Project Planning

COSC345

Software Engineering

Assignment

- Is anyone not in a group?
- Are there any groups of 2 (or 5)?
- The assignment sheet specifies a ***minimum***
- Think about what else you should include (cool stuff)
  - There ***are*** marks for the cool stuff.

Overview

- Success and Failure
- Software management activities
- Project plans
- Scheduling
- Plotting progress
  - Gantt and Pert charts
- Risks and risk management

Success and Failure

- Good management does not guarantee success

- Bad management nearly guarantees failure
  - Late deliverables
  - Cost overruns
  - Requirements failures

- Every project needs a “champion”
  - Someone who will fight for the project

Software Management is Hard

- The product is intangible – so monitor the project
  - Progress on a bridge is visible (you can ***see*** that)
  - Progress on software is not (you cannot ***see*** it)
    - What does 80% finished mean for software?
      - Can a product ever be finished?

- With software there are no standard processes
  - Building a bridge is a standard process
  - Building software is different each time
    - Some standard tools that can help
    - Some “rules of thumb” that can help
  - Software projects are usually unique
    - Prior experience may not be helpful
    - Technology changes make knowledge obsolete
      - Symbian / iPhone / Android / Harmony OS?

Management Activities

- Requirements and proposal writing
  - Objectives, cost, and schedule estimates

- Project planning and scheduling
  - ***Activities***
  - ***Milestones***
    - Tangible achievements, e.g.:
      - Objects or libraries completed
      - Hardware needed for the project received
      - Fully staffed
  - ***Deliverables***
    - Finished (deliverable) pieces, e.g.:
      - Documents
      - Pieces of software

Management Activities

- Project costing
  - Hardware
  - Software (compilers, libraries, debuggers, etc)
  - Staff (including yourself)
  - Consumables
- Project monitoring and reviews
  - Used in problem prediction
- Staff turnover management
  - Staff quality usually determined by price
  - Quality staff many not be available!
  - Training
- Report writing
  - For clients and upper management

Project Planning

- Software development plan
  - How and when the software will be developed
  - Foreseeable problems and solutions (risks)
  - Constituent parts of the software
  - Prerequisites
    - Hardware dependencies (e.g. embedded systems)
    - Software dependencies (e.g. third party libraries)

- Quality plan
  - Readability
  - Maintainability
  - Efficiency

- Validation plan
  - How the software will be shown to be valid

Project Planning

- Maintenance plan
  - Predicts maintenance requirements and costs
    - Changing software after delivery
      - Repair of faults (bug-fixing)
      - Increasing functionality
      - Adaptation to new environments (configurations)
- Configuration management plan
  - Different versions
    - Different operating systems
    - Different pricing schemes (Windows 10 has twelve editions!)
      - Home / pro / enterprise / education / pro education / enterprise LTSB / mobile entertprise / mobile / IoT / S / Team / Pro for Workstations
- Staff development plan
  - How staff skills will be used and developed

Project Tracking

- Establish project constraints
  - Delivery date, budget, hardware, software, staff levels
- Assessment of parameters
  - Software design, size, interdependencies
- Define milestones and deliverables

- While (project continues)
  - Initiate new activities
  - Review progress (typically weekly or daily)
    - If (problem)
      - Review problem, initiate solution
  - Revise project constraints, parameters, and schedule
  - Renegotiate constraints and deliverables

Deliverable: The Project Plan

- Executive Overview
- Introduction
- Project description
- Resource requirements
  - Prices, schedules
- Organization
  - People and roles
- Project breakdown
  - Identifiable activities, milestones, and deliverables
- Risk analysis
  - Possible risks, and solutions
- Project schedule
  - Dependencies between activities
  - Time to milestones
  - Allocation of people to tasks
- Monitoring and reporting
  - How the project will be monitored
  - When reports are to be delivered
- Conclusion

The Project Plan

- Must look good, read well, and be accurate
  - Presentation makes a big difference
  - Accurate project planning is vital
- Managers need information to manage
  - Software is intangible
  - Reports and deliverables are the only way to manage
  - Cost estimates and schedules must be kept up-to-date
- Milestones and deliverables
  - Must be concrete (not virtual or unverifiable)
  - Deliverables many consist of many milestones

Scheduling

- Necessary time and resources
  - Previous estimates are uncertain because
    - This project is unique
    - Different languages / OS / design methods may be used
- Usually optimistic
  - Even if not then the slack gets wasted
- Use management tools
  - Microsoft Project, Google Sheets for Project Management, etc.
  - Keep it up-to-date

Scheduling

- Divide the project into pieces and estimate each
  - Don’t make tasks too small (a week)
  - Don’t make tasks too large (8-10 weeks)
  - Many tasks might be done in parallel
  - Identify dependencies between tasks
  - Assume problems will occur
    - Mechanical failure, staff turnover, bad estimates, resource unavailability
    - Estimate as if no errors will occur, add contingency (50-100%)
  - Allow for staff issues (holidays, illness, personal problems)
  - Allow for dependencies on others (delivery of goods)
  - Include all schedulable resources (disk, CPU, people)

Estimation Rules of Thumb

- 1/3 (33%) Project planning
- 1/6 (16%) Coding
- 1/4 (25%) Component testing
- 1/4 (25%) System testing

- Estimate program size
  - Lines Of Code (LOC) or thousands of LOC (KLOC)
  - Industry output about 1000 LOC per developer per year
    - Working, used, and documented lines of code
  - About 240 working days per year
  - “garage developers” don’t write commercial software

LOCs

- Estimates based on:
  - Whole program size
  - Sum of the functional unit sizes

- Useful for:
  - Error rate estimation
  - Productivity rate estimation

- But dependant on:
  - Programmer style
  - Programming language

- Biggest problem:
  - Can only be known once the program is finished!

Basic COCOMO

- Constructive Cost Model (1981)

- Project Types
  - Simple projects
    - Well understood applications, small teams
  - Moderate projects
    - More complex, limited experience
  - Embedded projects
    - Complex, strongly coupled to hardware, software, regulations, etc.

- Metrics based on statistics drawn from a large number of software projects

Basic COCOMO

- Estimates:
  - Effort = a(KLOC)^b
  - Time = c(Effort)^d months
  - People = Effort / Time

	a	b	c	d
Simple	2.4	1.05	2.5	0.38
Moderate	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

- Good for quick estimation, but does not consider hardware constraints, programmer skill, or modern tools
  - Intermediate COCOMO (81) and COCOMO II (1997)
  - Not enough time to discuss these in detail in class
    - See INFO310 or the text book for more details

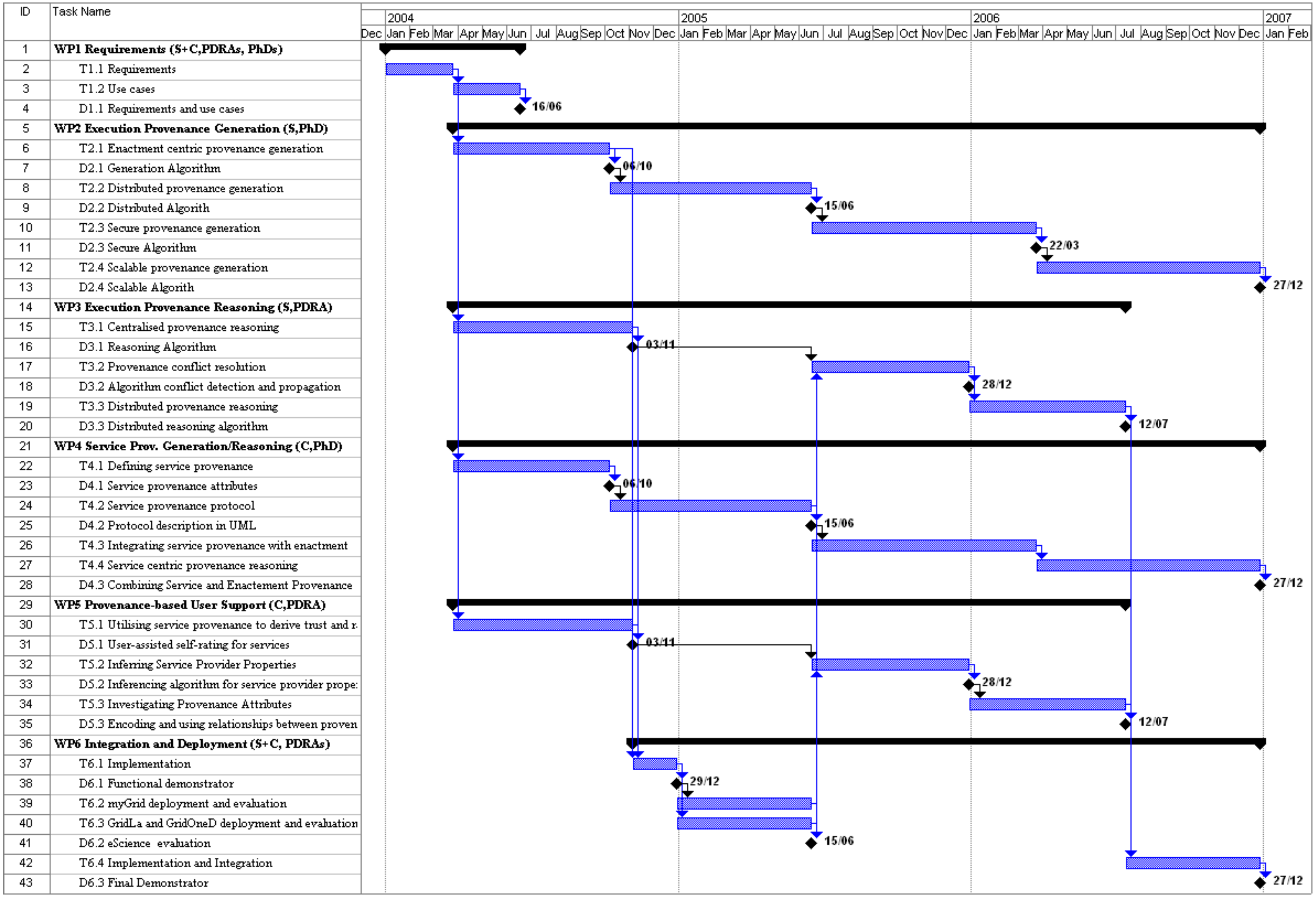
Functional Point Analysis (FPA)

- Size based on ***user perceived functionality***
  - Language and style independent
- Define the functional requirements
  - Categorise:
    - Outputs, inquiries, inputs, internal files, and external interfaces
  - Define complexity and assign some functional points
    - Productivity is measures in points implemented per month
  - We now have a cost based on user's requirements
    - And so can modify the "spec" based on cost
- International Standards include:
  - Common Software Measurement International Convention (COSMIC)
- Problems
  - Does not deal with algorithmic complexity or effort
  - Complexity estimates are estimator dependant
  - Biased towards data processing systems (because of the categories)

Gantt Charts

- Invented in 1917 by Charles Gantt
- Focus on tasks needed to complete a project
- Each task represented by horizontal (time) bar
  - Length of the bar represents length of task
- Arrows connecting tasks represent dependencies
- Diamonds are milestones and deliverables
- Come in many different forms
  - Often tool dependent
- Software will often identify critical paths

Gantt Chart



ID	Task Name
1	**WP1 Requirements (S+C,PDRAs, PhDs)**
2	T1.1 Requirements
3	T1.2 Use cases
4	D1.1 Requirements and use cases
5	**WP2 Execution Provenance Generation (S,PhD)**
6	T2.1 Enactment centric provenance generation
7	D2.1 Generation Algorithm
8	T2.2 Distributed provenance generation
9	D2.2 Distributed Algorith
10	T2.3 Secure provenance generation
11	D2.3 Secure Algorithm
12	T2.4 Scalable provenance generation
13	D2.4 Scalable Algorith
14	**WP3 Execution Provenance Reasoning (S,PDRA)**
15	T3.1 Centralised provenance reasoning
16	D3.1 Reasoning Algorithm
17	T3.2 Provenance conflict resolution
18	D3.2 Algorithm conflict detection and propagation
19	T3.3 Distributed provenance reasoning
20	D3.3 Distributed reasoning algorithm
21	**WP4 Service Prov. Generation/Reasoning (C,PhD)**
22	T4.1 Defining service provenance
23	D4.1 Service provenance attributes
24	T4.2 Service provenance protocol
25	D4.2 Protocol description in UML
26	T4.3 Integrating service provenance with enactment
27	T4.4 Service centric provenance reasoning
28	D4.3 Combining Service and Enactement Provenance
29	**WP5 Provenance-based User Support (C,PDRA)**
30	T5.1 Utilising service provenance to derive trust and r
31	D5.1 User-assisted self-rating for services
32	T5.2 Inferring Service Provider Properties
33	D5.2 Inferencing algorithm for service provider prope:
34	T5.3 Investigating Provenance Attributes
35	D5.3 Encoding and using relationships between proven
36	**WP6 Integration and Deployment (S+C, PDRAs)**
37	T6.1 Implementation
38	D6.1 Functional demonstrator
39	T6.2 myGrid deployment and evaluation
40	T6.3 GridLa and GridOneD deployment and evaluation
41	D6.2 eScience evaluation
42	T6.4 Implementation and Integration
43	D6.3 Final Demonstrator

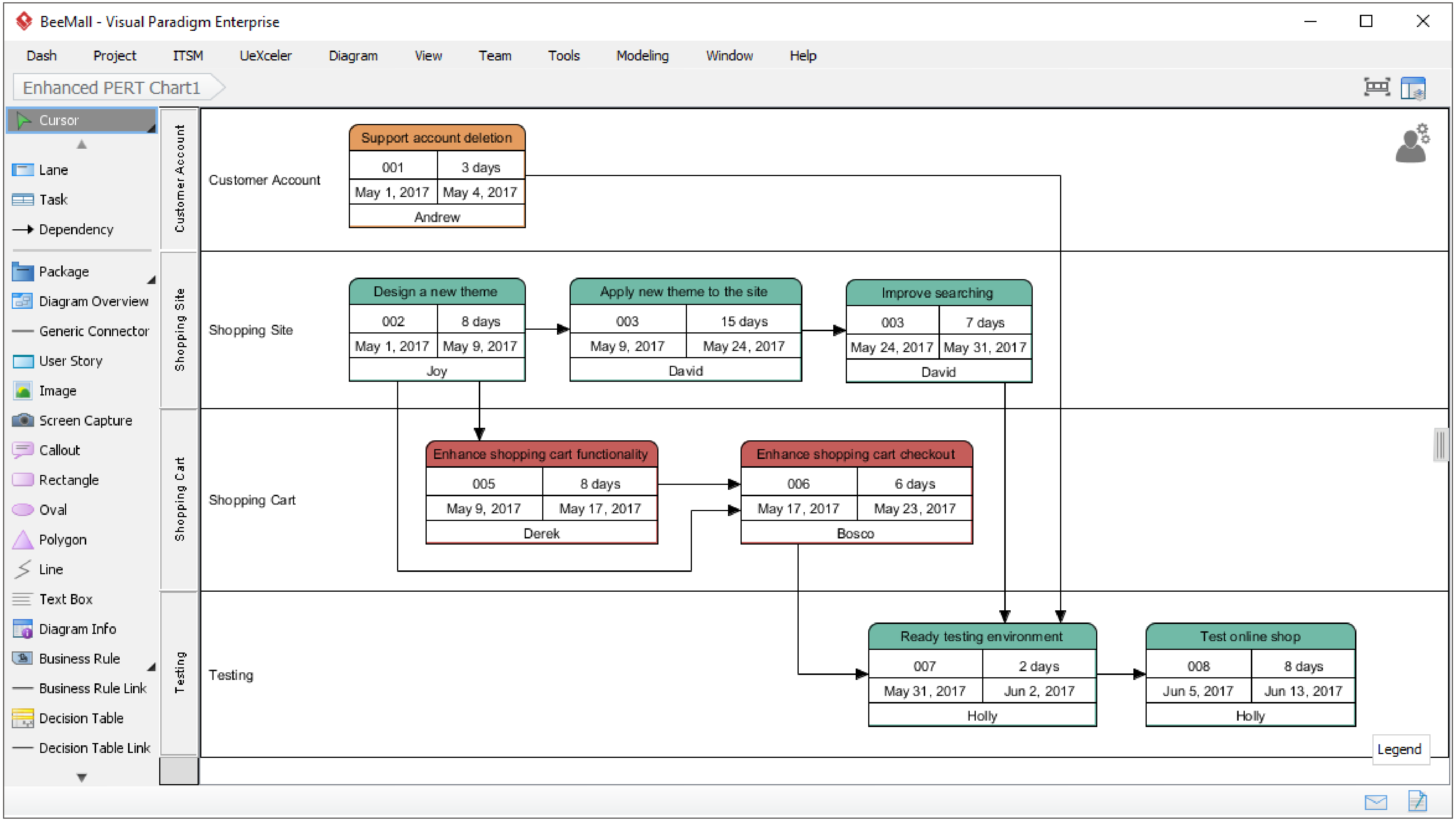
2004
2005
2006
2007
Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb
16/06
06/10
15/06
22/03
27/12
03/11
28/12
12/07
06/10
15/06
27/12
03/11
28/12
12/07
29/12
15/06
27/12

From: http://www.pasoa.org/case_files/pasoa-workplan.gif

PERT Charts

- Invented in the 1950s by the US Navy
- Completed tasks crossed out
- Partially completed tasks slashed out
- Details of task shown in box
- The critical path is highlighted

Pert Chart



Customer Account

Support account deletion
001 | 3 days
May 1, 2017 | May 4, 2017
Andrew

Shopping Site

Design a new theme
002 | 8 days
May 1, 2017 | May 9, 2017
Joy

Apply new theme to the site
003 | 15 days
May 9, 2017 | May 24, 2017
David

Improve searching
003 | 7 days
May 24, 2017 | May 31, 2017
David

Shopping Cart

Enhance shopping cart functionality
005 | 8 days
May 9, 2017 | May 17, 2017
Derek

Enhance shopping cart checkout
006 | 6 days
May 17, 2017 | May 23, 2017
Bosco

Testing

Ready testing environment
007 | 2 days
May 31, 2017 | Jun 2, 2017
Holly

Test online shop
008 | 8 days
Jun 5, 2017 | Jun 13, 2017
Holly

From: https://www.visual-paradigm.com/features/project-management-diagrams/enhanced-pert-chart/

Risks

- Project risks
  - Staff and management turnover
  - Hardware and other dependency (including spec) availability
  - Requirements change

- Product risks
  - Failure of third party tools (bad libraries, etc.)
  - Project size underestimation
  - Requirements change

- Business risks
  - Technology changes (e.g. introduction of smart watches)
  - Competitors introducing similar product
  - Requirements change

Risk Management

- Project managers
  - Anticipate risks
  - Take actions to avoid risks
  - Develop solutions to risks

- Project plan
  - Include risk analysis
  - Consequence of risks occurring
  - Cost of avoiding / fixing consequence
  - Contingency plans

- This is a continuous process

Risk Management

- Risk identification
  - Team exercise often through brainstorming

- Look for
  - Technology risks
    - Will technology change (e.g. popularity of Windows vs. MacOS)
  - People risks
    - Staff leaving / holidays / parental leave
  - Organization risks
    - Takeovers / mergers / management changes / corporate focus
  - Tools risks
    - Will the tools live up to requirements (software and hardware)
  - Requirements risks
  - Estimation risks
  - Society risks
    - Coronavirus in 2020

Risk Management

- Risk analysis
  - For each risk
    - Judge probability of occurring (low, middle, high)
    - Judge cost of recovery (catastrophic, serious, tolerable, insignificant)
- Risk planning
  - Plan for avoidance
  - Plan for impact minimization
  - Have contingency plans
- Risk monitoring
  - Regularly re-assess each risk

References

- F. Brooks, *Mythical Man Month*, Chapter 2

- I. Sommerville, *Software Engineering*, Chapter 5