Aston University
BIRMINGHAM UK

Software Project Management

Unit 6: Project Monitoring & Control (plus) Quality Management

Thais Webber
Richard Lee

Unit Objectives

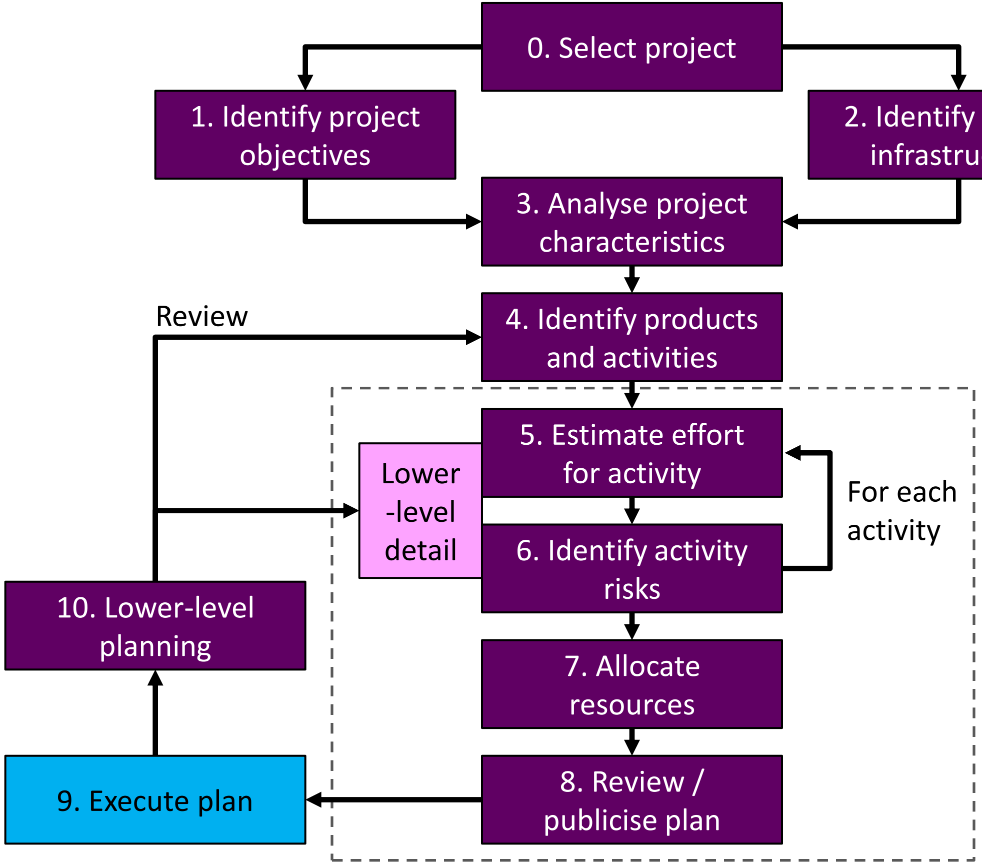
When you have completed this unit, you will be able to:

- Monitor project progress and assess the risk of slippage
- Visualise and assess the state of the project, revising targets as necessary to address drift
- Control changes to project requirements
- Quality management importance and standards in software projects

Outline

- Introduction
- Monitoring project progress
- Monitoring cost
- Getting the project back on target
- Quality management importance and standards

Step Wise


0. Select project
1. Identify project objectives
2. Identify project infrastructure
3. Analyse project characteristics
Review
4. Identify products and activities
5. Estimate effort for activity
Lower -level detail
For each activity
6. Identify activity risks
10. Lower-level planning
7. Allocate resources
8. Review / publicise plan
9. Execute plan

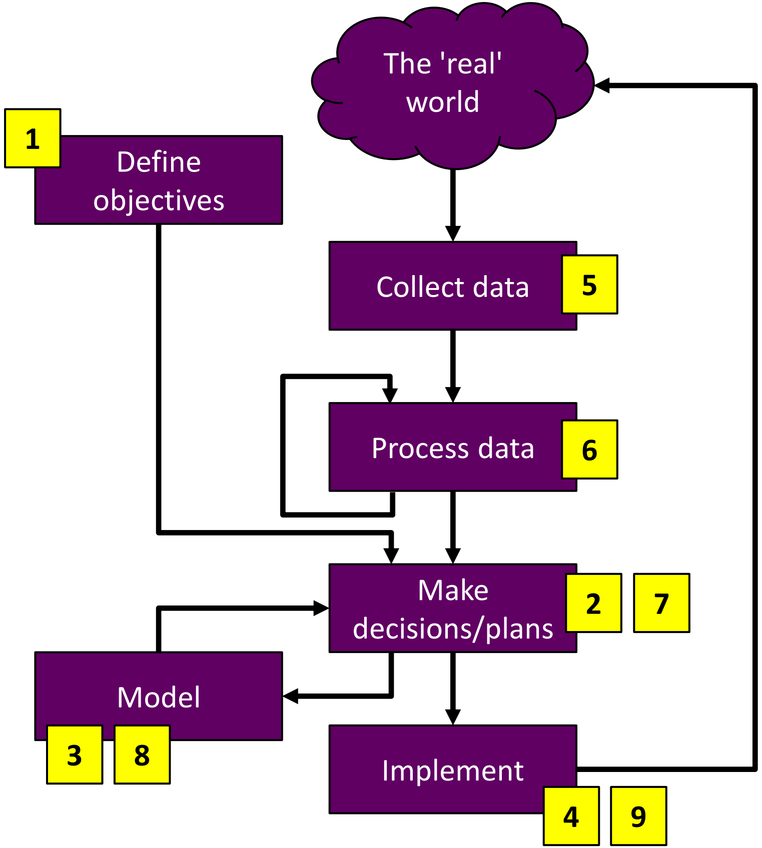
Motivations

- To ensure that the project progresses according to the plan
  - monitor what is happening and compare actual achievement against the schedule
  - may need to revise schedule to bring the project back on target as soon as possible

  Have you done this during the progress of your Final Year Project?

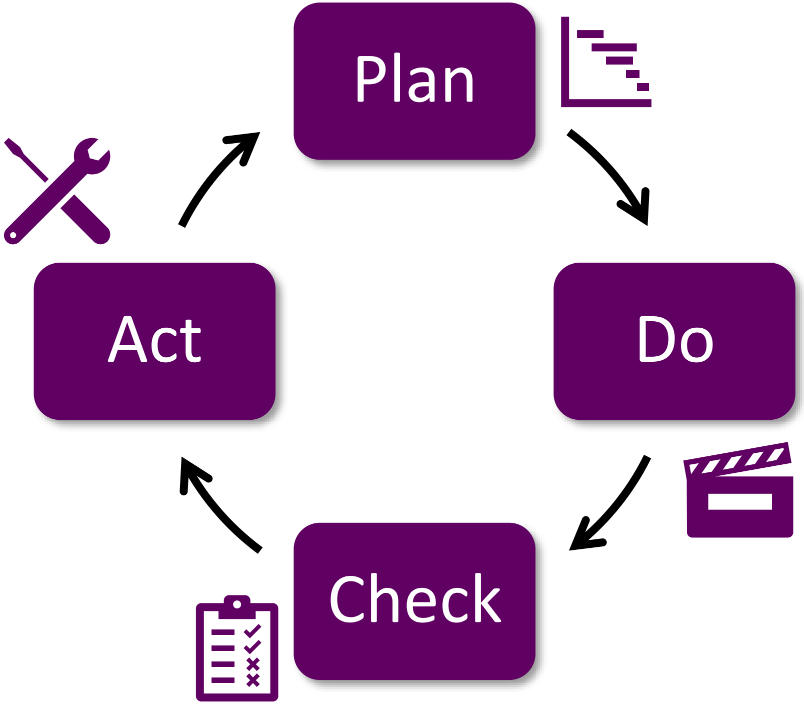
- To keep track of changes imposed from outside (e.g., changes in requirements)

A project control cycle



An alternative view

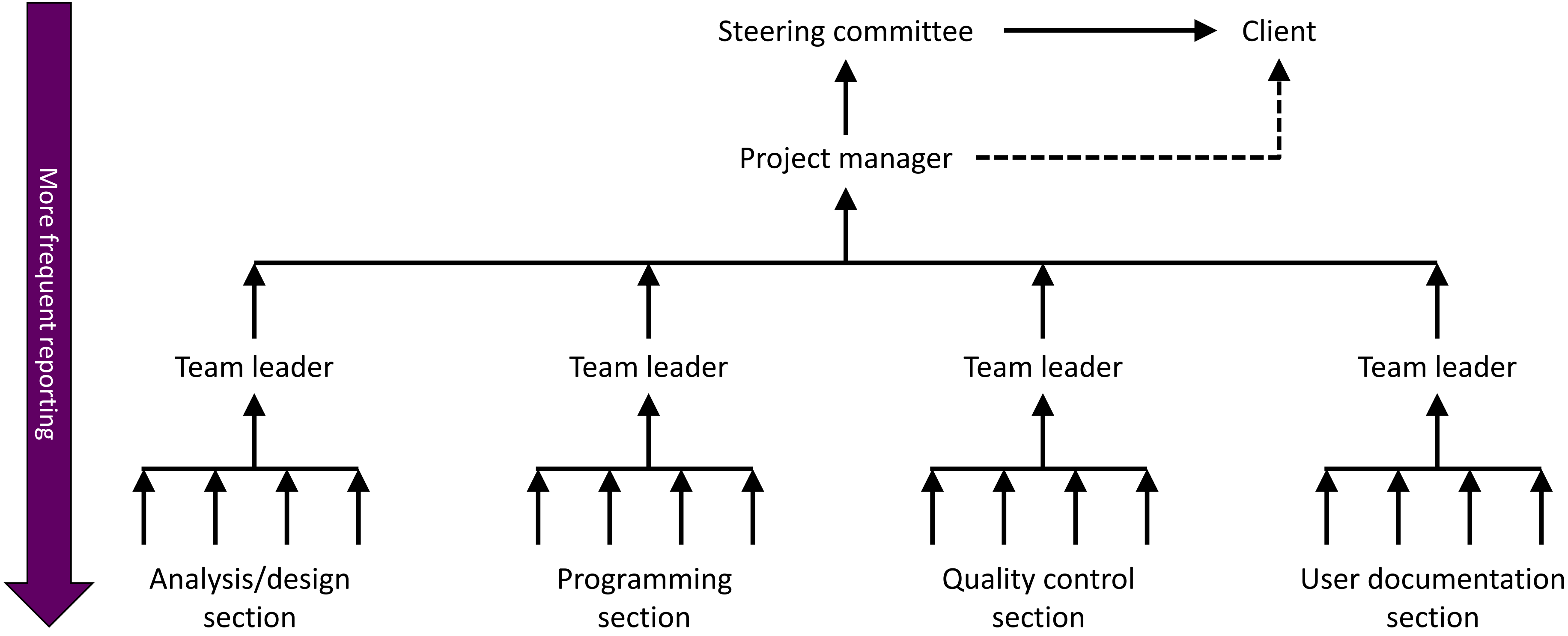
PDCA cycle



Responsibilities

- The overall responsibility for ensuring satisfactory progress on a project belongs to the Project Board
  - alternative names include project steering committee and project management board
- Day-to-day responsibility is the role of the Project Manager
  - aspects of this can be delegated to team leaders
- A reporting hierarchy that help teams stay aligned

Reporting frequency across the hierarchy



An alternative view (Scrum)

- Beyond hierarchies
- Focus on collaboration

An alternative view (Scrum)

Product Owner	value-based decisions
Scrum Master	supports the team embracing principles
Development Team	works collaboratively as one unit

Outline

- Introduction
- Monitoring project progress
- Monitoring cost
- Getting the project back on target
- Quality management importance and standards

Assessing progress

- A **checkpoint** is a point in the project where progress is checked

- **Two types of checkpoint:**
  - **Event-driven:** check takes place when a particular event has been achieved (e.g., at the end of project stages in PRINCE2)
  - **Time-driven:** date of the check is pre-determined (e.g., between sprints in Scrum)

- Checkpoints ensure that intermediate products are compatible. They support project monitoring.

- FYP checkpoints: e.g., the proposal form and the first report submission

Collecting progress details: challenges

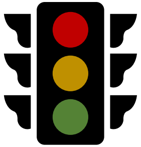
- Dealing with partial completions is a problem

- Estimates are affected by the **99% completion syndrome**
  - developer reports that the task is 25%, 50% and 75% complete at the end of weeks 1, 2 and 3, respectively
  - however, at the end of week 4 it is reported that the task is 99% complete

- Possible solutions are based on **objective verification of sub-task completion**:
  - control of products, not activities
  - subdivide into small sub-activities that each generates a product

Red / Amber / Green (RAG) reporting

- **Traffic-light method steps:**
  - identify key tasks
  - break down into sub-tasks; assess subtasks as:



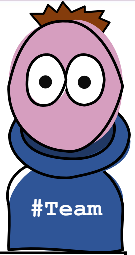
    - **Red: not on target and recoverable only with difficulty**
    - **Amber: not on target but recoverable**
    - **Green: on target**
  - **status of critical tasks is particularly important**

- However...
  - Involves estimates based on imprecise data
  - Imposes discrete classifications on a potentially more nuanced situation
  - Takes time

RAG example

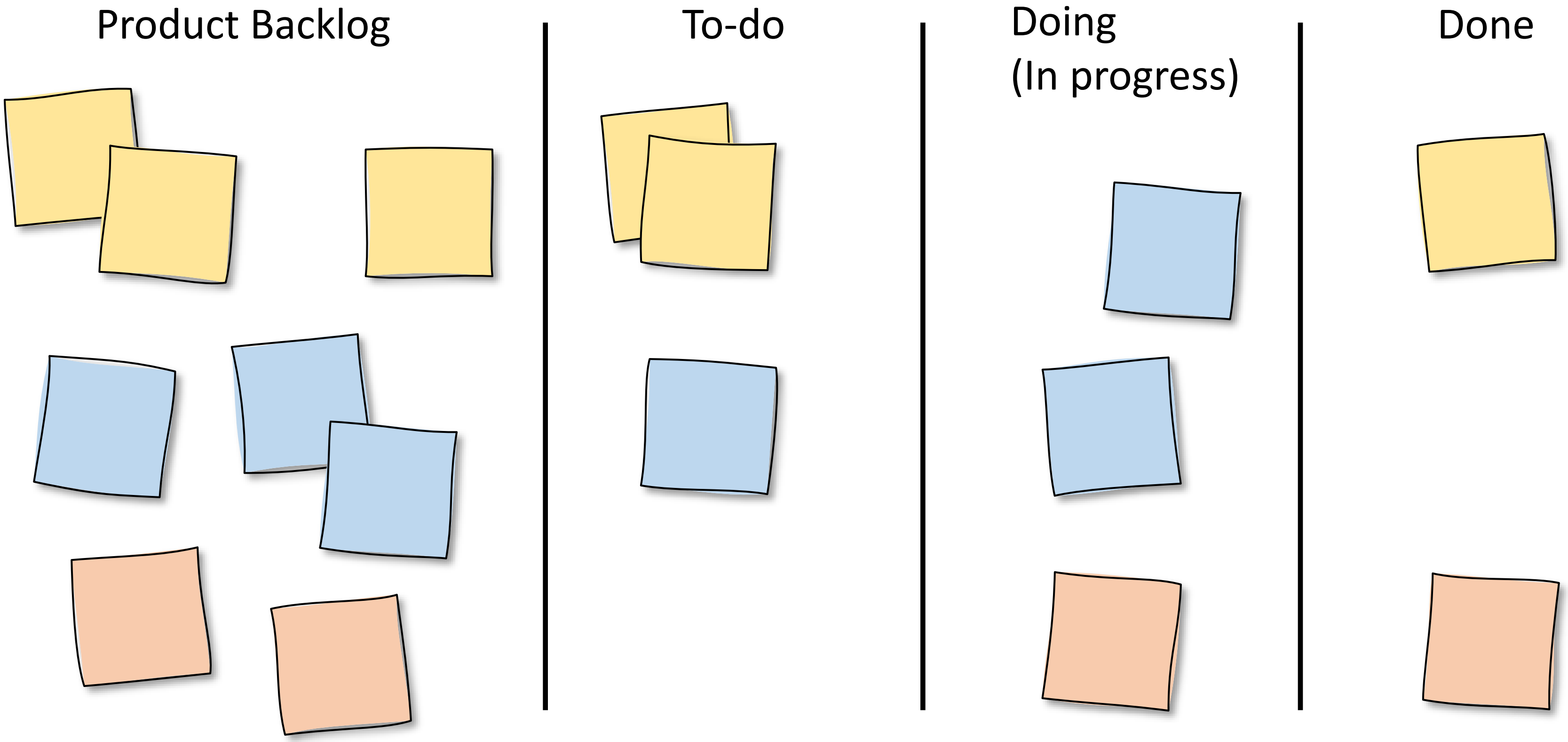
Staff: Digital Rick

Activity: Code & test module C



Week number:	13	14	15	16	17	18	
Activity summary	G	A	A	R			Major issue in update script
Component							**Comments**
Unit tests	G	A	A	G			No issues — tests stable.
File update procedures	G	G	A	R			The file update script failing on large files — needs fix!
Refactoring	G	G	G	A			Refactoring uncovered hidden bug — fixing now.
Compilation	G	G	G	A			Build warnings increasing — must review.
Documentation	G	G	G	A			Draft written, but examples still missing.

An alternative view (Scrum/Kanban)



Product Backlog

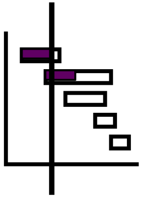
To-do

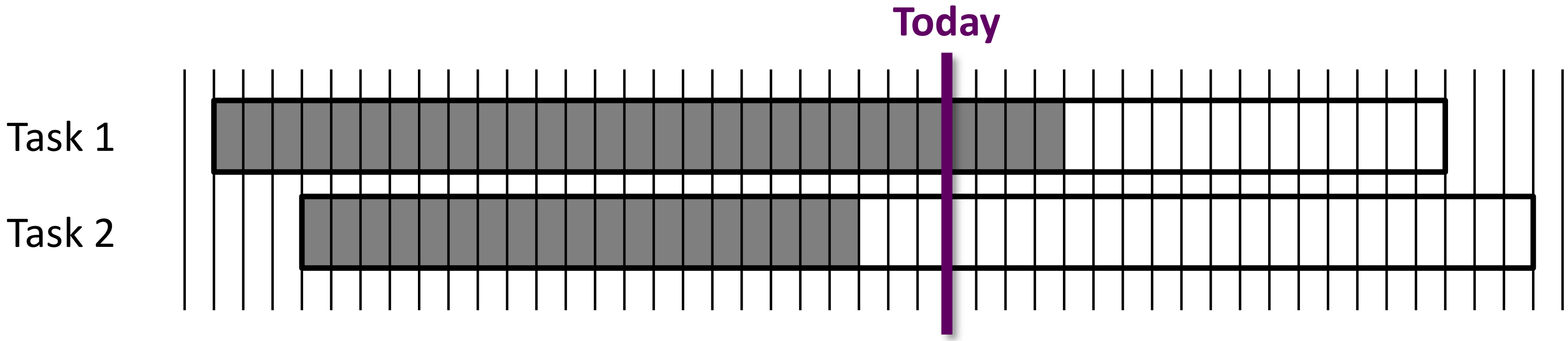
Doing
(In progress)

Done

Using Gantt charts to track project progress

- Gantt chart containing the project schedule from the initial project planning is now updated and extended:
  - by shading activity bars to reflect **reported progress**
  - by adding a "today cursor" to indicate **expected progress**

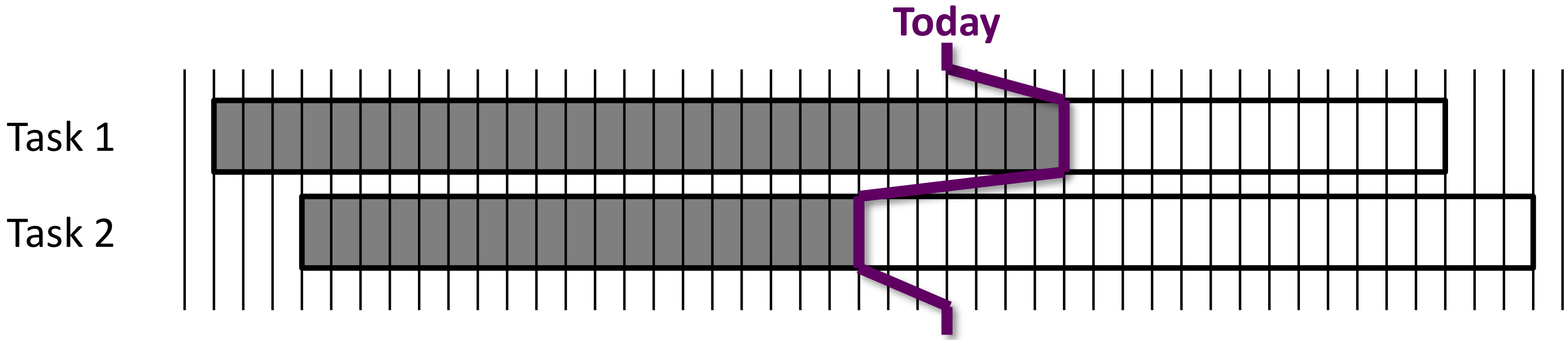




Task 1 is ahead of schedule; task 2 is behind schedule

Using slip charts to track project progress

- A Gantt chart variant with a **bent 'today' cursor** to indicate activity positions
- The more jagged the line, the more inconsistent the progress
- In this event, resources might be reallocated from task 1 to task 2



Limitations of Gantt / slip charts

- Neither shows clearly the **slippage of the project completion date** through the life of the project

- They do not allow the analysis of the **project trends**:
  - We cannot see how delays are growing or improving
  - We cannot see if things are getting better or worse after we make changes

- This information is essential in deciding what action to take:
  - In case a delay comes from **productivity consistently low** (e.g., each week of work from the original plan takes a week and a half), the final completion date will keep slipping unless appropriate action is taken in the project.
  - In case a delay is a **one-off event**, a small fix — like adding a team member briefly — may be enough.

Outline

- Introduction
- Monitoring project progress
- Monitoring cost
- Getting the project back on target
- Quality importance and standards

Cost monitoring

- Cost monitoring is an important component of project control, but also as an indicator of the effort that has gone into the project:
  - A project could be late because the staff originally committed have not been deployed – in this case the project will be behind time but under budget
  - A project could be on time but only because additional resources have been added and so be over budget
- There is a need for the monitoring of both achievements and costs

Earned Value Approach

- A popular approach to **cost monitoring**
  - Proposed by the US Department of Defence for their contractors
- Each task is assigned a ‘**value**’ based on the original cost estimates
  - Often as financial cost, but could be effort - e.g., (person days)
  - This attribute of a task is termed the task **planned value**
- Tasks are also assigned an **earned value** that is calculated based on the state of the task:
  - A task on which work has not started has an earned value of zero
  - A completed task has an earned value equal to the planned value
  - **The earned value of a project is the sum of the earned values for all its tasks** – a quantity that increases over the lifetime of the project

Partly-completed tasks

- The **0/100** technique
  - Earned value of task is zero until completion, and 100% of the planned value on completion
- The **50/50** technique
  - Half of the planned value allocated at start, the other half on completion
- The **milestone** technique
  - Earned value of task is the planned value of last milestone achieved
- The **percentage** technique
  - Works only **if** there is a way of assessing the percentage of the task that was completed (e.g., what percentage of 500 data records have been manually typed into a database)

Exception planning

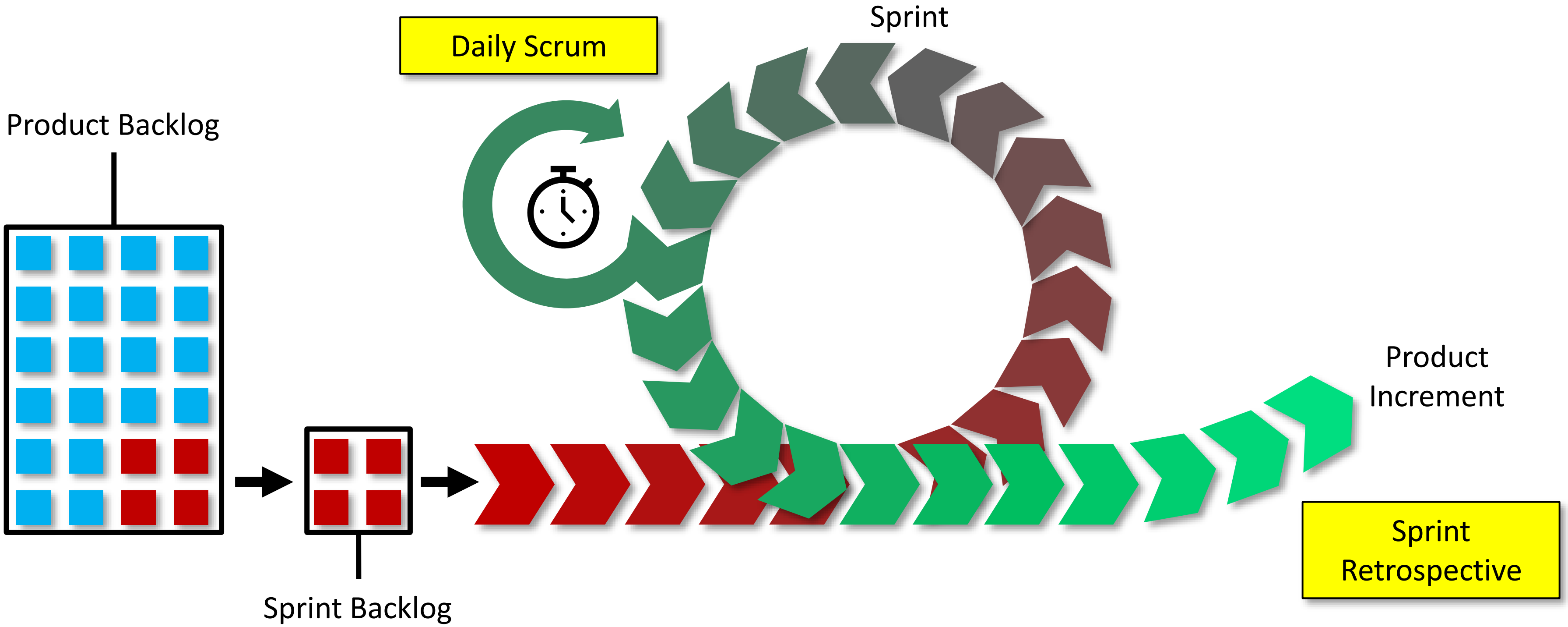
- Project manager typically allowed to change project plan as long as the agreed project outcomes are produced on time and within budget

- But changes to delivery date, scope and/or cost of project can affect:
  - Users (e.g., reductions in the scope of the project)
  - The business case (e.g. costs increase reducing the potential profits of delivered software product)

- In these cases, an **exception report** is needed (unless using Scrum)
  - Written by the project manager to explain the reasons that justify such a deviation from the existing plan

An alternative view (Scrum)

* Scrum does not use formal exception reports



* issues are handled through sprint reviews, replanning, and backlog adjustments

Prioritising monitoring

- Monitoring takes time and uses resources
  - Should be applied with different levels of detail/effort

- We want to focus more on monitoring certain types of activity:
  - Critical path activities
  - Activities with no free float – if delayed later dependent activities (although not the whole project) are delayed
  - Activities with less than a specified amount of float
  - High-risk activities
  - Activities using critical resources

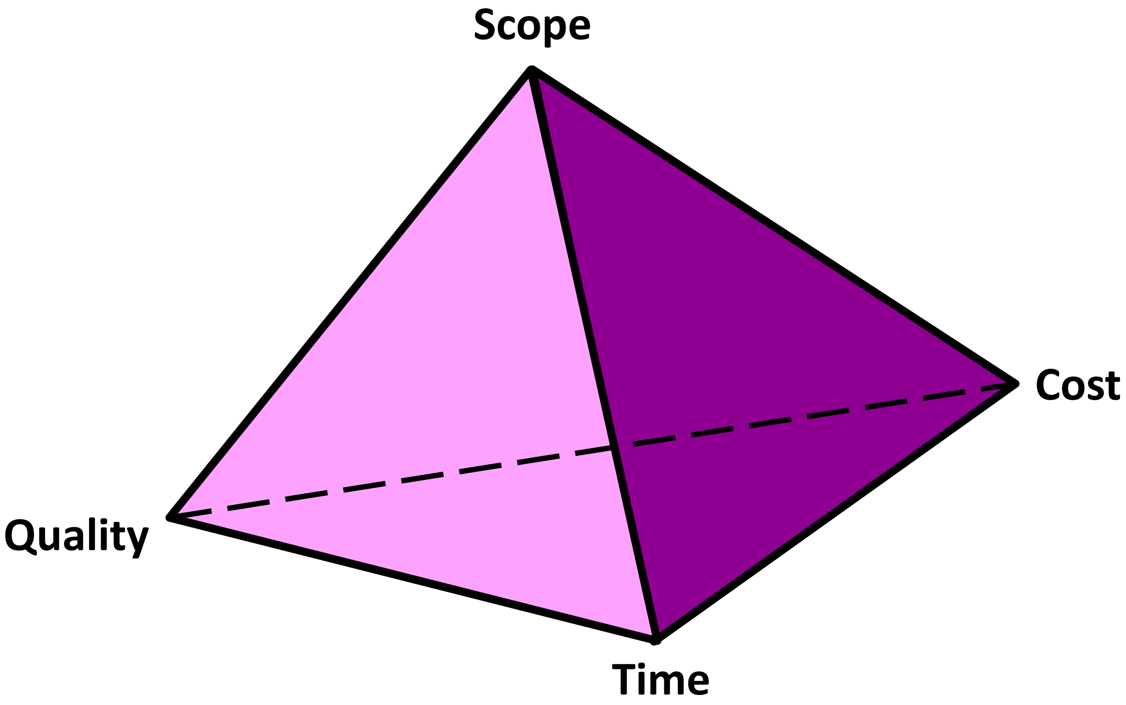
Outline

- Introduction
- Monitoring project progress
- Monitoring cost
- Getting the project back on target
- Quality management importance and standards

Getting back on track: options

- Most projects experience – at one time or another – **delays and unexpected events**
  - Monitoring project progress identifies these events
- **Several options for mitigation are available**
  - Renegotiate the deadline
  - Shorten critical path
  - Reconsider activity dependencies:
    - overlap the activities so that the start of one activity does not have to wait for completion of another – this may have a negative impact on quality
    - split activities

Getting back on track: balancing factors



Summary key points in Monitoring & Control

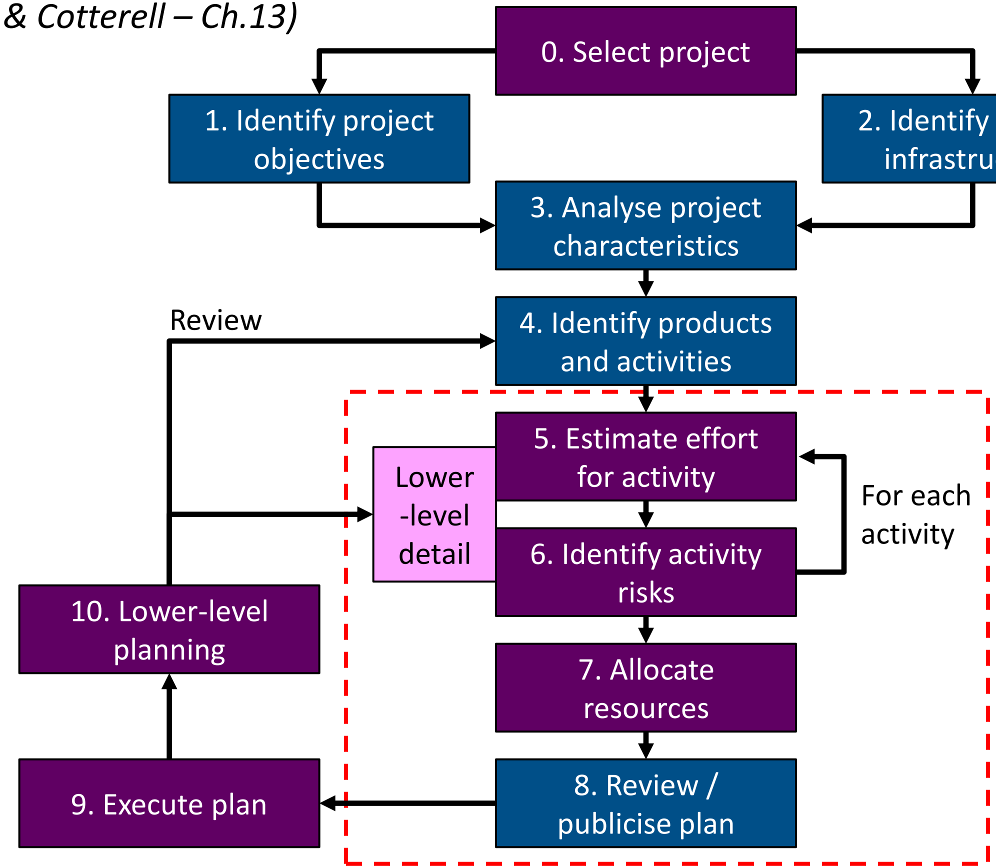
- **Monitoring compares actual progress with the plan** to detect delays early
- **Reporting hierarchy** ensures clear communication from team members
- **Checkpoints** (event-driven and time-driven) help track progress at key moments
- **Percentage** completion can be misleading, especially near the end of tasks
- **RAG reports** give simple status but lack trend detail
- **Agile boards** show task-level flow but not overall project health
- **Gantt and slip charts** show schedule status but not trends over time
- **Cost monitoring** shows the relationship between effort, schedule, and spending
- **Earned value techniques** provide objective measures of progress (PV, EV)
- **Exception planning** escalates changes to time, cost, or scope
- **Monitoring effort should vary** depending on float, risk, and criticality

Outline

- Introduction
- Monitoring project progress
- Monitoring cost
- Getting the project back on target
- Quality management importance and standards

Quality in the 'Step Wise' framework

(Book SPM Hughes & Cotterell – Ch.13)


0. Select project
1. Identify project objectives
2. Identify project infrastructure
3. Analyse project characteristics
Review
4. Identify products and activities
5. Estimate effort for activity
Lower -level detail
For each activity
6. Identify activity risks
10. Lower-level planning
7. Allocate resources
9. Execute plan
8. Review / publicise plan

*Steps **1,2,3,4** and **8** are quality-related*

Quality definition and importance

Quality often vaguely defined

Stakeholders have **different views on quality**

Common fallacy: quality can be defined and improved later after all functionality is built

Software Quality refers to the degree to which a software product:

- meets specified requirements (functional/non-functional)
- satisfies user needs and expectations
- performs reliably under specified conditions

How to define quality in a project

- **Start with requirements** (F/NF) and specifications
- **Use measurable, objective metrics** to track quality
- **Define clear criteria** so you can check whether quality goals are met

Examples of metrics:

- Performance: system response time under 2 seconds, etc.
- Defect limits: fewer than 5 critical defects per release; less than 10 defects per 1,000 lines of code, etc.
- User satisfaction: average score of 4.5/5 in post-task surveys, etc.

PRODUCT *vs.* PROCESS Quality Management

PRODUCT or deliverable

Is the product good?

Termed **quality control (QC)** – evaluation of the final product against requirements

Ideal but difficult – complex and costly

PROCESSES used to build the product

Did we build it in a good way?

Termed **quality assurance (QA)** – evaluation of the processes behind product development

Easier to manage

Can also be applied to third-party software

Standards and Models in Quality Management

Process Quality Management

The purpose is to ensure that the project conforms to the mutually agreed requirements, specifications, and expectations.

Maintain consistency and **reliability** throughout the **project lifecycle**

- **CMMI** (Capability Maturity Model Integration)
- **Six Sigma**
- **Agile and DevOps Perspectives:**
  - **Agile Quality Management Practices**
    - Agile Testing Quadrants
  - **DevOps Quality Management Practices**
    - Continuous Integration (CI)/ Continuous Delivery (CD)

Software Quality and the ISO 9126 Standard

Product Quality Management

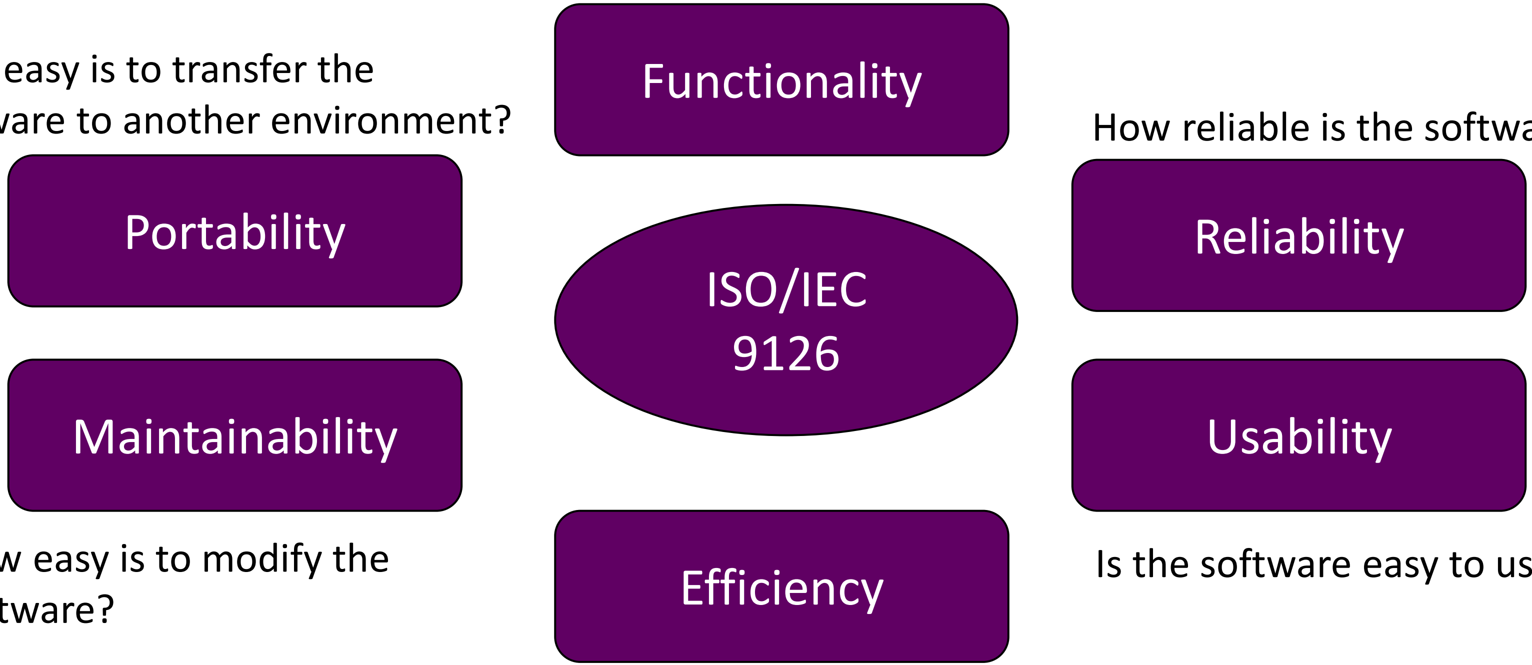
ISO 9126: Evolution of Software Quality Standards

- **Development of checklists**: Increasingly comprehensive software quality checklists led to the ISO 9126 standard (1991)
- **ISO 9126 updates**: Expanded versions released periodically

- **Separate documents for stakeholders**:
  - **Acquirers**: obtaining software from suppliers
  - **Developers**: building the software
  - **Independent evaluators**: assessing software quality for users

Software Quality and the ISO 9126 Standard

Six primary quality characteristics



Are the required functions available in the software?

Functionality

How easy is to transfer the software to another environment?

Portability

How reliable is the software?

Reliability

ISO/IEC 9126

Maintainability

Usability

How easy is to modify the software?

Efficiency

Is the software easy to use?

How efficient is the software? (performance)

Evolution into newer standards like ISO/IEC 25010

- It is part of the **ISO/IEC 25000 series**
- Also known as **SQuaRE** - Software Quality Requirements and Evaluation
    - This standard replaces and expands upon the earlier **ISO 9126** model.

- Expanded Quality Model is divided in two key models:
    - **Product Quality Model** (**8 key characteristics** of <u>software product </u>itself)
        - Added security and compatibility, refined functionality and efficiency
    - **Quality in Use Model** (**5 characteristics**, focuses on <u>user experience</u>)
        - Focuses on the impact of the software in real-world use

Summary of key points in Quality Management

- **Quality is essential in software**
- **Quality must be built-in from the start**, not added as an afterthought
- **Quality-related requirements** need clear, measurable definitions so we can verify them objectively
- **Quality Control (QC)** checks the **product**, and **Quality Assurance (QA)** improves the **process** — both are needed for success
- **Modern quality management combines frameworks** (CMMI, Six Sigma, Agile, and DevOps) focusing on processes: continuous improvement, testing, and automation
- **Best practices and standards** improve capability but do not guarantee quality

Software Project Management

Unit 6: Project Monitoring & Control (plus) Quality Management

Thais Webber
Richard Lee