CM1202
Developing Quality Software

Module Leader - Helen Phillips
Room N/2.21

Lecturer - Philipp Reinecke
Room WX/3.10

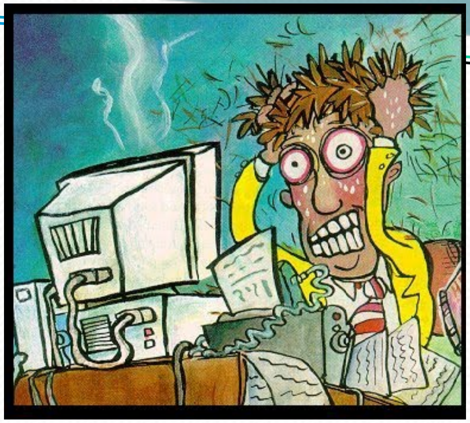
Todays Lecture

- An introduction to the module.
- The core activities of software development.

What are the main aims when developing a piece of software?

What does the word quality mean?

Quality?



What is needed , on time and on budget

The right code written at the right time

Developing and maintaining software systems that

- behave <u>reliably</u> and <u>efficiently</u>,
- are <u>affordable</u> to develop and maintain,
- and <u>satisfy</u> all the requirements that <u>customers</u> have defined for them.

About this module

IS

- Principles & Practices to develop software of quality code
  - Techniques of software development.
  - Different development life cycles
  - Good practice when developing as a team or individual
- Developing what the client wants
- Learning new skills
- Team working (participating of all)

IS NOT

- Teaching programming
- What you want
- Doing what you can already do
- Just the keen ones

Techniques & tools plus some theory

- Introduce you to a series of techniques, and skills:

  Including:
  - Developing functional and non functional requirements
  - UML
  - Risk and Project management
  - Agile Development, plus a series of other ways to develop software applications.
- Plus some theory on
  - Team Dynamics and Communication
  - Quality

Introducing skills needed for the rest of the degree

- You will be expected to use and refer to the skills covered in this module in:

  2nd Year Group project

  Final Year project

  Database systems

  Data structures and OO development

How the module will be organised

Delivery

- Module Both semesters, November to May
- In Tutorial and laboratory session - opportunity of putting into practice what is cover in lectures. Learn through Doing
- Some weeks you will be required to listen to video's, read through PowerPoint slides prior to the sessions
- Module guides you through the various elements of the software development lifecycle.

Teams

- You will be working in a Team, selected by me, for the whole of this module undertaking a series of tasks that work through the stages of developing a piece of software.
- The process is as important as the final output
  - Gain an understanding of the challenges and rewards of working in a team.
  - Each person should have a go at everything.
- You will meet your team in Tuesday's tutorial
- Let me know asap if you don't have access to the CM1202 module on learning central.

Sit in your teams

More of this

Less of this

Other points

- During the module I will suggest that teams provide me with draft version of elements of the portfolio,

- I will provide you with formative feedback, so your team can learn from the assessment and make improvements,

- I will be asking for self and peer assessment of team members contribution as part of each coursework submission,

Assessment (For learning) This semester

Coursework 1,

User Requirements, 15%,

- **Hand out Monday 5th November**
- Opportunity to obtain feedback on Draft versions during tutorial sessions and by emailing a draft pdf document to phillipshr@Cardiff.ac.uk 9:30am on Monday 19th November
- **Hand in 9:30 am Wednesday 3rd December 2018**
- Team submission, pdf document via Learning Central
- Feedback in Week 12 (January)

Coursework 2,

Detailed Design Model, Project Plan and Risk Analysis 15%

- **Hand out Week 11, week beginning 10th December**
- **Spring semester, Wednesday 4th February 2019**

Assessment (For learning) Spring semester

Coursework 3 (30%) Group Demonstration & test cases

Test Cases 5% &

Prototype System to demonstrate key functionality 25%

Hand in 9:30 am, Week 10 week day to be confirmed

Individual Report 40%

Report to evaluate techniques and reflect on learning,

Hand in 9:30 am, Friday week 11, 12th April 2019

Questions?

- What else do you want to know?

On with the module

Developing Software

- Software Development is a profession that looks for concepts, skills and techniques needed to develop software of a suitable quality.

- Software is important to us BUT we’re not very good at creating it.

  This is why subjects like this are important

Why is Software Development so challenging?

- Changeability
- Complexity
- Conformity
- Invisibility

F.P. Brooks, No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, 20(4):10 - 19, April 1987

Software needed has changed

- Characteristics of software projects have changed, they used to be
  - Scope of the projects was small with well-defined tasks
  - Difficult to make changes, Hardware was a limiting factor
- Projects have grown larger in scope and complexity.
- The development of large-scale, complex software systems needs to be considered as a **team** activity
- Techniques developed for programming small projects *did not work* on larger & more complex projects.
- The problem became widespread coining the term "software crisis"

The Solution?

- In 1969, at a conference organised by NATO, the term *Software Engineering* was first mentioned.
- Engineering projects were seen to be much more successful than software projects.
- Engineering is the application of scientific and mathematical principles toward **practical** ends.
- The development of software could benefit by taking an engineering approach

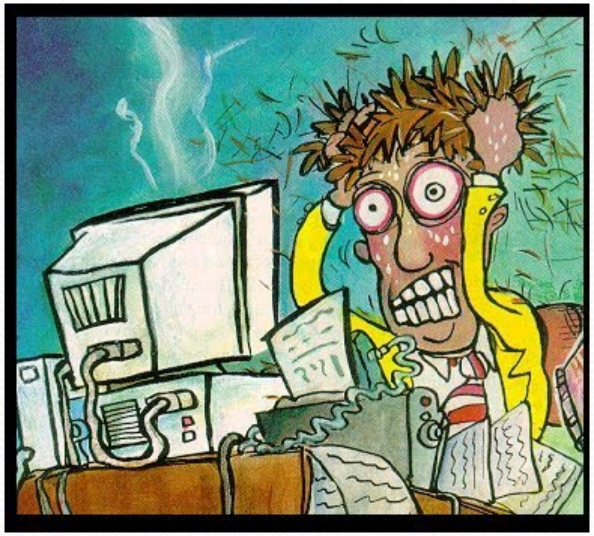
No solution, but tools can help

- Yourdon (1993) argues that developers need to use a combination of tools, techniques and methods to improve quality and productivity in software development
- The combination of appropriate methods, tools and techniques depends on the particular software development project
- We therefore need to be able to justify our choices depending on the context

Software should support users effectively

- Functionality should fit users’ tasks
- Not only build system in right way –build the right system!

  Verification and Validation
- Need to consider quality aspects as well as functionality



Core activities of software development

There are several different software development methods / methodologies. However they all contain the same ‘building blocks’ Core activities – the difference is the order and detail that each activities is gone into at any time.

- Requirements gathering
- Design
- Coding and debugging – (development)
- Testing
- Deployment

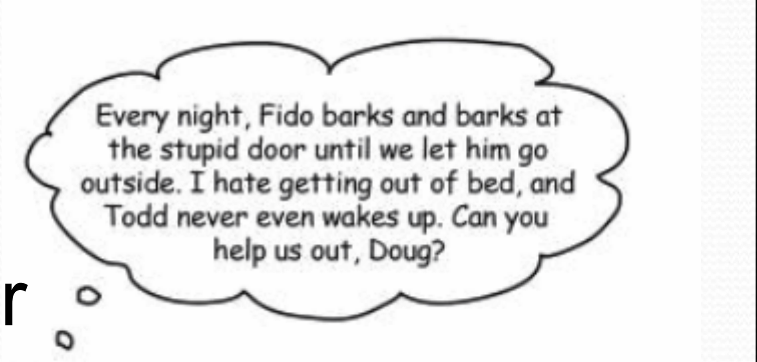
Plus

- Maintenance

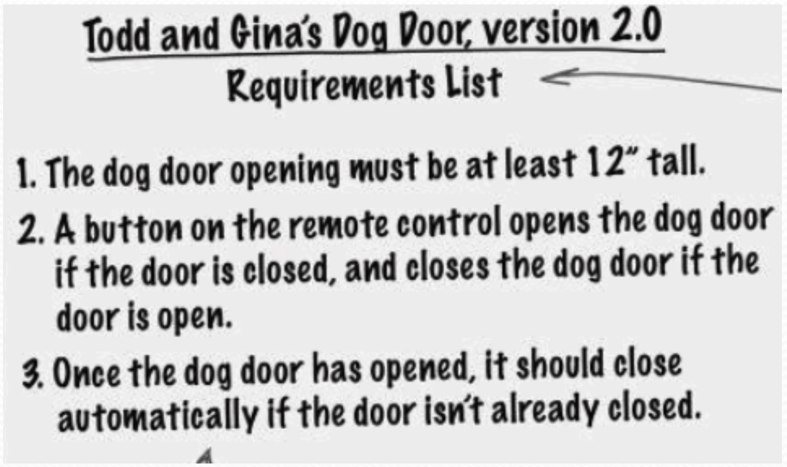
Requirements gathering

Requirements Engineering

I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

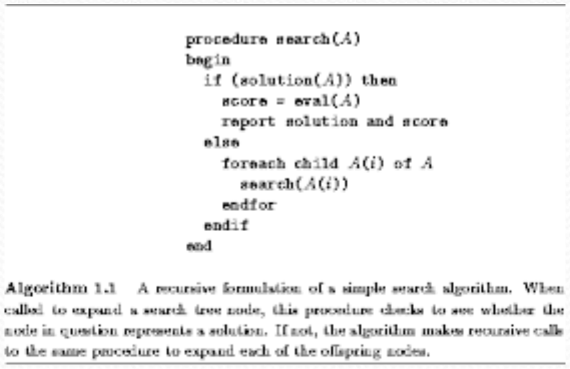

Every night, Fido barks and barks at the stupid door until we let him go outside. I hate getting out of bed, and Todd never even wakes up. Can you help us out, Doug?

- Analysing what the customer wants
- In terms that the customer understands
- What not How
- Acceptance tests written


Todd and Gina's Dog Door, version 2.0
Requirements List

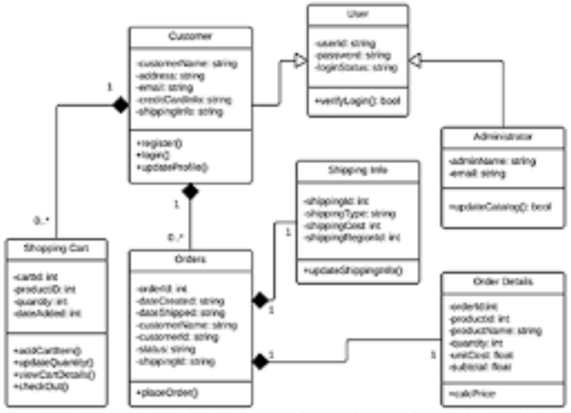
1. The dog door opening must be at least 12" tall.
2. A button on the remote control opens the dog door if the door is closed, and closes the dog door if the door is open.
3. Once the dog door has opened, it should close automatically if the door isn't already closed.

Design



```
procedure search(A)
begin
  if (solution(A)) then
    score = eval(A)
    report solution and score
  else
    foreach child A(i) of A
      search(A(i))
    endfor
  endif
end
```

Algorithm 1.1 A recursive formulation of a simple search algorithm. When called to expand a search tree node, this procedure checks to see whether the node in question represents a solution. If not, the algorithm makes recursive calls to the same procedure to expand each of the offspring nodes.



- Overall architecture

  For quality software : understandable and maintainable
- Interfaces
- between components
- between the system and its environment
- Data structures are specified
- Algorithms
- Design the tests

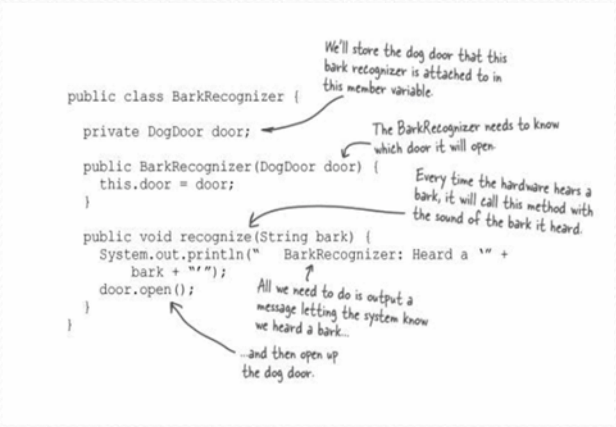
Development

- <u>Modularity</u> break code up into smaller manageable parts.
  - Each individual piece can them be tested independently.
- Object Oriented
- Version control (GIT Lab)
- Test Driven Development

```
<!-- BEGIN: Google Trusted Stores -->
<script type="text/javascript">
  var gts = gts || [];

  gts.push(["id", "STORE_ID"]);
  gts.push(["badge_position", "USER_DEFINED"]);
  gts.push(["badge_container", "GTS_CONTAINER"]);
  gts.push(["locale", "PAGE_LANGUAGE"]);
  gts.push(["google_base_offer_id", "ITEM_GOOGLE_SHOPPING_ID"]);
  gts.push(["google_base_subaccount_id", "ITEM_GOOGLE_SHOPPING_ACCOUNT_ID"]);

  (function() {
    var gts = document.createElement("script");
    gts.type = "text/javascript";
    gts.async = true;
    gts.src = "https://www.googlecommerce.com/trustedstores/api/js";
    var s = document.getElementsByTagName("script")[0];
    s.parentNode.insertBefore(gts, s);
  })();
</script>
<!-- END: Google Trusted Stores -->
...
...
<div id="GTS_CONTAINER"></div>
```



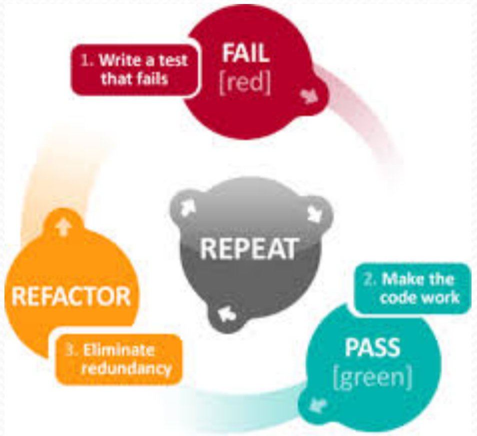
```
public class BarkRecognizer {

  private DogDoor door;

  public BarkRecognizer(DogDoor door) {
    this.door = door;
  }

  public void recognize(String bark) {
    System.out.println("   BarkRecognizer: Heard a '" +
        bark + "'");
    door.open();
  }
}
```

We'll store the dog door that this bark recognizer is attached to in this member variable.
The BarkRecognizer needs to know which door it will open.
Every time the hardware hears a bark, it will call this method with the sound of the bark it heard.
All we need to do is output a message letting the system know we heard a bark...
...and then open up the dog door.



Testing

- Unit Testing:

  Each unit of code is tested.

  Black box – to spec / White box – code

Integration and Systems Testing:

- Individual components are integrated to form the complete system;
- System is tested as a whole.

Validation Testing

Ensuring software actually does what it is supposed to

- Validation – Functional Testing &
- Software Quality Assurance

Deploy & Maintenance

- From introduction to phased out.

SO why does this matter to US as developers, (Quality)

Changes to the software will be needed to:

- Repair software faults;
- Accommodate changes in operating environment;
- Add or modify system functionality.

Software Development lifecycle

- There are a number of different software development lifecycles
- Which one to select will depend on the nature of the task.
  - How complex is the problem?
  - How defined are the user requirements?
  - Are these requirements likely to change?
  - How much input do you want from the customer and When do you give the customer working code?
- The software development lifecycle selected can be used as a basis for developing a project schedule.

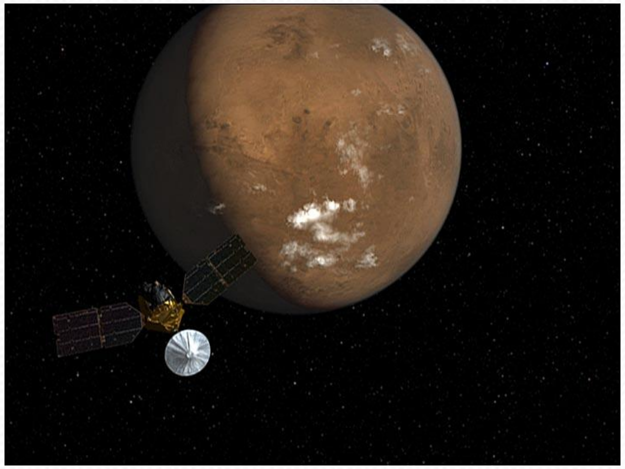
When to code

Online store

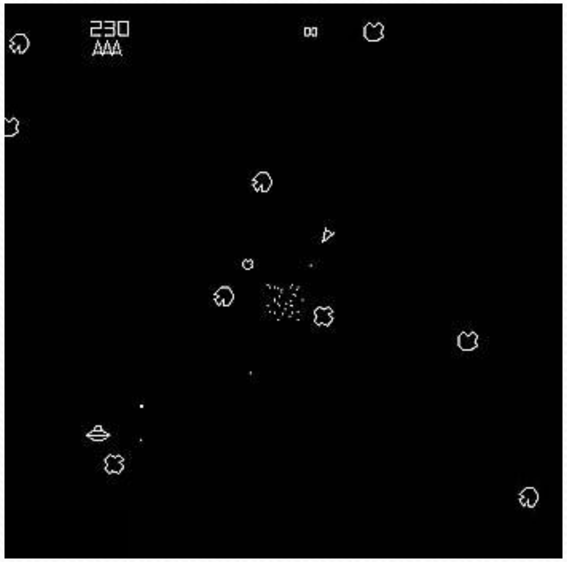
STARSHIP MODELER

SINCE 1996

A Complete Web Resource for Sci-Fi and Space Scale Modeler



Mars mission



Simple game

Tutorials (In the Lab)

- Group A – 9am
- Group B – 10am

Same Groups as for CM1101

Preparation for Tomorrow's Tutorials

Material on Learning Central

- Read
  - '*Guidelines for Student Group Projects'*
  - *'Code of Conduct for Student Team Projects'*
  - *Belbin's in a nutshell*
- *Listen to Video's*
  - *Software Engineering*
  - *Belbin's team Roles*