Enterprise Computing Introduction

Professor Stephen Gilmore
School of Informatics
The University of Edinburgh

January 15, 2015

About this course

- Enterprise Computing is a Level 10 course.
- It is normally taken by undergraduate students in year 3.
- There is a coursework component and a written exam.
  - The coursework component is worth 25% of the assessment.
  - The written exam is worth 75% of the assessment.
- The coursework is in two parts.
  - Part 1 is zero-weighted: it is just for feedback.
  - Part 2 accounts for 100% of the coursework mark.
- The examination is a 2-hour paper taken in the main exam diet in April/May.

What is “The Enterprise”?

If you were at the lecture then you heard the discussion of what “the enterprise” is.

Enterprise computing systems (1/2)

- Enterprise computing systems are very different in nature from self-governed development projects such as apps and hobby projects.
- For hobby projects developers themselves determine what is the best arrangement of the application, subject to some feedback from users.
- In enterprise computing systems one may have to work to determine what is the best arrangement of the application: it may not be at all obvious.

Enterprise computing systems (2/2)

- Enterprise computing systems may be governed by performance or other regulatory requirements and are subject to scrutiny from external regulators. External regulators can apply fines or other penalties for non-compliance.
- An enterprise computing system depends crucially on data. Data doesn't solve any problems. Data can *be* the problem.
- Enterprise computing systems may be required to serve a very broad user base, and cannot narrow their user base down to a small demographic. There can be legal requirements to make systems accessible to all.

Enterprise computing: different from apps (1/3)

- Mobile phone apps or web apps are subject to regulation on their design, and implementation and content.
- The App Store scrutinises apps which are submitted for content, and for malware.
- The Google Play Store does not vet apps in the same way, although all Android apps should follow the three major Android Design Principles (Enchant Me / Simplify My Life / Make Me Amazing).
- However, in use these apps are largely unregulated, with the possible exception of in-app purchases.

Enterprise computing: different from apps (2/3)

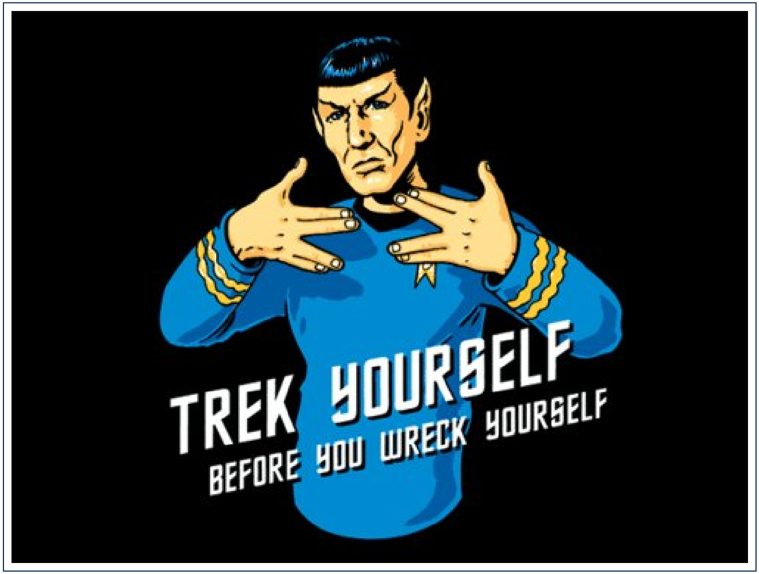
Let's hear from an app innovator, Tech CEO Thomas Fisk of PicSong.

`https://www.youtube.com/watch?v=zpNgsU9o4ik`

Enterprise computing: different from apps (3/3)

- Apps can be simply for fun or because we can.
    - "Enchant Me." / "If you want to do that for some reason."
- In contrast, enterprise computing systems are purposeful systems.
- We have an enterprise: we have a mission. People care.
- The spirit that you bring to enterprise development can be quite different from the spirit that you bring to app development.
- People need to use these systems. They are "real world" systems. Enterprises depend on them.
- Enterprises have stakeholders who may have widely different points of view about how the enterprise would best be run.

Enterprise computing is professional computing



The structure of enterprise computing systems

- Enterprise computing systems consist of distributed components.
- They will often consist of servers such as app servers and database servers in the back-end; and mobile apps, responsive web apps, or web sites in the front-end.
- They will typically contain web services which supply semi-structured data.
- They will typically exchange semi-structured data in the form of XML or JSON.

Outline of the coursework

- The goal of the coursework is to make use of a newly-released Open Data API made available by Transport for Edinburgh.
- The API is available at `http://tfe-opendata.readme.io/v1.0`
- From the API you can request data about stop locations, service routes, journey planning, timetables and live bus locations for the city of Edinburgh.
- To access the API and use it you need an API key which looks like this:`0c627af5849e23b0[redacted]030[redacted]73525508[redacted]`

Some of the letters and numbers of the API key have been blanked out. If you were at the lecture then you heard me say which letters and numbers these were.

Functional requirements (1/2)

- The functional requirements for the project are intentionally open. You are required to use the data which is made available by the Transport for Edinburgh Open Data web service, but it is not specified how you are to use it.
- Underspecification such as this is very common in practical software development. In contrast, it is very rare indeed to receive a fully formal specification which details all of the development work which is to be done.
- The guidance that you have been given is that you should make the open data information accessible. The data gives you the potential to do something: you decide what.

Functional requirements (2/2)

- As is often the case in practical software development, for the software development part of this course you have some freedoms which you should use as you see fit, and some constraints which you just have to learn to live with.
- You are free to choose the technology which you use to implement your system on the server side (if you even have a server side). You could use Python, PHP, Java, or another language of your choice.
- You are constrained to use TypeScript on the client side. TypeScript is a typed dialect of the JavaScript language.

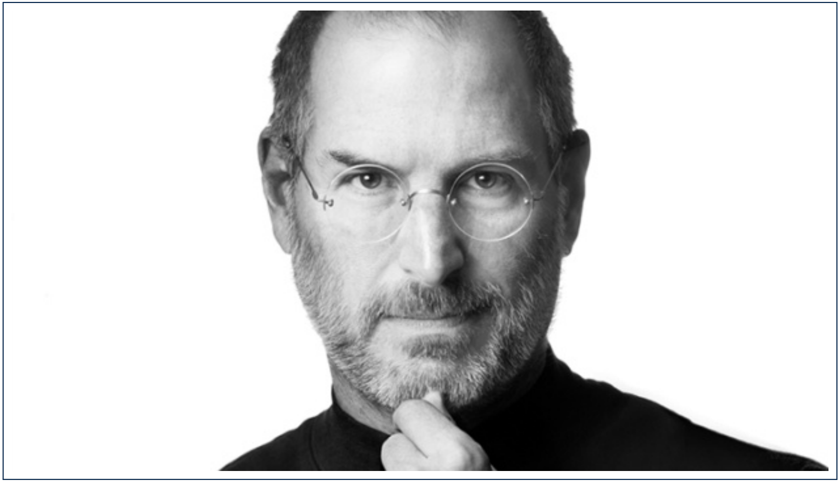
Non-functional requirements (1/2)

- Non-functional requirements are different in nature from functional requirements.
- They are more difficult for non-experts to express precisely.
- They can be subjective. *"The old system is too hard to use: the new one should be easy to use"*.
- They are often not expressed precisely in quantitative terms. *"The old system is too slow: the new one needs to be really, really fast"*.
- Non-functional requirements are not always listed in order of importance. (*"I wish they'd told us that first."*) They may be presented in a more-or-less random order.

Non-functional requirements (2/2)

- Non-functional requirements are more likely to be spoken in a meeting than to be written down. It may be your job to write them down. That is the case here.
- They may be issued one-at-a-time over a series of meetings, rather than presented in full at the beginning of a project. Also the case here.
- A client may be more likely to change their mind about a non-functional requirement than a functional one.
- A client may be more hesitant about a non-functional requirement than a functional one. (*"It should be like this."* versus *"It must do this."*)
- Non-functional requirements may be more likely to be formulated negatively (*"We don't want . . . "*).

Non-functional requirement (#1 of 10)



This picture represents non-functional requirement #1. If you were at the lecture then you heard me explain in words what it means.

The Thirteen Coursework Teams

1. Team Klingon
2. Team Romulan
3. Team Vulcan
4. Team Kirk
5. Team Spock
6. Team Scotty
7. Team Bones
8. Team Sulu
9. Team Uhura
10. Team Chekov
11. Team Transporter
12. Team Phaser
13. Team Tricorder

If you were at the lecture then you heard me tell you which team you were on.

Three terms you should know

When discussing software projects and their viability, there are three terms which you should know.

Three terms you should know

- "-ilities".
- Technical debt.
- Bus factor.

If you were at the lecture then you heard the discussion of what these terms mean.

Things to do now

Six things to do now

- Arrange a meeting of your team. How will you organise yourselves?
- Start to define the work (be creative!) and to divide up the work (be fair!).
- Learn about the different skills on your team.
- Visit the Transport for Edinburgh Open Data API and begin learning about it.
  - `http://tfe-opendata.readme.io/v1.0`
- Find out about TypeScript. Start learning TypeScript.
- Find out about JSON. Start learning JSON.