + good written work
▶ Good grade
- good practical work + poor written work
▶ Can't tell if it is good
- poor practical work + good written work
▶ Nothing to write about
- poor practical work + poor written work
▶ Failure :-(

We expect you to

- set realistic objectives
- work independently
- foresee potential problems
- react to changing circumstances
- present results coherently
- recognise and overcome difficulties
- identify essentials
- evaluate your own work

Things that can go wrong

Being: vague,
over-ambitious,
under-ambitious,
distracted, ...

Not working hard enough

Vagueness

- being vague about what you are going to do
- failing to properly define the problem you intend to address
- failing to decide precisely what you will produce

Not Planning

- not planning your practical work before you do it
- not deciding what you will put in your report until it's time to write it
- not thinking about how you will judge the quality of your work

Scope

- Setting the scope of the project
 - too wide (over-ambitious)
 - too narrow (under-ambitious)
- Concentrating on learning tools and techniques rather than solving a problem or addressing an issue
- Losing motivation

Not Working

- Not doing any work
about 300 hours productive work for an average student to get an average grade

only about 7 hours of tutorials
- Concentrating on the quantity of time you are spending rather than the quality of the work you are doing

Aims, Project Objectives, System Requirements and Personal Goals

- An aim is a general, overall thing you would like to accomplish.
- An objective is specific, concrete and testable.
- A personal goal is something *you* want to learn or to achieve. Personal goals do not usually make sensible project objectives.
- System requirements are not project objectives.

Objectives

Your project objectives should indicate

- how much work you intend to do;
- what kind of work it will be;
- what standard you expect to achieve;
- what you intend to produce.

How do I develop my initial idea?

Your topic needs to be turned into a sequence of concrete and achievable objectives.

Even if you are not taking one of the Project Suggestions, look at the way they are formulated to see how a project is broken down into stages and objectives are set out.

Setting objectives

Your objectives should be SMART:

Simple

Meaningful

Achievable

Relevant

Testable

The 'SMART' test

SMART stands for Specific, Measurable,
Achievable, Realistic and Timely

or

Specific, Measurable, Agreed, Realistic and
Time-bounded

or

Simple, Meaningful, Achievable, Relevant, and
Testable

[At level 3 you need to be able to cope with conflicting information, evaluate alternatives]

Simple: Each objective identifies a single, simply stated, goal, even though the project as a whole will not be simple.

Meaningful: Objectives should be written in language that others find easy to understand, using technical terminology only where absolutely necessary, avoiding jargon, slang and acronyms.

Achievable: It should be possible for you to achieve the goal in the time available and with the resources at your disposal. But remember that an investigative project can set up a hypothesis to test; if it turns out to be false, you have still achieved an objective of testing the hypothesis.

Relevant: The goal should be directly related to your project; not be a personal goal. Thus, to "learn C++" might be necessary, but it is not an objective.

Testable: It should be possible to say whether or not it has been achieved, for example: "I will create a data dictionary".

Objectives exercise:

(source: Tom Angelo, Victoria University, Wellington, NZ)

A number of statements

Are these objectives?

Some you may not be sure about

1. Look at the statements on your sheet, and mark them "+" for **objective** or "-" for **not** (as appropriate).

If you are not sure, put "?"

2. Find 2 others and find out whether you agree. Discuss the disagreements!

3. When I bring the discussion to a close, mark any statements where the discussion caused you to change your mind!!

Where were the disagreements?

- Read a book about how to design databases (normalisation, etc).
- Go to the LRC and spend time looking for relevant papers.
- Read the paper "How java programs interact with virtual machines at the microarchitectural level".
- Download the latest Java ME developer tools from Sun, including the Wireless Toolkit.
- Install Dreamweaver, and get it talking to mySQL.
- Talk to a friend who is familiar with the requirements for systems like that being done for the project.
- Go to the LRC and make a list of relevant papers.
- Learn how to use Dreamweaver, and use it to create a "Home Page" for the site.
- To increase my knowledge of Web/WAP programming.
- System should be scalable in the event of pursuing the project in the real world.

Now turn these into 'proper' objectives

There is just about enough space on the sheet for you to write down what you think these objectives *should* say.

My Answers!

Read a book about how to design databases (normalisation, etc).

becomes

Design and implement a fully normalised database for the required system.

My Answers!

Go to the LRC and spend time looking for relevant papers.

becomes

Produce an annotated bibliography of papers about database design.

My Answers!

Read the paper "How Java programs interact with virtual machines at the micro-architectural level".

becomes

Compare the results obtained with those described in the paper "How Java programs interact with virtual machines at the micro-architectural level".

My Answers!

Download the latest Java ME developer tools from Sun, including the Wireless Toolkit.

becomes

Install the Java ME developer tools and Wireless Toolkit, and write and test a "hello world" program.

My Answers!

Install Dreamweaver, and get it talking to MySQL.

becomes

Implement a simple page to list all the products in the database.

My Answers!

Talk to a friend familiar with the requirements for systems like that being done for the project.

becomes

Create a first draft requirements catalogue.

or (much better)

Design and implement a database to support the requirements identified, and enter suitable test data to explore these requirements.

My Answers!

Go to the LRC and make a list of relevant papers.

is objective, but even better would be

Produce an annotated bibliography of papers about database design.

My Answers!

Learn how to use Dreamweaver, and use it to create a "Home Page" for the site.

is objective

My Answers!

To increase my knowledge of Web/WAP programming.

becomes

Produce a WAP front-end to the system

My Answers!

System should be scalable in the event of pursuing the project in the real world.

becomes

Measure the performance of the system as the transaction rate increases

or (not quite so good)

Document steps taken to ensure that the system is scalable.

Managing your time

- Set aside time for project work on your weekly timetable
 - use your 'project time' slots to work on your project
 - don't let other work interfere with progress on your project
- Make an action plan
 - check it regularly
 - notice when you are slipping behind
 - change it when necessary

- Conduct a short feasibility study before you start your project: do a "hello world" project ASAP.
- Start planning your final report as early as possible, and leave plenty of time for writing up your work

Putting together a Project

- Find a problem to solve
- Think about how you might solve it
- Make sure it is something you are interested in working on
- Work out a set of ‘high level’ objectives for your project

- Check that you fully understand your project objectives (try them on someone else)

- Check that your objectives are achievable
- Make sure that the resources you require will be available when you need them, or preferably **NOW**.

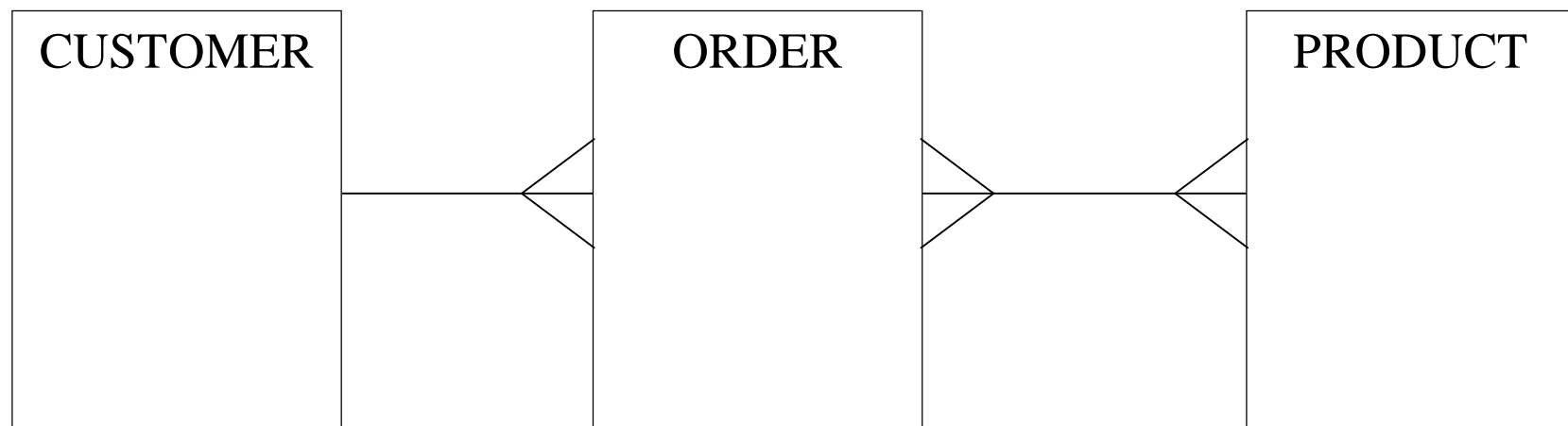
The importance of getting started NOW

- The hardest part of any project is deciding what you will do
- If you put off the decision, or keep things vague, you
 - make the decision harder
 - make starting on the practical work harder

- Students who leave it until late to decide what they are going to do tend not to do well in their projects
- Students who make up their minds early have more opportunities to revise their projects later

Extending a basic project

Suppose you can only think of a simple system...



How can you extend this?

- Extend the requirements
- Measure what you have built
- Compare alternatives
- Focus on the process

Extend the requirements

- Allow the company to nominate salesmen responsible for different customers, and empower them to offer different discounts to their customers, depending on circumstances.
- Allow price changes (without changing old billing information).
- Time limited special offers (3 for 2, BO-GOF, *etc*).
- Special offers based on purchase history.

- Allow kits of parts, with instructions, possibly for a special price; allow customers to edit their order.
- Allow leasing deals (where a customer gets a cheap lease on a piece of capital equipment in return for promising to buy supplies for it).
- Produce monthly reports comparing sales with targets for particular products/salesmen.

- Allow "call off orders" where a customer commits to buying a large quantity of a short shelf life item (to get a good price), but it is only delivered in small batches when requested.
- Stock control, supplier orders, choosing supplier depending on what mix of products is low on stock, increasing order level slightly to get a better discount.

- CRM facilities: exchanging contacts with other systems in the company (targetted, not just a database dump)
- Data warehousing: spotting trends, common purchasing patterns, *etc*
- Implement your system as a Web Service
- Interact with Google Maps

Building Community

Allow customers to talk to each other
(*not* just a chat facility)

Customer comments on product quality

Customer hints on how to use products

Marketing

targeted emails

e.g.

- Other people who bought a widget also bought one of these.
- Why don't you buy a widget cleaning kit?
- Your widget is now 6 months old and could do with cleaning (only £20!)

Measure what you have built

- Proper evaluation of the user interface and act on the results
- Measure performance of your database as number of customers goes up
- Generate realistic test data automatically, to allow large scale testing

Compare alternatives

- Build the same system with a different database
- Build the same system using XML
- Use a different tool/language to build the GUI
- Use a different approach to constructing the user interface

AND *reflect* on the differences

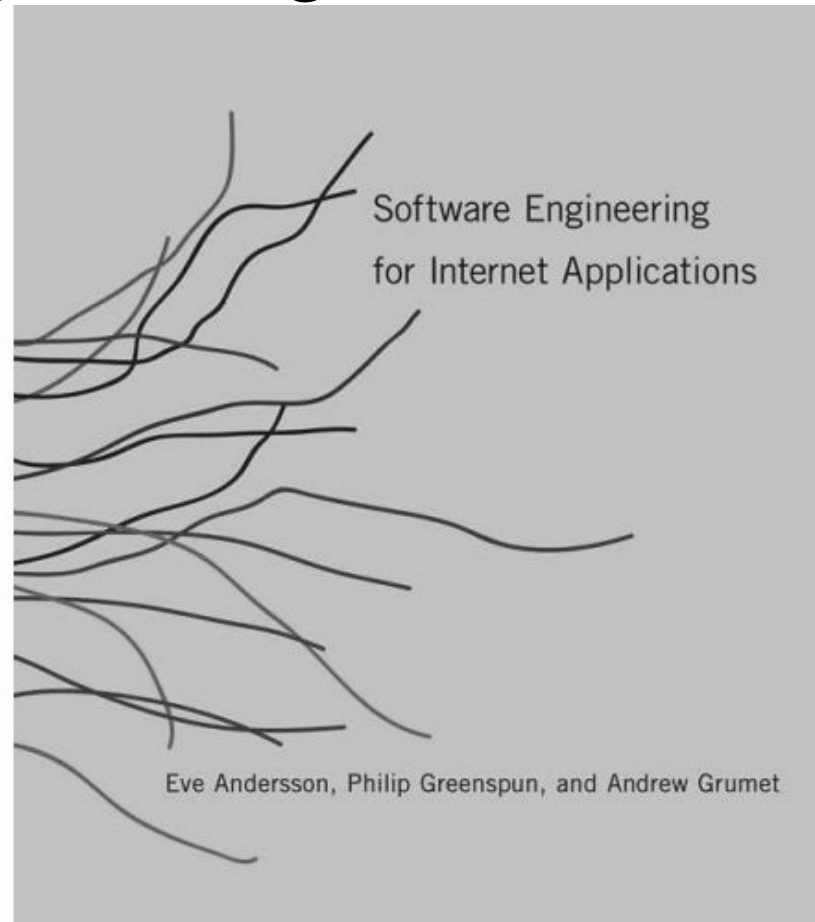
Focus on the process

- Take a scientific or software engineering approach to the task
- Document properly the design

BUT *everyone* should be doing that.

Sources of help

Software Engineering for Internet Applications



Software Engineering for Internet Applications

by Eve Andersson, Philip Greenspun, and Andrew Grumet

MIT Press 2006; ISBN 0262511916

<http://philip.greenspun.com/seia/>

LRC: 005.276 AND (14 copies)

Detailed project proposal (DPP)

First you need to prepare your Detailed Project Proposal (DPP) and get started.

Due: Friday 16th October, worth: 1%

Electronic submission only, by 23:59

(Web form)

We will tell you about that in the *next* lecture

Reminder of the process

Your DPP will specify a second and third choice of projects (which must be from the suggestions lists).

If your first choice project is over allocated, you will be allocated your second or third choice.

You will be allocated a Project Supervisor after the DPP has been handed in (expect an email) and meet weekly in term time for about 15 minutes.

If you do not submit a DPP by one week after the deadline, you may get allocated a project.

The end

Please leave the room quickly and quietly.

Wait outside if you want to ask questions;
there is another lecture in here now.

Don't forget

<http://www.studynet.herts.ac.uk/>

http://homepages.feis.herts.ac.uk/~cs4_proj

Go, read the suggestions, and choose a project!