Inf2- SEPP
Lecture 20: Software deployment and maintenance

Cristina Adriana Alexandru

School of Informatics
University of Edinburgh

Up until now

- Requirements engineering
- Design
- Construction/implementation
- Refactoring
- Verification, validation and testing

This lecture

- Deployment
  - What is deployment
  - Is deployment the reason why software projects fail?
  - Key issues around deployment
- Maintenance
  - What is maintenance?
  - Maintenance challenges
  - Being disciplined in software evolution: Release management
  - Maintenance technique: Re-engineering

What is deployment?

Getting software out of the hands of the developers into the hands of the users.

Some stats on software projects:

- More than 50% of commissioned software is not used, mostly because it fails at deployment stage.
- 80% of the cost of (commissioned) software comes at and after deployment.

Is deployment the problem?

Not always.

Often, problems *show up* at deployment which are actually failures of requirements engineering.

Such problems can be very hard or impossible to fix, in a large system. e.g. National Programme for IT

However, there are also genuine transition issues.

Key issues around deployment

- Business processes. Most large software systems require customers to change the way they work. Has this been properly thought through?
- Training. No point in deploying software if its customers can't use it.
- Deployment itself. How physically to get the software installed.
- Equipment. Is the customer's hardware up to the job?
- Expertise. Does the customer have the IT expertise to install the software?
- Integration with *other* systems of the customer.

Deployment itself

Tools are available to help you deploy software. Such systems can:

- make the system installable on different platforms
- package the software
- make it available (nowadays over Internet)
- give the user turn-key installers, which will:
  - check the system for missing dependencies or drivers etc.
  - install the software on the system
  - set up any necessary licence managers
  - . . .

What is maintenance?

The process of changing a system after it has been delivered.

Kinds

- **Fixing bugs and vulnerabilities**:
  not only in code, but also design and requirements
- **Adapting to new platforms and software environments**:
  e.g. new hardware, new OSes, new support software
- **Supporting new features and requirements**:
  necessary as operating environments change and in response to competitive pressures

Maintenance challenges

- Popularity of maintenance work
  - unpopular – seen as less skilled, can involve obselete languages
- Often a new team has to understand the software
- Development and maintenance often separate contracts
  - De-incentivises developers paying attention to maintainability.
- How software structure changes over time
  - Structure degrades, making maintenance harder
  - Not only code impacted, also other software aspects, e.g. user documentation
- Working with obselete compilers, OSes, hardware

Being disciplined in software evolution: Release management

Discipline in the evolution of software is (at least) as important as in its development.

- gather change requirements: new features, adapting to system/business change, bug reports
- evaluate each; produce proposed list of changes
- go through normal development cycle to implement changes – *ensuring that you understand the software*, which may be non-trivial.
- issue new release

Unfortunately, emergencies happen, and things have to be done with urgency. If at all possible, go through the normal process afterwards.

Maintenance technique: Re-engineering

Re-engineering is the process of taking an old or unmaintainable system and transforming it until it's maintainable. This *may* be considerably less risky and much cheaper than re-implementing.

Re-engineering may involve:

- Source code translation e.g. from obsolete language, or assembly, to modern language.
- Reverse engineering i.e. analysing the program, possibly in the absence of source code.
- Structure improvement, especially *modularization*, *architectural refactoring*
- Data re-engineering, reformatting and cleaning up data.
- Adding adapter interfaces to users and newer other software

Issues:

- What are the requirements?
- Which bugs do you deliberately preserve?

Reading

Recommended: Sommerville SE Chapter 9: ”Software maintenance”