3rd Year Projects

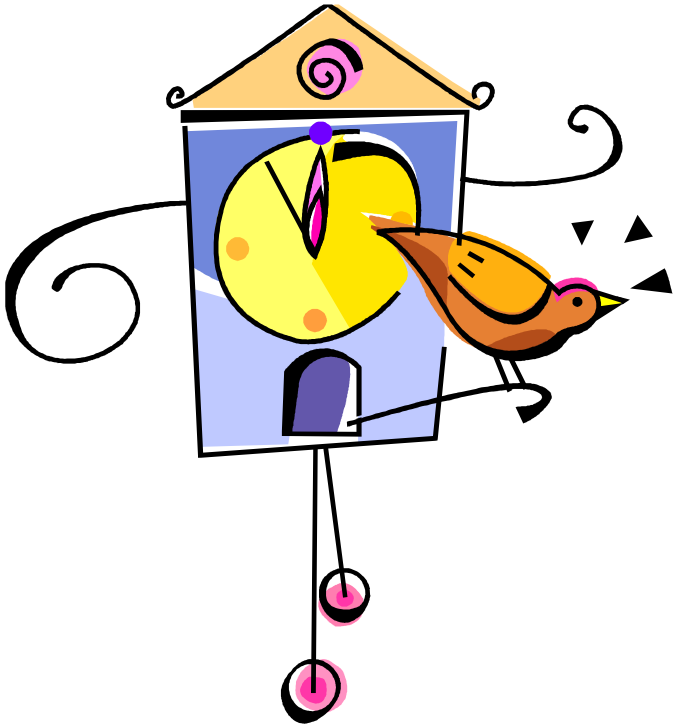
Writing your final report

Slides by Andy Gravell and Kirk Martinez

Plan of the Talk

- The 5 Minute Executive Summary
  - in case you get bored easily ☺
- Detailed Advice on Writing Up
  - applies to all projects
- Writing Up Software Projects
  - specifically about software projects
- Summary

Five Minute Summary



Review of main points

- Hand it in on time
- Don’t use more than 10,000 words
  - in the main body of your report
- Proof-read it
  - perhaps swap-read with a friend
  - someone with better writing skills than you ☺
- Use proper references
- Leave plenty of time to print and bind it….

Content of a typical report

- A clear statement of the problem and goals of the project
- A review of the background literature
- An analysis and specification of the solution to the problem
- A detailed design
- The implementation
- Testing strategy and results
- A critical evaluation
- Conclusions and future work
- References to the literature
- Appendices (if needed)

A typical structure

- Title page
- Abstract
- Contents list
- Acknowledgments
- Chapters:
  - 1. Introduction describing the problem
  - 2. a chapter reviewing approaches/literature
  - 3. a chapter introducing final approach
  - 4... further chapters discussing system implementation or experiment...
  - *n*-1. a chapter reviewing results
  - *n*. Conclusion and proposal of further work and ideas
- References
- Appendices (optional)

Talk to your supervisor

- Your supervisor will advise on structure
  - every project is different
  - so you may need to vary the typical structure
- But don't expect proof-reading from them
  - if you are dyslexic, the University's mentoring service may be able to help (mentors@soton)
  - if your English is particularly poor, you may need to use a professional proof reader, but this is expensive and slow ☹

Look at:

Index of past projects on the web site:

- https://secure.ecs.soton.ac.uk/notes/comp3020/archive/
- you can borrow a copy from Zepler reception

The “project report standards” pages in the *detailed project guidelines*

The *mark scheme*

Detailed Advice

Things You Should Know

- How to use your word processor
  - to check your spelling and grammar
  - to generate the table of contents
  - to manage references
  - using styles, chapter & section numbering, …
- What your supervisor & examiner expect
  - report format and structure
  - writing style (use of personal pronoun)

Why A Report?

- So you can qualify as
  - an honours graduate
    - this report is your dissertation
  - an engineering professional
    - and get a job
- So you can defend yourself
  - to the external examiner
    - if you are on a borderline
  - and (perhaps in future) in a court of law

Working to A Word Limit

- Use concise language
- Use figures
  - diagrams, photographs, tables
- Move detailed material to the appendix*

* Remember to cite your figures (fig. 1) and appendix (see appendix A) as well as your external sources

Approaches to Writing

- Top down
  - start with a list of chapters
  - add section and sub-section headings
  - write each section
- Bottom up
  - start by writing something
  - whatever seems important
  - arrange the material into sections & chapters
- In reality – you will do a mixture of both

Writers Block

- Some of you may struggle to start
  - you get stuck trying to perfect your first sentence
  - you are distracted by displacement activities
  - you can’t start writing yet because you are still building/testing/waiting to be told what to do
- This is an engineering document
  - “fit for purpose”; doesn’t need to be perfect
- Professional writers are usually very disciplined
  - they write a certain amount of words every day
  - usually in the same time and the same place

Backing Up

- It is certain that one of you here will suffer a hard disk crash, or burglary, or fire, or …
- You must be able to carry on regardless
  - in severe cases we can allow one or two days extra, but don't expect several weeks
- This means you need to *backup* over the network, and/or keep your backup memory stick/disk in a *separate room and building*

Statement of Originality

- You are strongly encouraged to include a one or two paragraph statement of originality
  - “this is all my own work” is rarely true
  - you should acknowledge the help you have received
- Was the idea for the project yours, or was it based on an earlier project, or your supervisor’s research?
- The examiners will assume that the analysis, design, implementation, testing, … are your own work
- So tell them where this is not true
  - the design of component X follows a standard technique/pattern described in [source]
  - this is my own code except for <package/class/method> which I have copied from <Internet site/author>

Re-cycling (self-plagiarism)

- Normally, you would be penalised if you re-use material from one assignment in another
- In this case you are encouraged, however, to copy all or part of your progress report
  - this is OK because it's all the same project
  - and has been taken into account in the mark scheme

Project Marking Scheme

- The examiners will consider
  - your progress (interim) report
  - project management and planning
  - technical approach, engineering, analysis, design
  - testing, evaluation, reflection
  - achievement, innovation, challenge, contribution
  - report writing, format, structure, references
  - knowledge & understanding in your viva/demo/report
- The project web page gives assessment descriptors for each of these aspects

Interpreting the Marks Scheme

- There are many kinds of project
  - design/build/test, scientific experiment, systems analysis, large-scale survey, …
- The kind of design, implementation, and testing you do will vary accordingly
  - design implies planning
  - implementation is carrying out your plan
  - testing shows you have achieved your goals
- If you are not sure, consult your supervisor

What Type of Project?

- What has been your main project focus:
  - hardware, software, or something else?
- How much time and intellectual effort have you spent on each aspect?
- Your write-up should cover the aspects that, in your opinion, deserve the credit
  - the limit is 10,000 words (30–40 pages)
  - plus appendixes for e.g. detailed test results

A Poor Beginning

- Don’t start your report, for example, with

  “I decided to use Java Swing for my GUI”
- Why not?

A Poor Beginning

- Don’t start your report, for example, with
  “I decided to use Java Swing for my GUI”
- Why not?
- Before this you should have
  – project goals
  – detailed requirements
  – high level design or architecture
  – comparison of alternative technologies

A Poor Ending

- Don’t finish your report with “I tested it and everything worked”

A Poor Ending

- Don’t finish your report with “I tested it and everything worked”
- After this you should have
  - measurements
  - comparative and critical evaluation
  - reflection
  - future work
  - summary and conclusions

The Report Philosophy

- A *justified* approach
  - alternatives, feature list, selection, known-good practice
- Claims supported by *evidence*
  - measurements, audit trail, log book
- *Appropriate* use of tools, techniques, metrics and methods
  - fit for purpose (engineering perspective)
  - to gain marks (educational perspective)

Evidence

- Evidence helps to avoid misunderstanding
  - and prevent scientific fraud
- Depending on the type of project, you have
  - design diagrams
  - source code
  - measurements, or questionnaire results
  - interview transcripts
- Too much to include in the main report
  - or even as a printed appendix
  - so include a CD/DVD-ROM inside the back cover
  - and a *printed* list of its contents

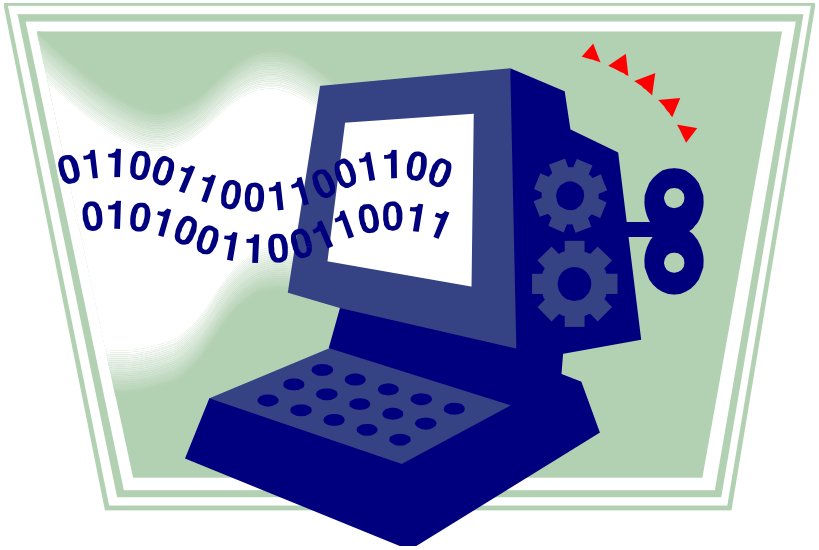
Evaluation and Reflection

- Comparative evaluation (cf. competition)
  - performance graphs, feature lists
- Critical evaluation
  - with respect to your project goals and plan
- Reflection
  - in hindsight, did you use the right tools, techniques, metrics and methods?
  - what did you learn?
  - were your goals and plan sensible?
  - how could you have done it better/differently?

Project Management and Planning

- You should account for your time
  - this is the major expense for most projects
- Compare your initial plan with how things actually went
  - perhaps include project diary as an appendix
- If you fell behind how did you catch up
  - or decide which features to drop
- Did you consider and allow for risks
  - illness, equipment failure or delays

Writing Up Software Projects



BCS Requirements

- the problem & the objectives of the project
- review of the context/literature/competition
- the life-cycle stages undertaken
- the development tools used
- use of V&V at each stage
- rationale for design/implementation decisions
- critical evaluation, review of the plan & any deviations from it, lessons learnt

When to Start Your Write-Up

- Now would be good
- Allow
  - two weeks for final testing
  - three weeks for writing up
  - two weeks for polishing
- So you should stop coding by the end of term at latest, and testing by 1st April
  - even if a few (minor) errors still remain

Appropriate Tools

- Source code control (CVS/RCS/SVN)
- CASE tools (Visual Paradigm/Visio/...)
- IDEs (Eclipse/Visual Studio)
- Build scripts (Ant/Make)
- Code analysers (BoundsChecker/Lint)
- Automated testing (Junit, RoboTest)
- Other: profiling, GUI builder, optimiser, code or document generator

V&V Techniques

- Prototyping or animation
- Customer feedback or sign-off
- Automated model-checking or proof
- Source code analysis
- Assertion checking
- Testing: unit, system, regression, performance, portability, and scalability

Appropriate Design Techniques*

- Requirements: use cases/features/CRC cards
- Architecture: package/class diagram
- Protocol: sequence/collaboration diagram
- Control/GUI app: state diagram/FSM/WebML
- Information System: class diagram/call graph
- Database: ER/database/class diagram

*UML notation is preferred for OO projects

Implementation Techniques

- Interesting or important data structures
- Use pseudo-code or an activity diagram, but *not* flow charts, to explain interesting or important algorithms
  - EL students may use ASM charts, however ☺
- Use/reuse of class libraries or components
- Justified use of trendy technologies
  - XML, web services, RSS, AI, encryption, …
- Any interesting or significant errors
  - how you located and corrected them

Metrics

- Simple counts: LOCs, classes, methods, features implemented
- OO metrics: methods/class, coupling
- Test coverage: branches/methods executed
- Execution time (and complexity?)
- Memory usage (and complexity?)
- Achievement: tests passed, features delivered
- Satisfaction: end-user questionnaires
  - appropriate descriptive/analytical statistics

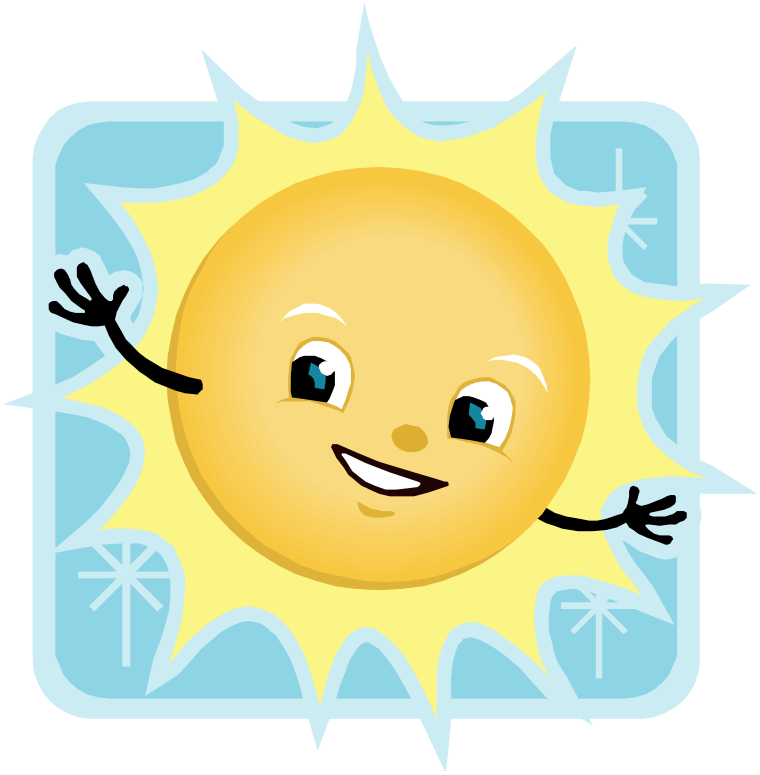
Methods

- Which process did you follow:
  - waterfall, iterative, evolutionary?
- How does this show in your plan?
- Why did you choose this process?
- How well did it work out for you?

Frequently Asked Questions

- Do I need a user manual?
  - is the UI an important part of your project?
  - then print some screen shots, or include the help file as an appendix
- Do I need a system manual?
  - will other people need to build your code?
  - then include your build script as an appendix
  - and an architecture/package/deployment diagram
- Do I need to print the source code?
  - if the code is an important part of your project then yes
  - but if it is more than ~10 pages, save trees and put it on a DVD-ROM
- What are the costs of a software project?
  - your time
  - review your progress against your original plan
- My project isn't object-oriented, do I still need to use UML diagrams?
  - see the previous slide entitled "**appropriate** design techniques"

Summary



Summary

- A professional report and dissertation
- Appropriate tools, techniques, metrics and methods
  - UML class diagram, test plans, …
  - justify your decisions
- Evidence of achievement
  - test and questionnaire results
  - metrics such as #LOCs/tests/features completed
- Comparative and critical evaluation
  - review of progress against plan
- Reflection