COM1030 – Requirements Engineering

Crossover part 1

Lecturers: Prof Mike Holcombe

Dr. Salim Vanak

Lectures: Mondays 3.00-5.00

Laboratories: Tuesday 10.00-11.00 Lewin Lab, Regent Court.

Assessment is by Project Assignment only – no exam.

Module structure

- Combination of :
  - Lectures
  - Laboratories
  - Team project work
- The aim is to experience the processes involved in building a large software system
- You will work in a team – your tutor group
- You will be managed by experienced software engineers
- You will use a professional software development environment and tools

What is Software Engineering?

- Deciding what software to build – *business analysis* and *requirements elicitation*
- *Specification* and *design* – translating user needs into technical descriptions
- *Implementation* – programming the system
- *Testing* – making sure that it is *fit for purpose*
- *Delivery, installation* and *maintenance*

Why is it difficult?

- All the stages are hard to do
- Requirements can change as we build the system
- Most systems are extremely large
- Safety and business critical systems are particularly tricky and expensive to build
- Strong pressure on software engineers to deliver quickly and cheaply and to budget

Why is Sheffield good for Software Engineering?

- The first Software Engineering degree
- Top department for Testing
  - Mercedes-Benz uses our software to test their automotive software
  - ARM's testing has been improved by 1000% using our methods
  - Early adopters of agile methodology – *Extreme Programming* – strong support from Kent Beck
  - NASA has adopted our design methods for their next generation smart satellites

Our unique industrial aspects

- Year 1 Xover – teamwork, big systems, full software engineering process, design tools
- Year 2 Software Hut – sponsored by ACCENTURE and IBM – teams build software for real business clients
- Year 3 Genesys Taster – sponsored by IBM – Microsoft Innovation Centre
- Year 4 Genesys – full experience – IBM mentors
- Genesys is a commercial software house entirely run by senior students.
- Genesys counts as industrial experience

Special Sheffield features

- Famous software engineer from IBM will teach you next Monday – she will teach you how to improve your negotiation and client skills
- Accenture experts teach Software Hut students how to manage a team
- IBM experts teach Genesys students about how to design effective user interfaces

Software engineering – a brief overview

- Started as a recognised subject in the 70s
- Prior to that writing software was a craft activity – everyone did their own thing
- Problems arose due to:
  - Programs getting bigger – not just written by one person
  - Maintenance was very hard – especially of code written by someone else

An engineering process

- Lessons were taken from other engineering subjects
- Make the process more organised
- Document it carefully
- Introduce a Quality Assurance process
- Standardise the languages and notations
- Monitor progress and improve it

Progress was slow

- Many still carried on in the old ways
- Software quality was terrible
- Software was very expensive to build
- Many projects were abandoned
- Languages and design notations proliferated
- There was little credible research about the best ways to build software

Welcome to 2006!

- Fashions have changed
  - Cobol, Fortran, Algol, Pascal, C, C++, Java etc.
- Each language has good and bad points
- Each language becomes a religion!
- Methodologies come and go
  - Waterfall, Spiral, Rapid Applications Development, Agile etc.
- Each with its adherents

Basic Problems still

- Many projects fail
- Many are over budget
- Many are delivered late
- Many are of poor quality
- Many are difficult to maintain
- Have we really progressed?

Failures

- For some software sectors more than 50% of projects either do not get delivered or are unusable
- 25% are usable – just – but are regarded as of very low quality
- 25% can be used as intended
- No other industry has this sort of record!

WHY?

Let's look at some failures

- NHS system – not failed yet but in trouble
- LIBRA – National system to run the Courts of Justice - abandoned
- Tax Credit system – beset with incorrect payments
- Air Traffic Control – Swanwick – partially finished but still needs a lot of work

But there are successes

- Photoshop
- CS2 Illustrator
- …..

Where do projects go wrong?

- Bad business analysis – does not identify the objectives of the system and how it will fit into the business – Sinclairs
- Incorrect requirements identified – *if the requirements are wrong everything that follows is wrong*
- Design of the system is poor – technology is inappropriate; the system cannot be built in the way wanted; performance is inadequate etc.

More problems

- Programming is of poor quality
- Testing is inadequate – more than 50% of any project is taken up with testing and debugging – this has to be done well
- System cannot be maintained – no-one can understand the code and documentation when new features need to be added – a big problem with legacy code – eg. Cobol

What can be done?

- Think about these problems and construct a design process that mitigates them, if possible
- Identify those parts of the process where errors are common and make them automated
- Insist on best practice in all aspects
- Program defensively – errors will happen - try to avoid the common ones
- Analyse one's performance and seek to improve

Methodologies

- Waterfall
- Spiral
- V-model
- Agile

Waterfall

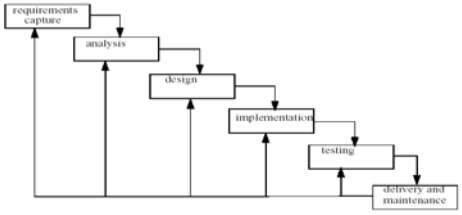

Figure 1. The Waterfall model of software development.

Spiral

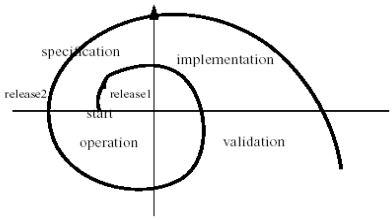

Figure 2. The Spiral model

V model

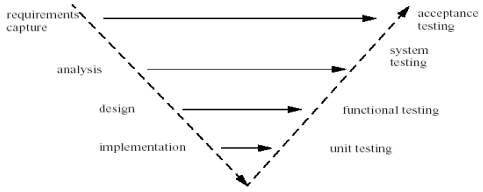

Figure 3. The V model