Topic 5

Design and development

Contents

Learning Objectives

- *Appreciation of the steps involved in building systems*
- *Awareness of design principles and design types*
- *Familiarity with the System Development Life Cycle*
- *Familiarity with the Software Life Cycle*
- *Familiarity with the Programming Life Cycle*
- *Awareness of the variety of development methodologies*
- *Appreciation of the issues involved in team working*

Rome wasn't built in a day. (Proverb of French origin)

5.1 Getting the system built

You should have read Articles 2, 3 and 4 of "The Case of the Killer Robot" before starting on this topic. The list of topics which are being investigated for Assignment 2 all relate to this topic. Each topic on the list is a methodology for system analysis and design or development or a means of procuring software or a complete project management system. SSADM is a systems analysis and design tool. The development models are DSDM, Waterfall Model, Unified Process Model, Rapid Prototyping and Extreme Programming. The software sources are Reusable Software and Open Source and PRINCE is a project management system. You will be explaining these to each other in your tutorial group.

Licker (1987) identifies seven stages in the System Development Life Cycle

1. System study, problem definition
2. Preliminary investigation, feasibility study
3. Logical (functional) design
4. Physical design (blueprinting)
5. Implementation (construction)
6. Installation
7. Operation, maintenance

We have looked at the first two stages in the previous topic. In this topic we shall concentrate on the design and development stages (stages 3, 4 and 5).

There are many different sets of design principles and there is no universal agreement on which is best. We shall look at an example set of principles. We shall also look at some different approaches to design. There are also many different development methodologies and, again, much debate about which is best. You will be reporting on these to each other and once you have learnt a little about some of them you will see that they are often applicable to one or other of the design approaches or offer different benefits and carry different drawbacks.

Large tasks require teams of people. Successful teamwork doesn't just happen as a result of putting all the "best" people together. In fact, a team composed entirely of stars is likely to be plagued with problems. We shall look at why some teams work whilst others fail.

© HERIOT-WATT UNIVERSITY 2002

5.2 Design

Design is a creative exercise and so it is not easy, or indeed desirable, to prescribe a fixed set of rules for getting it right. There are however some design principles which can help to guide you towards good designs.

5.2.1 Design principles

Amongst the many sets of design principles, the following six have attracted general support

1. Structure
2. Parsimony
3. Modularity
4. Portability
5. Transparency
6. Conviviality

We shall take each one in turn and briefly look at what they say.

Structure

This is the way the design is organised. An organisational structure which is consistent throughout the design is crucial. Confusion results from disorganisation.

Parsimony

Simplicity and economy are the keys here. Simple things are readily understandable and safe. Minimalist solutions are generally the cheapest.

Modularity

Modularity describes the relationship between the sub-components of a design - the modules. There are two very important measures in assessing the modularity of a design -

1. **Cohesion** is a measure of the strength of the functional association of the elements within a module. Increasing cohesion generally increases *understandability*.

2. **Coupling** is a measure of the degree of interdependence between modules. Decreasing coupling generally increases *adaptability*.

Portability

A system will be applicable to a wider set of problems if it is not unnecessarily tied down to a specific platform or environment. Platform dependence might manifest itself as a reliance on a particular processor, operating system or peripheral. Environmental dependence might result from unwarranted assumptions about the use to which the design or system will be put or through assumptions about the users. Of course, the principles of user-centred design encourage us to design systems with particular user groups in mind. Judgement is called for here.

© HERIOT-WATT UNIVERSITY 2002

Transparency

Systems should be designed to be readily understandable and obvious in use. Hidden assumptions should be avoided.

Conviviality

This principle refers to comfort of use. User friendliness and user-centered design are the key concepts here. See the trade-off with portability though.

5.2.2 Design types

In addition to the design principles described above it can also be helpful to try to identify the type of design which you are undertaking. There are three pairs of alternative design types

1. Logical versus Physical Design
2. Functional versus Object-Oriented
3. Top-Down versus Bottom-Up

The options within each pair are not necessarily mutual exclusive. They might both have a role to play in the overall design but they shouldn't be used simultaneously or a messy and confused design will result. Again, we shall take each pair in turn and briefly look at how they differ.

Logical versus physical design

A logical design describes *what* you are trying to deliver. It is concerned with the relationships between the components of the design. It is most appropriate at the earlier, higher level, stages of the design process. Physical design is more detailed and describes *how* the functionality is to be delivered. It details the nuts and bolts of the system and is more appropriate in the later, lower level, stages of the design process.

Functional versus object-oriented

Functional designs identify what is to be done, the data that is required and how the function will be achieved. This type of design is works with processes, activities or procedures. Indeed, it is sometimes called procedural design. Object-oriented design takes a different perspective. It first identifies what the parts are. The attributes which the parts possess and what can be done to/with them then become the focus of the design exercise.

Top-down versus Bottom-up

Top-down designs break the whole down into parts whilst bottom-up designs build the whole up from the parts.

5.3 Development

Most of what you will learn about development methodologies will be supplied by your fellow tutees in your tutorial group. It is essential that you play your part in this by

© HERIOT-WATT UNIVERSITY 2002

delivering the best information that you can to your colleagues.

We shall confine ourselves to discussing some general points here. We have already met the System Development Life Cycle (Licker 1987). There are two further life cycles which are often presented in the Software Engineering literature.

The Software Life Cycle (Sommerville 2000) has five stages

1. Requirements definition
2. System & software design
3. Implementation & unit testing
4. Integration & system testing
5. Operation & maintenance

The Software Life Cycle commences with the, by now familiar, analysis and specification of the system and proceeds through the design stage to the implementation and testing of individual units or modules. Large systems are generally composed of many modules and these need to be integrated and each sub-system tested as the integration proceeds. Finally, the life of a piece of software does not end with its delivery. Operation and maintenance are also key parts of the overall life cycle.

The life cycle describes a highly iterative process with each stage possibly looping back to any previous stage. This iteration reflects a reality which the theorist often overlooks but which the seasoned practitioner is all too well aware of

It is sometimes only in the later stages of a development that a flaw in the logic of an earlier stage becomes evident.

The implementation stage of the Software Life Cycle can be expanded on to produce a Programming Life Cycle (Stair & Reynolds (2003)

1. Systems investigation
2. System analysis
3. System design
4. Language selection
5. Program coding
6. Testing & debugging
7. Documentation
8. Implementation (conversion)

This life-cycle is embellished to produce an eight-stage process which, once again, starts with the systems analyst but now ends at the implementation. Along the way key issues such as language selection, debugging and documentation are explicitly represented. These sub-divisions of the implementation stage in the earlier life-cycles can facilitate the assignment of personnel within a team to the various tasks required.

© HERIOT-WATT UNIVERSITY 2002

5.4 Teamwork

Large projects require a team effort. A good team will be composed of individuals with different specialisms and also different character traits. Teams composed of very similar members can generate much friction and be quite unproductive. Try to spot some of the following eight types of team member amongst the Robbie CX30 development team.

1. **The plant**

   A creative and imaginative but unorthodox individual. This type of person can be a bit hard to manage but provides many of the team's ideas.

2. **The enabler**

   A mature and confident individual who can chair meetings. This kind of person can be a bit manipulative but they listen well and can clarify the team's goals.

3. **The fixer**

   An extrovert but amiable individual who is always on the go. This type of person may be a bit undisciplined but often has a lot of useful contacts.

4. **The shaper**

   A dynamic and outgoing individual who likes to take the lead. This kind of person has a tendency to bully others but is good and finding ways around obstacles.

5. **The monitor**

   An introvert and thoughtful individual who rarely gets things wrong. This type of person can be a bit slow but is very discerning and generally the ”rock” of the team.

6. **The counsellor**

   A conciliatory and perceptive individual who is sensitive to other people's needs. This kind of person can be a bit indecisive but is essential to the team's harmony.

7. **The workhorse**

   A disciplined and reliable individual who gets on with things in a practical way. This type of person can be a bit unimaginative but is good at making other people's ideas work.

8. **The worrier**

   A stickler for detail who is rather pedantic and never lets go of things. This kind of person can be very annoying but is useful at spotting mistakes and ensuring that deadlines are met.

You can probably recognise elements of yourself in some of the above. More than one of them in fact. Remember that. We all have many different aspects to our characters.

Team dynamics is an important area of study in social psychology. Issues which have been identified as of major importance in teamwork include communication, leadership and conflict resolution. How were these matters handled on the Robbie CX30 project?

© HERIOT-WATT UNIVERSITY 2002

5.5 Summary and assessment

At this stage you should be able to

- explain the steps involved in building systems
- discuss design principles and design types
- outline the System Development Life Cycle
- outline the Software Life Cycle
- outline the Programming Life Cycle
- use a variety of development methodologies
- relate the issues involved in team working

End of topic 5 test

5 min

Q1: Which of the following is NOT a stage in the Systems Development Life Cycle?

a) Installation
b) Physical design
c) Operation and maintenance
d) Winding up

Q2: Design is what kind of process?

a) Algorithmic
b) Creative
c) Provable
d) Random

Q3: Which of the following was NOT suggested as a design principle?

a) Conviviality
b) Modularity
c) Parsimony
d) Sequentiality

Q4: Designing from detailed components through to complete systems is

a) Bottom-up design
b) Logical design
c) Object-orientated design
d) Physical design

Q5: Functional design is concerned with

a) Attributes
b) Components
c) Procedures
d) Relationships

Q6: Who will be telling you most about development methodologies?

© HERIOT-WATT UNIVERSITY 2002

a) Fellow tutees
b) This material
c) Tutor
d) You

Q7: The Software Life Cycle differentiates between the following types of testing ?

a) Good & bad
b) Primary & secondary
c) System & unit
d) Thorough & partial

Q8: The Programming Life Cycle explicitly refers to the selection of

a) Algorithms
b) Data structures
c) Programming language
d) Hardware platform

Q9: Teams need to be composed of individuals who

a) Are very similar
b) Get on well
c) Have different skills
d) Work hard

Q10: A thoughtful but introvert person might make a good

a) Counsellor
b) Enabler
c) Monitor
d) Worrier

5.6 Assigned task

Assigned task

1. Read ”The Case of the Killer Robot” Articles 7 and 8 (Epstein 1997 or Taylor 2002) before embarking on Topic 6.

2. You should now turn the notes you made on the topic assigned to you in Topic 4 into a 1000 word submission for **Assignment 2**. This assignment should be submitted at your next tutorial. It will be assessed and the mark will account for 33% of your final mark in the Praxis Unit.

3. You should also identify about 3 key points from Assignment 2 for presentation at your next tutorial

© HERIOT-WATT UNIVERSITY 2002

Reminder of Assigned Topics 10 - 18

Structured Systems Analysis & Design Method (SSADM)	1st Tutee
Dynamic Systems Development Method (DSDM)	2nd Tutee
Waterfall Model of Software Development	3rd Tutee
Unified Process Model of Software Development	4th Tutee
Rapid Prototyping	5th Tutee
Extreme Programming (XP)	6th Tutee
Reusable Software	7th Tutee
Open Source	8th Tutee
PRINCE Project Management Method	9th Tutee

5.7 References

Epstein, R.G., 1997, *The Case of the Killer Robot*. John Wiley & Son.

Licker, P.S., 1987, *Fundamentals of Systems Analysis*. Boyd & Fraser.

Sommerville, I., 2000, *Software Engineering, 6th edition.* Addison Wesley.

Stair, R.M. & Reynolds, G.W., 2003, *Principles of Information Systems, 6th edition.* Thomson

Taylor, N.K., 2002, *The Killer Robot* [online]. Heriot-Watt University (MACS), 16th December 2002 [cited 7th July 2003]. SHTML. Available from:http://www.macs.hw.ac.uk/~nkt/praxis/epstein/index.sht

© HERIOT-WATT UNIVERSITY 2002

Answers to questions and activities

5 Design and development

End of topic 5 test (page 7)

Q1: d) Winding up

Q2: b) Creative

Q3: d) Sequentiality

Q4: a) Bottom-up design

Q5: c) Procedures

Q6: a) Fellow tutees

Q7: c) System & unit

Q8: c) Programming language

Q9: c) Have different skills

Q10: c) Monitor

© HERIOT-WATT UNIVERSITY 2002