Lecture 20: Software Deployment and Maintenance

Inf2-SEPP

Adriana Sejfia

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

Deployment

What is Deployment?

- Getting software out of the hands of the developers into the hands of the users.
- Key statistics:
  - 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.
- Problems that show up at deployment are often actually failures of requirements engineering.
  - Such problems can be very hard or impossible to fix in a large system.
  - Example: National Programme for IT (UK)
- However, there are also genuine transition issues at deployment time.

Key Issues Around Deployment

- Business processes
  - Most large systems require customers to change how they work — has this been properly thought through?
- Training
  - No point in deploying software if 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

- Tools are available to help deploy software. Such systems can:
  - Make the system installable on different platforms
  - Package the software
  - Make it available (nowadays over the Internet)
  - Provide turn-key installers, which will:
    - Check the system for missing dependencies or drivers
    - Install the software on the system
    - Set up any necessary licence managers

Maintenance

What is Maintenance?

- The process of changing a system after it has been delivered.
- Kinds of maintenance:
  - Fixing bugs and vulnerabilities
    - Not only in code, but also design and requirements
  - Adapting to new platforms and software environments
    - 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 obsolete languages
- A new team often 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 documentation and other artefacts
- Working with obsolete compilers, OSes, hardware

Release Management

Being Disciplined in Software Evolution: Release Management

- Discipline in the evolution of software is at least as important as in its development.
- Process:
  - 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
- Emergencies happen — if possible, go through the normal process afterwards

Maintenance Technique: Re-engineering

Re-engineering

- Taking an old or unmaintainable system and transforming it until it's maintainable.
- May be considerably less risky and much cheaper than re-implementing from scratch.
- Re-engineering may involve:
  - Source code translation (e.g. from obsolete language or assembly to modern language)
  - Reverse engineering — analysing the program, possibly without source code
  - Structure improvement — especially modularisation, architectural refactoring
  - Data re-engineering — reformatting and cleaning up data
  - Adding adapter interfaces to users and newer software

Re-engineering: Key Issues

- What are the requirements?
  - Original requirements may be poorly documented or lost entirely
- Which bugs do you deliberately preserve?
  - Some systems may depend on existing bugs as unofficial features

Reading

- Recommended:
  - Sommerville, Software Engineering — Chapter 9: "Software Maintenance"