# COMP4920 Management and Ethics
# SENG4920 Ethics and Management

Wayne Wobcke

*School of Computer Science and Engineering*
*University of New South Wales*



## Management and Ethics?

- ***Software* Project** Management
  - Practical experience in all phases of the planning and execution of a software project
  - Such as . . .
- **Professional Issues** and Ethics
  - Appreciate the responsibilities of a professional software engineer
  - Understand the ethical dimensions of the IT industry as applied to specific issues
    - Such as . . .
- **Should be done in final year of study**





## Two Student Groups

- COMP Students
  - Will do software project management in COMP3900
  - Do **not** have to do it in COMP4920!
- ENG Students
  - Software Engineering, Computer Engineering, Bioinformatics
  - Have not done (sufficient) software project management
- SENG students enrol in SENG4920
- COMPENG and BINF students enrol in COMP4920 but attend **SENG4920** seminars and meetings





## Learning Objectives

- Professional Issues and Ethics
  - Understand the responsibilities of a professional software engineer
  - Appreciate and apply ethical frameworks to make professional judgements
  - Develop critical thinking in relation to professional issues
  - Gain in-depth understanding of several topical professional issues
  - Appreciate different ways of managing intellectual property
  - Understand the societal context of technology developments
  - Improve communication skills needed to present reasoned arguments

# Learning Objectives

- Software Project Management (ENG)
  - Understand the range of activities involved in a large software project
  - Be able to plan and successfully execute a team-based software project
  - Take on different roles to contribute to team success
  - Understand and appropriately apply software engineering processes
  - Understand the importance of teamwork and communication in a software project





# Course Schedule (Subject to Change)

| | | |
|---|---|---|
| 1 | Introduction to Project Management and Ethics | Wayne Wobcke |
| 2 | Theoretical Underpinnings of Ethics | Stephen Cohen |
| 3 | Agile Software Processes in Practice | Glenn Bieger |
| 4 | Moral Reasoning and Professional Ethics | Stephen Cohen |
| 5 | Legal Perspectives on the Software Industry | David Vaile |
| 6 | Flexibility Week | |
| 7 | Intellectual Property and Software Patents | Stuart Irvine |

*Live lectures may be in Physics Theatre or on Zoom or the lecture may be a recording from 2019





# Assessment (COMP)

- Seminar Participation (20%) – 30 hours
  - Emphasis on participation and preparation **in advance**
  - **Microphone and video expected**
- Lecture Summaries (10%) – 20 hours
  - Summaries and reflections on 4 lectures
- Student Seminar x 2 (30%) – 25-30 hours
  - Team-based presentation on a specific topic of interest
- Company Case Study (40%) – 35-40 hours
  - Report on company-related professional/ethical issue(s)
  - Greater depth than for ENG students





# Assessment (ENG)

- Seminar Participation (10%) – 15 hours
  - Emphasis on participation and preparation **in advance**
- Lecture Summaries (10%) – 20 hours
  - Summaries and reflections on 4 lectures
- Student Seminar x 1 (15%) – 10-15 hours
  - Team-based presentation on a specific topic of interest
- Company Case Study (15%) – 10 hours
  - Report on company-related professional/ethical issue(s)
- Software Project (50%) – 50-60 hours **per person**
  - Project plan – **record** on taskboard (10%)
  - Software product – **implement** project plan (40%)
    - Team-based, agile methods, project management tools

# Assessment Calendar (COMP)

| Week | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| 1 | 13 Sep | 14 Sep | 15 Sep | 16 Sep | 17 Sep | 18 Sep | 19 Sep |
| 2 | 20 Sep | 21 Sep | 22 Sep | 23 Sep | 24 Sep | 25 Sep | 26 Sep |
| 3 | 27 Sep | 28 Sep | 29 Sep | 30 Sep | 1 Oct | 2 Oct | 3 Oct |
| 4 | 4 Oct | 5 Oct **Labour Day** | 6 Oct **Student Seminar** | 7 Oct — | 8 Oct — | 9 Oct **Lecture Summaries 1** | 10 Oct |
| 5 | 11 Oct | 12 Oct **Student Seminar** | 13 Oct — | 14 Oct — | 15 Oct — | 16 Oct **Peer Reviews 1** | 17 Oct |
| 6 | 18 Oct **Flexibility Week** | 19 Oct Jul — | 20 Oct Jul — | 21 Oct Jul — | 22 Oct Jul — | 23 Oct Jul — | 24 Oct Jul — |
| 7 | 25 Oct | 26 Oct **Student Seminar** | 27 Oct — | 28 Oct — | 29 Oct — | 30 Oct **Lecture Summaries 2** | 31 Oct — |
| 8 | 1 Nov | 2 Nov **Student Seminar** | 3 Nov — | 4 Nov — | 5 Nov — | 6 Nov **Peer Reviews 2** | 7 Nov **Company Case Study Essay Plan** |
| 9 | 8 Nov | 9 Nov **Student Seminar** | 10 Nov — | 11 Nov — | 12 Nov — | 13 Nov — | 14 Nov |
| 10 | 15 Nov | 16 Nov **Student Seminar** | 17 Nov — | 18 Nov — | 19 Nov — | 20 Nov | 21 Nov **Company Case Study Report** |





# Assessment Calendar (ENG)

| Week | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| 1 | 13 Sep | 14 Sep | 15 Sep | 16 Sep | 17 Sep | 18 Sep | 19 Sep |
| 2 | 20 Sep | 21 Sep | 22 Sep | 23 Sep | 24 Sep | 25 Sep | 26 Sep |
| 3 | 27 Sep | 28 Sep **Project Pitch** | 29 Sep — | 30 Sep — | 1 Oct — | 2 Oct — | 3 Oct **Project Taskboard** |
| 4 | 4 Oct | 5 Oct **Labour Day** | 6 Oct **Student Seminar** | 7 Oct — | 8 Oct — | 9 Oct **Lecture Summaries 1** | 10 Oct |
| 5 | 11 Oct | 12 Oct **Student Seminar** | 13 Oct — | 14 Oct — | 15 Oct — | 16 Oct **Peer Reviews 1** | 17 Oct |
| 6 | 18 Oct **Flexibility Week** | 19 Oct Jul — | 20 Oct Jul — | 21 Oct Jul — | 22 Oct Jul — | 23 Oct Jul — | 24 Oct Jul — |
| 7 | 25 Oct | 26 Oct **Student Seminar** | 27 Oct — | 28 Oct — | 29 Oct — | 30 Oct **Lecture Summaries 2** | 31 Oct — |
| 8 | 1 Nov | 2 Nov | 3 Nov | 4 Nov | 5 Nov | 6 Nov **Peer Reviews 2** | 7 Nov **Company Case Study** |
| 9 | 8 Nov | 9 Nov | 10 Nov | 11 Nov | 12 Nov | 13 Nov **Project** | 14 Nov **Peer Assessment Project Diary** |
| 10 | 15 Nov | 16 Nov **Presentation** | 17 Nov — | 18 Nov — | 19 Nov — | 20 Nov — | 21 Nov |





# Course Improvements This Year

- Assessment Load
  - Fewer assessment items; more spread out
  - No debate, no final exam for COMP students
  - No participation, project plan document for ENG students
- Project Management
  - Moved to COMP3900 for COMP students
- Contact Hours
  - 12→8→6 lectures; 6→4 summaries; 8→7→3 seminars
- Company Case Study
  - 2000→1000→500-750 words for ENG students
- Course Forum
  - Questions and consultations





# Tips for a Successful Project

- Choose your team wisely!
  - One person cannot produce the expected work of a team of 5
- Make the project management tool work for you
  - The taskboard is the Oracle; you are **not** a slave to the tool
- Writing good user stories is hard
  - Not too big (bigger than a sprint) – hard to understand/estimate
  - Not too small (e.g. login) – too many stories clog up the taskboard
  - Breakdown into tasks and clear user acceptance tests/criteria
  - Product Owner should check user acceptance tests/criteria
  - Solicit early feedback from real users (i.e. not your team)
- You *can* use tools other than Pivotal Tracker, Jira
  - But they **must** have full agile planning and estimation

# Ethical Reasoning

- What should/ought I do? (Not: What do I do?)
  - Essentially takes into account the interests of others
- Two approaches to ethical reasoning
  - Based on rules/principles/duties/rights/obligations
    - Determine a principle and apply to the present case
  - Based on consequences
    - Examine consequences of each course of action and choose one that maximizes overall "utility"
- Ethical dilemmas – usually no clear right thing to do
- Emphasis is on making a judgement, taking responsibility, and justifying the choice on an ethical basis, **not** blindly following rules





# Example for Students/Academics

Java algorithm

Status: **Open** | Posted on: **13/05/2013**
Project budget: **$100.00**
Project ID: **43572**

PLACE BID | VIEW MESSAGE BOARD (13) | WATCH

Buyer cannot cover the payment on-site.

**Posted by:** mich
**Time remaining:** 3 day(s)
**Tags:** Java

**Project description:**

Please read the description here. easy money!!

http://www.cse.unsw.edu.au/~cs2911/13s1/assignments/ass02.html

PLACE BID | VIEW MESSAGE BOARD (13)

| Freelancer | Total | Deliver in | Date of bid | Rating |
|---|---|---|---|---|
| htt210 | $50.00 | 2 days | 15/05/2013 | Not rated |

Dear customer, this assignment is very easy and can be done very quickly. I'm very good at Data Structure and Algorithm and Object Oriented Programming so your assignment will be perfect.





# Why Do Software Projects Fail?

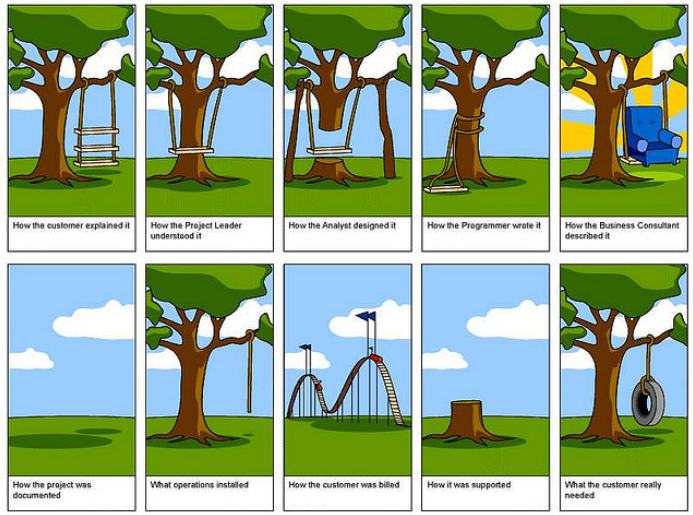
1. **Unclear requirements**: "Most people don't know what to build because they've never defined it. When they build the software, it fails because it doesn't meet people's needs."
2. **Overly optimistic and/or unrealistic schedules**: "People rush or skip things if the schedule isn't realistic. Also, companies are panicking due to the economy. They're compressing projects and schedules."
3. **Lack of user input**: This links back to requirements mistake #1. "Developers don't talk to people who are going to use the software."
4. **Lack of executive sponsorship and support**: "When management doesn't support and protect the project, it can often be undermined by internal politics and budget cuts."
5. **Turnover and layoffs**: "Projects often fail when key people leave the project early in its lifetime."

http://itknowledgeexchange.techtarget.com/software-quality/why-software-projects-fail-and-more-will-fail-in-2009/





# Common Understanding is Critical



# Spectacular Software Failure: Ariane 5

Ariane 5 Video

- Disintegration after 39 sec
- Caused by large correction for attitude deviation
- Caused by wrong data sent to On Board Computer
- Caused by software exception in Inertial Reference System after 36 sec





# Sequence of Events

| | |
|---|---|
| H0 = 093359 | 4th June 1996 |
| H0+7.5s | Ignition of solid booster stages and normal liftoff |
| Up to H0+37s | Flight guidance and trajectory normal. At this moment the velocity of the launcher was Mach 0.7 (807 kph) and its altitude 3,500m. |
| | Sudden swivelling of both solid booster nozzles up to the limit, recorded by telemetry. |
| H0+37s to H0+39s | This caused the launcher to tilt sharply, giving rise to intense aerodynamic loads on the launcher structure resulting in breakage. |
| | Following loss of the launcher integrity, destruction of all launcher elements by the onboard neutralisation system. |

http://www.esa.int/For_Media/Press_Releases/Flight_501_failure-_first_information





# Ariane 5 Flight Control System

- SRI (Inertial Reference System): Angles and velocities are calculated on the basis of information from a "strap-down" inertial platform, with laser gyros and accelerometers. The design of the Ariane 5 SRI is practically the same as that of an SRI used on Ariane 4.
- OBC (On-Board Computer): Executes the flight program and controls the nozzles of the solid boosters and the Vulcain cryogenic engine, via servovalves and hydraulic actuators
- Redundancy at equipment level: Two SRIs operating in parallel (identical hardware and software), two OBCs

http://www.di.unito.it/~damiani/ariane5rep.html





# Events Leading to Explosion

1. SRI-1 declares a failure due to a software exception; OBC switches to SRI-2
2. SRI-2 declares a failure due to a software exception; but OBC cannot switch back to SRI-1
3. Part of the data from SRI-2 does not contain proper flight data, but shows a diagnostic bit pattern of the computer of SRI-2
4. OBC commands full nozzle deflections of the solid boosters and the Vulcain main engine
5. Boosters separate from the main stage, in turn triggering the self-destruct system of the launcher!!

http://www.di.unito.it/~damiani/ariane5rep.html

## Article by James Gleick

*It took the European Space Agency 10 years and $7 billion to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.*

*All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a small computer program trying to stuff a 64-bit number into a 16-bit space.*

*One bug, one crash. Of all the careless lines of code recorded in the annals of computer science, this one may stand as the most devastatingly efficient. From interviews with rocketry experts and an analysis prepared for the space agency, a clear path from an arithmetic error to total destruction emerges.*

Gleick, J. *A Bug and a Crash*. http://www.around.com/ariane.html





## Code Analysis During Development

- Analysis of every operation which could give rise to an exception, including an Operand Error. Conversion of floating point values to integers analysed, and operations involving seven variables at risk of leading to an Operand Error.
- Protection added to four of the variables, evidence of which appears in the Ada code. The reason for the three remaining variables, including the one denoting horizontal bias, being unprotected was that further reasoning indicated that they were either physically limited or that there was a large margin of safety.
- The decision to protect certain variables but not others was taken jointly by project partners at several contractual levels.

http://www.di.unito.it/~damiani/ariane5rep.html





## Code Running Unnecessarily

- The original requirement accounting for the continued operation of the alignment software after lift-off was brought forward more than 10 years ago for the earlier models of Ariane, in order to cope with the rather unlikely event of a hold in the count-down.
- The period selected for this continued alignment operation, 50 seconds after the start of flight mode, was based on the time needed for the ground equipment to resume full control of the launcher in the event of a hold.
- This time sequence is based on a requirement of Ariane 4 and is not required for Ariane 5.

http://www.di.unito.it/~damiani/ariane5rep.html





## Code Reused from Ariane 4

- In Ariane 4 flights using the same type of inertial reference system, there had been no such failure because the trajectory during the first 40 seconds of flight was such that the particular variable related to horizontal velocity cannot reach, with an adequate operational margin, a value beyond the limit present in the software.
- Ariane 5 has a higher initial acceleration and a trajectory which leads to a build-up of horizontal velocity that is five times more rapid than for Ariane 4. The higher horizontal velocity of Ariane 5 generated – within the 40 second time frame – the excessive value that caused the inertial system computers to cease operation.

http://www.di.unito.it/~damiani/ariane5rep.html

# More Subtle Diagnosis of the Failure

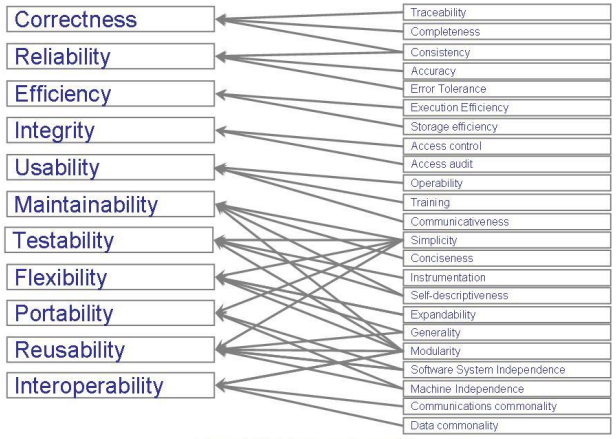
- Duplication of a component to provide a backup system is a standard design approach for **random hardware faults**
- Software failures do not have the same characteristics as hardware failures (so when one SRI failed, so did the other)
- The SRI was not tested prior to the Ariane 5 launch because the SRI was considered "fully qualified at equipment level"
- The SRI specification (a requirements document for the SRI) does not contain the Ariane 5 trajectory data as a functional requirement
- Any testing was done only with Ariane 4 trajectory data, but the Ariane 4 trajectory differs from that of the Ariane 5
- This is a **reuse specification error**; there is no precise specification of the Ariane 4 code; the requirement that the horizontal bias should fit in 16 bits was stated in an obscure part of a document, but not in the code itself!!

http://www.di.unito.it/~damiani/ariane5rep.html
Jézéquel, J.-M. & Meyer, B. *Design by Contract: The Lessons of Ariane.* Computer, **30**(1), 1997.





# What is Good Quality Software?

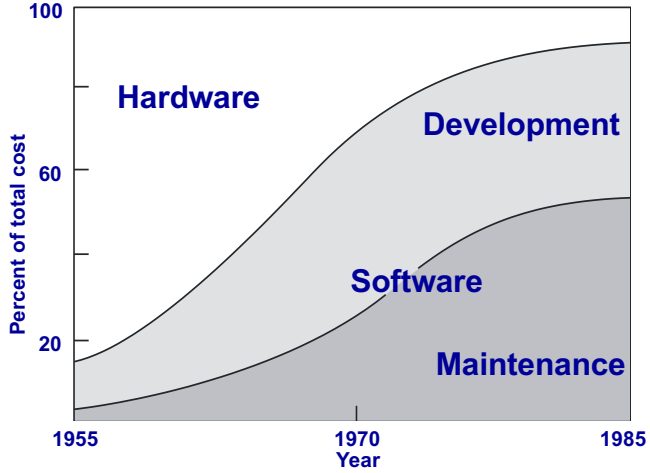



Pfleeger, S. & Atlee, J. *Software Engineering: Theory and Practice*. Fourth Edition. Pearson Education, 2010.





# Relative Distribution of Costs




van Vliet, H. *Software Engineering*. Third Edition. John Wiley & Sons, 2010.





# Why Do Software Projects Succeed?

1. Clear requirements and specifications
2. Clear objectives and goals
3. Realistic schedule
4. Effective project management skills/methodologies (project manager)
5. Support from top management
6. User/client involvement
7. Effective communication and feedback
8. Realistic budget
9. Skilled and sufficient staff
10. Frozen requirements
11. Familiarity with technology/development methodology
12. Proper planning
13. . . .

Nasir, M.H.N. & Sahibuddin, S. *Critical Success Factors for Software Projects: A Comparative Study.* Scientific Research and Essays, **6**(10), 2011.

# Standard View of Requirements

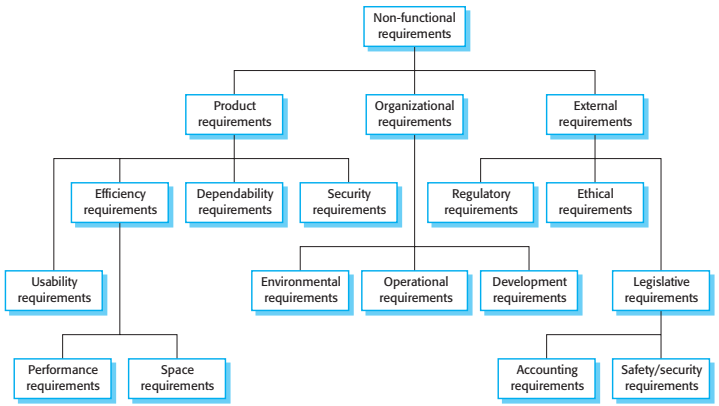
- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations
  - May state what the system should not do
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Types of Non-Functional Requirement




Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Non-Functional Requirement Metrics

| | |
|---|---|
| Speed | Processed transactions/second<br>User/event response time<br>Screen refresh time |
| Size | Mbytes<br>Number of ROM chips |
| Ease of use | Training time<br>Number of help frames |
| Reliability | Mean time to failure<br>Probability of unavailability<br>Rate of failure occurrence<br>Availability |
| Robustness | Time to restart after failure<br>Percentage of events causing failure<br>Probability of data corruption on failure |
| Portability | Percentage of target dependent statements<br>Number of target systems |

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Requirements Elicitation Problems

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders have conflicting requirements
- Organizational and political factors influence system requirements
- Requirements change during the analysis process
- New stakeholders emerge and the business environment changes

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.

# Requirements Changes

- Business and technical environment changes
  - New hardware, integration with different systems, change in business priorities (consequent changes in support), new legislation and regulations
- People who pay are usually not the users
  - Organizational and budgetary constraints conflict with end-user requirements and, after delivery, new features have to be added for user support
- Diverse user community with different requirements
  - Final system requirements are a compromise, the balance of support given to different user groups needs to change

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Software Project Management

- Project Management
  - Risk management, Managing people, Teamwork
- Project Planning
  - Software pricing, Plan-driven development, Scheduling
- Quality Management
  - Reviews and inspections, Measurement and metrics
- Configuration Management
  - Change management, Version management, Release management

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Traditional Project Plan

- Document Control: Version, History, Distribution
- Introduction: Purpose of Document, Glossary
- Project Definition
  - Background, Objectives, Benefits, KPIs, Scope, Assumptions and Dependencies, Deliverables
- Implementation Plan
  - High Level Schedule, Milestones, Approach, Quality Plan, Completion Criteria
- Roles and Responsibilities
- High Level Requirements
- Timeline for Execution and Dependencies
- Project Considerations
  - Risks, Project Documents, Naming Convention, Meetings & Reports
- Post-Implementation Review





# PERT Chart

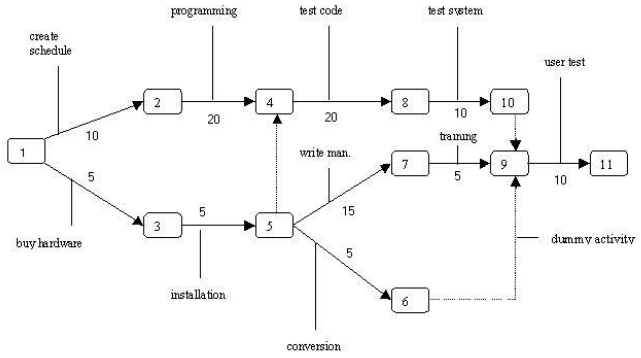
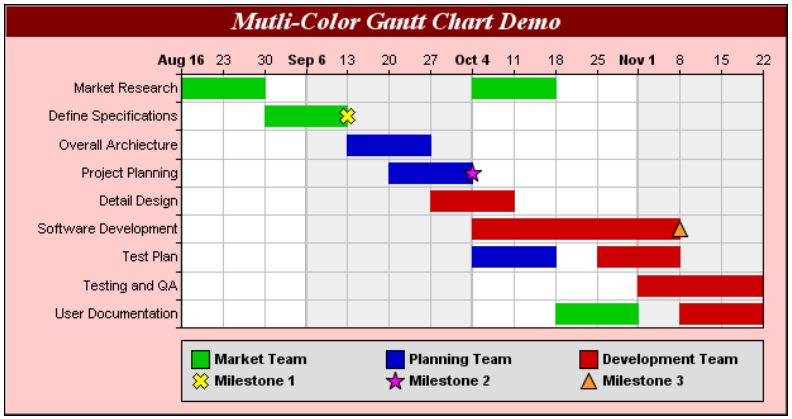



Fig. 1: PERT Chart

* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

http://searchsoftwarequality.techtarget.com/definition/PERT-chart

# Gantt Chart

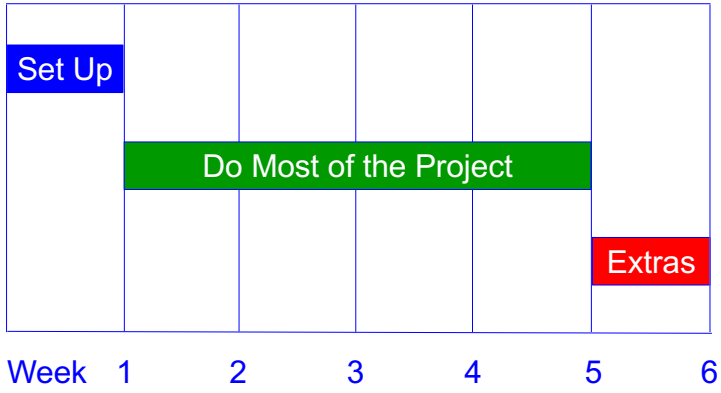



http://www.advsofteng.com/gallery_gantt.html





# How *Not* to Do a Project Plan



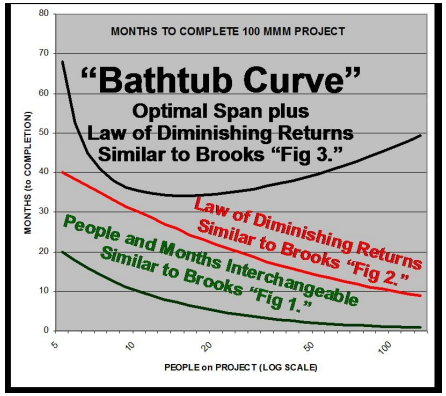

Thanks to past students ...





# Management Problem: Brooks' Law

- Adding people to a late software project makes it later




Brooks, F. *The Mythical Man Month*. Addison-Wesley,1975.
Image from http://iraknol.wordpress.com/





# Death March Projects: Reasons

- Politics, politics, politics
- Naive promises made by marketing, senior executives, naive project managers, etc.
- Naive optimism of youth: "We can do it over the weekend!"
- "Startup" mentality of fledgling, entrepreneurial companies
- "Marine Corps" mentality: *Real* programmers don't need sleep
- Intense competition caused by globalization of markets
- Intense competition caused by appearance of new technologies
- Intense pressure caused by unexpected government regulations
- Unexpected and/or unplanned crises – e.g. your hardware or software vendor just went bankrupt, or your three best programmers just died of Bubonic Plague

Yourdon, E. *Death March*. Prentice Hall, 1997.

## Teamwork

- What employers want most from our graduates . . .
- Teamwork is about
  - Adopting common goals
  - Management, planning and coordination
  - Communication and sharing information
  - Monitoring progress and helping out others
  - Putting the team first (ahead of individual interests)





## Team Organization

- Small software engineering groups are usually organized informally without a rigid structure
- For large projects, there may be a hierarchical structure where different groups are responsible for different sub-projects
- Agile development is always based around an informal group on the principle that formal structure inhibits information exchange

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





## Informal Teams

- The group acts as a whole and comes to a consensus on decisions affecting the system
- The group leader serves as the external interface of the group but does not allocate specific work items
- Rather, work is discussed by the group as a whole and tasks are allocated according to ability and experience
- This approach is successful for groups where all members are experienced and competent

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.

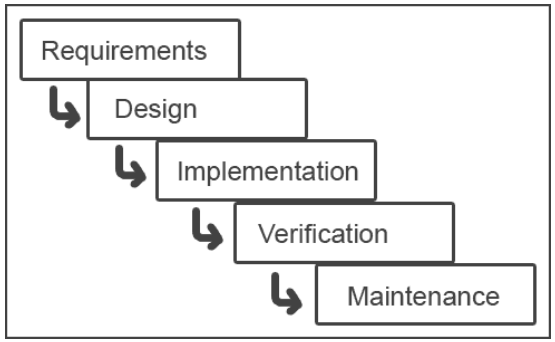




## Software Processes

- Plan-Driven Processes
  - Waterfall
  - V-model
  - Prototyping model
  - Incremental development
  - Boehm's spiral model
- Agile Processes
  - XP (Extreme Programming)
  - Scrum

# Waterfall Model (Idealized)

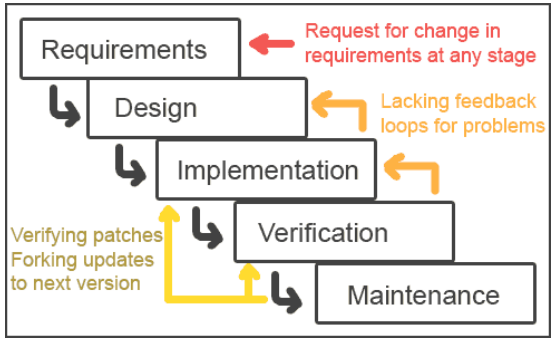



http://compsci.ca/blog/educational-flaws-programming-with-the-waterfall-model/





# Waterfall Model (Actual)

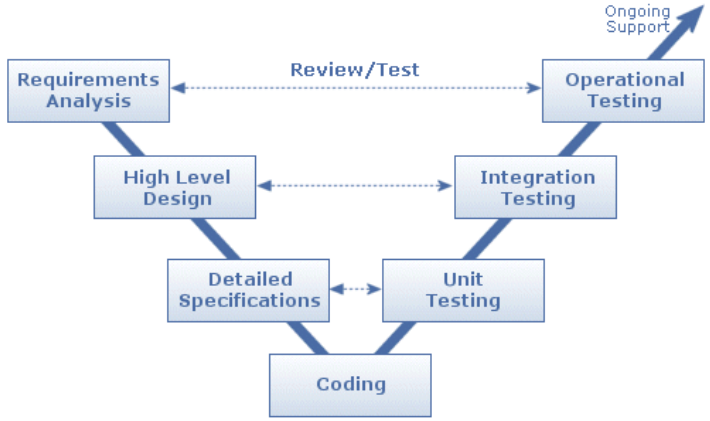



http://compsci.ca/blog/educational-flaws-programming-with-the-waterfall-model/





# V-Model

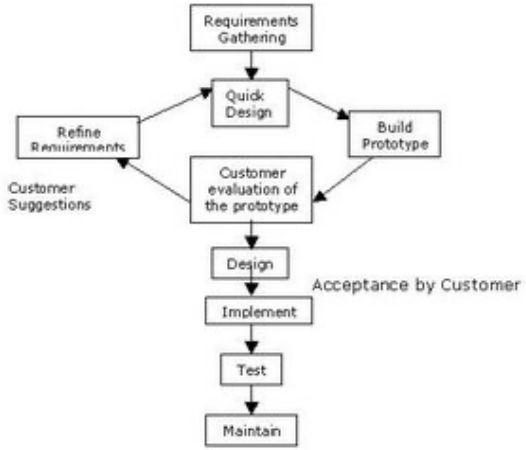



http://softwareandme.wordpress.com/2009/10/20/



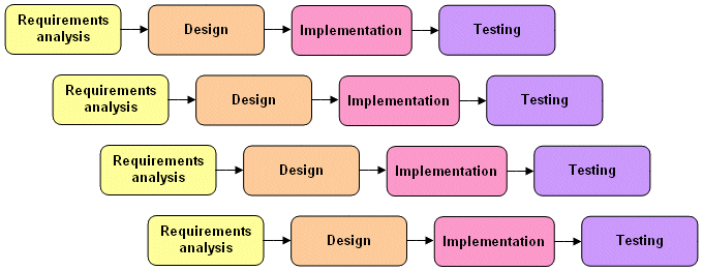


# Prototyping Model




http://rajeevprabhakaran.wordpress.com/2008/11/20/

# Incremental Development

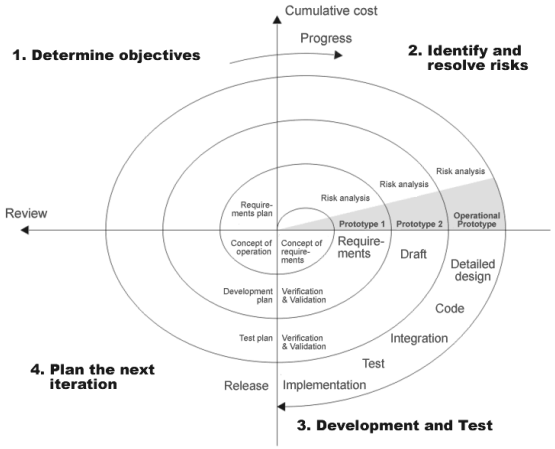



http://gkmaurya.blogspot.com.au/2011/04/





# Boehm's Spiral Model




http://leansoftwareengineering.com/2008/05/05/boehms-spiral-revisited/





# Agile Manifesto

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

http://www.agilemanifesto.org/





# Extreme Programming


http://software-document.blogspot.com.au/2011/06

# Extreme Programming Practices

| | |
|---|---|
| Incremental planning | Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. |
| Small releases | The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release. |
| Simple design | Enough design is carried out to meet the current requirements and no more. |
| Test-first development | An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented. |
| Refactoring | All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable. |

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Extreme Programming Practices

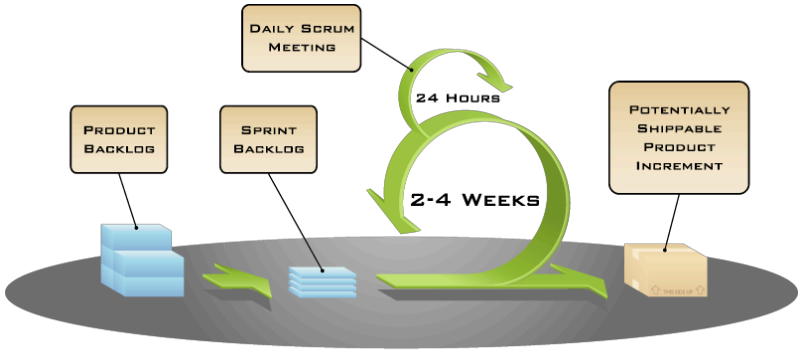
| | |
|---|---|
| Pair programming | Developers work in pairs, checking each other's work and providing the support to always do a good job. |
| Collective ownership | The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything. |
| Continuous integration | As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass. |
| Sustainable pace | Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity. |
| On-site customer | A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation. |

Sommerville, I. *Software Engineering.* Ninth Edition. Pearson Education, 2011.





# Scrum

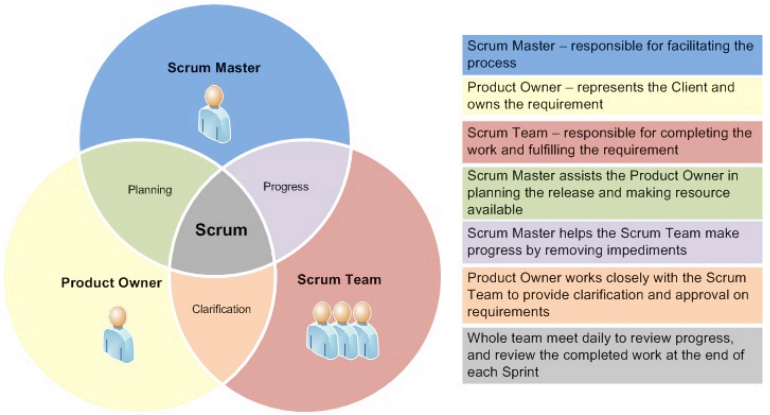



http://www.mountaingoatsoftware.com/scrum/overview





# Scrum Roles




http://scrum-stuff.blogspot.com.au/2010/08/scrum-roles.html