# CE29x Team-Project Challenge

## Estimating task durations

Professor Anthony Vickers

Room: 1NW.3.17 e-mail: vicka

with acknowledgements to Keith Primrose, Iain Langdon, Mike Fairbank and adaptavist

# Recap of Previous Lecture

Previous Lecture

- Agile vs Waterfall
- Gathering Requirements
- Modelling Requirements
  - SRS
  - SMART Criteria
  - User Stories

# Recap of Previous Lecture

Benefits of Agile:

Agile methods Embrace Change:

* Requirements refine
* Information becomes available
* Technology evolves
* Legislation changes
* Business needs evolve

# This Lecture

- Agile: Scrum Versus Kanban
- Techniques for Estimation
  - Estimating whole Projects
    - Pricing Projects
    - The Work Breakdown Structure
  - Estimating Tasks
    - Story Points
- Duration, Effort + Resources

# Scrum vs Kanban

| Scrum | Kanban |
| --- | --- |
| Iterations (Sprints) | Continuous Flow |
| Commit to delivery at start of each Sprint | Always deliver next priority item |
| Regular frequent delivery | Continuous Delivery expectation |
| Sprint Ceremonies and Roles | |
| Embrace change | |
| Encourage early feedback | |
| Continuous Integration and testing | |
| Visual boards for progress | |

# Techniques for Effort Estimation

# Effort Estimation

- Managers must plan projects by predicting necessary effort (and cost) and assigning resources appropriately.
- Effort is measured in person-days, person-week, person-month or number of hours spent.
- It is tempting to pluck figures out of thin air, but is there any more rational approach?
- The desire by managers for improved methods of resource estimation provided one of the original motivations for deriving and using metrics and measures.

# Warm-up Estimation Exercise

- How long do you think it would take you to program "Asteroids" from scratch?
- Take the survey on Moodle in Unit 2


30-second demo of Asteroids
https://www.youtube.com/watch?v=psaM7kK5Toc

# Approaches to Effort Estimation

- Estimating whole projects:
  - Estimating by Analogy
  - Algorithmic model for estimates
  - Using a Work Breakdown Structure
  - Pricing projects
- Estimating individual tasks:
  - Expert Judgement or Guesses : based on non-documented experience
  - Delphi method
  - Using increasingly fine sub-tasks
  - Using Story Points

These are described further in the following slides:

# Estimating whole projects

1. Estimating by Analogy
2. Algorithmic method
3. Work Breakdown Structure
4. Pricing Projects

(Useful when you are bidding for a piece of work)

# Estimating by Analogy

- The results of similar projects from the past can be used as a basis for the estimate.
- Such project data can be found in local in-house project databases compiled in the past, or from a database obtained from a user group
- Danger of false analogy by lack of rigour in the choice of data and lack of formal method

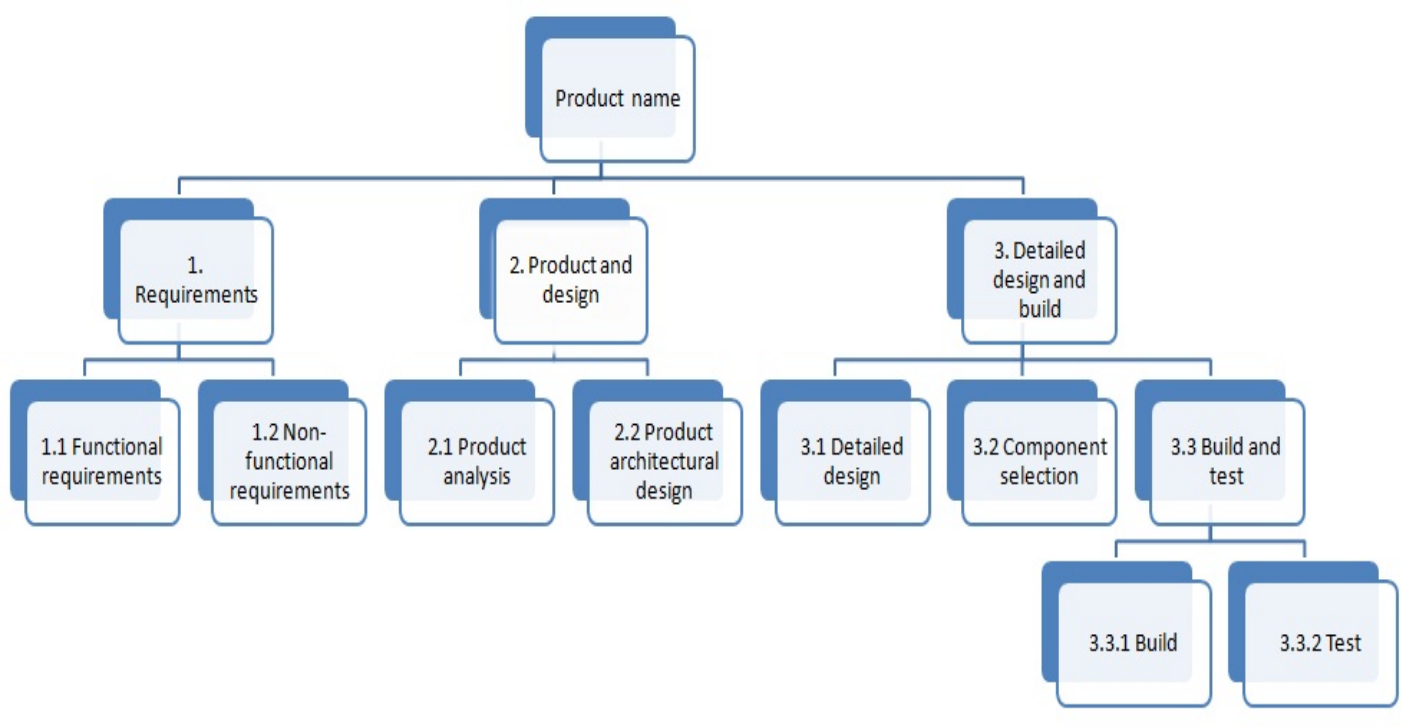
# Algorithmic model for Estimates

- Effort is expressed as a function or algorithm of one or more variables
  - Such as size of the product, complexity,
    - Number of database tables in the design
    - Number of windows in the design
  - capability of the developers
  - level of reuse
    - From previous similar project
    - Off-the-shelf open-source libraries

# Algorithmic model for Estimates

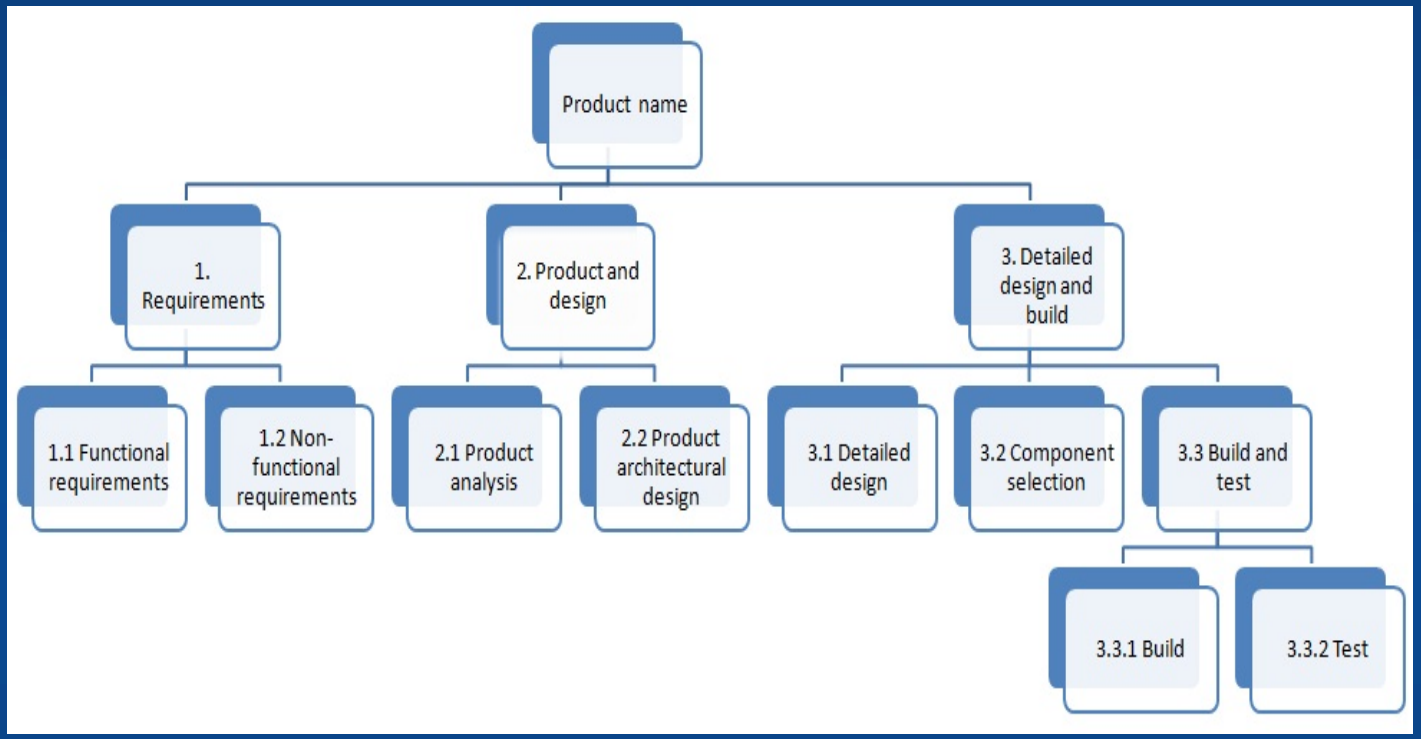
- A Work Breakdown Structure can be used to get an overview of the whole project (Waterfall)

# Work Breakdown Structure (WBS)



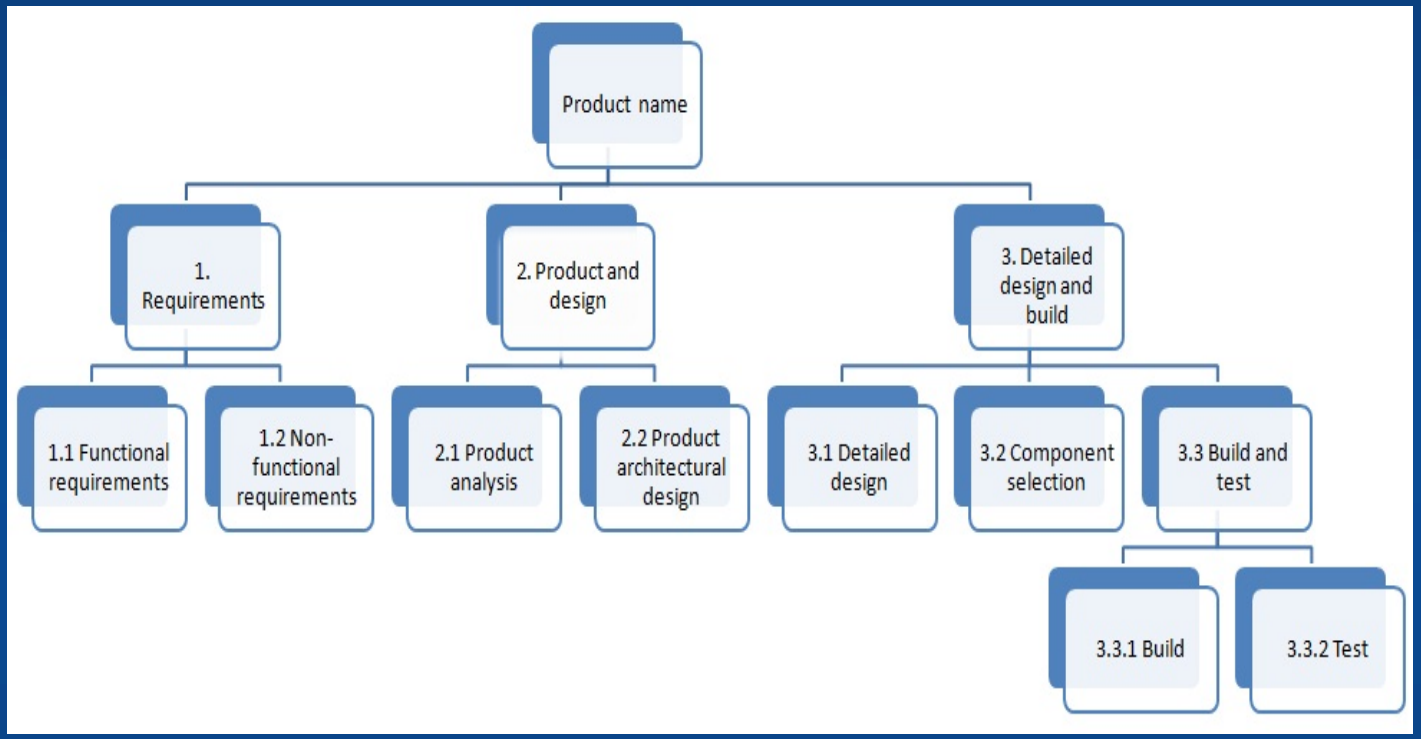

* Watch the video tutorial on WBS

# Work Breakdown Structure (WBS)



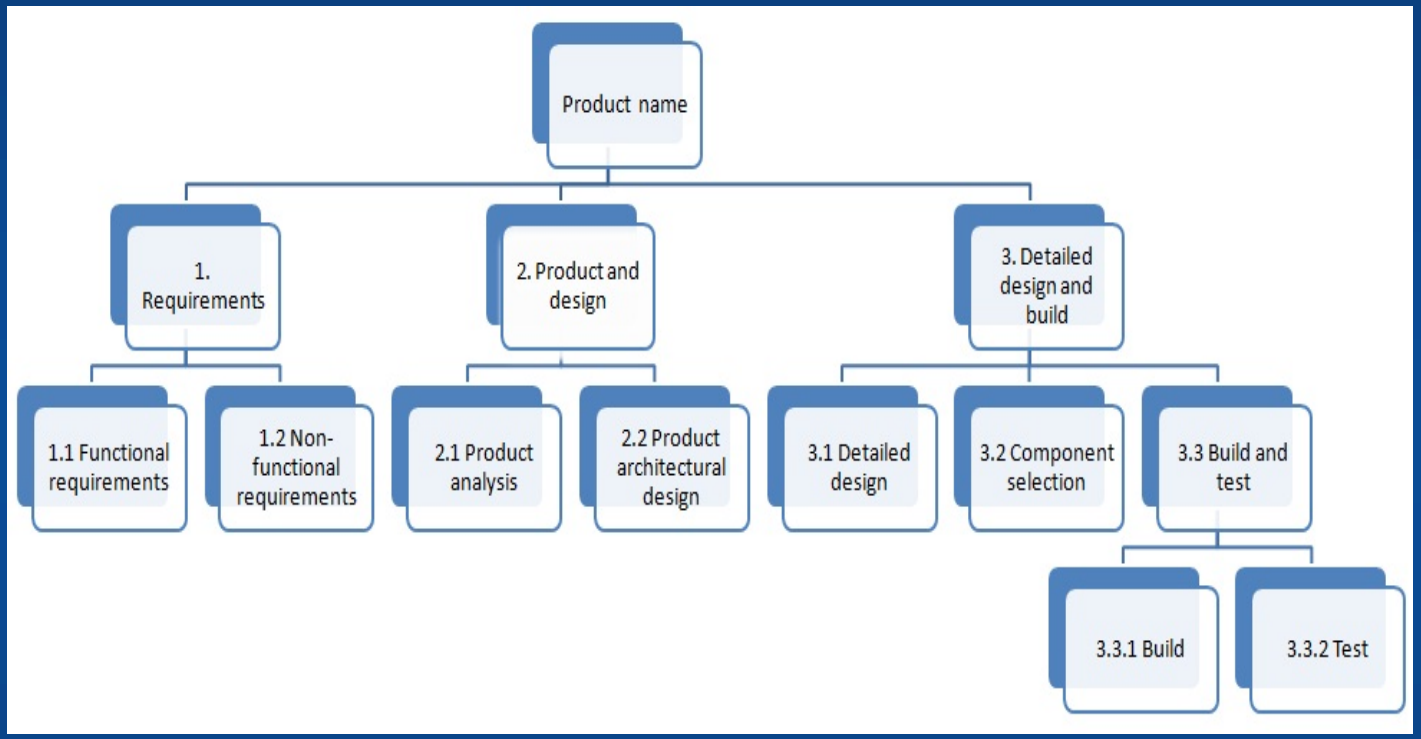

- Each descending level of the WBS consists of ordered sets of activities and tasks and represents an increasingly detailed description of the project elements

# Work Breakdown Structure (WBS)



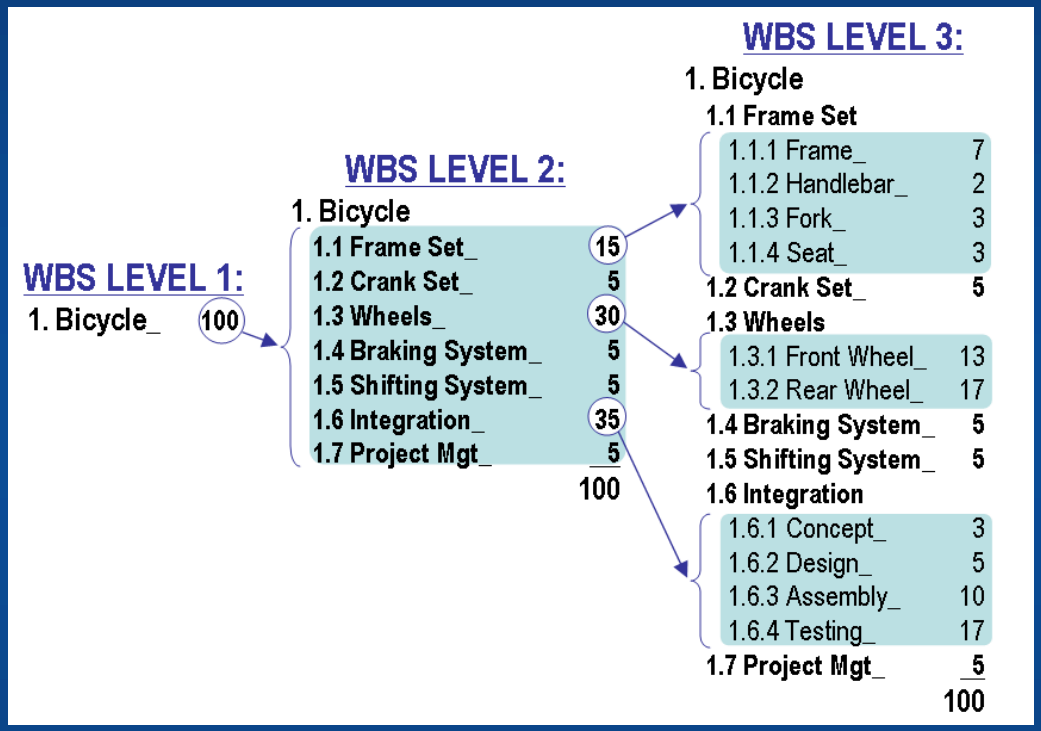

- For most engineering projects, the WBS usually consists of "deliverables"
- The WBS defines the whole scope of a project ("100% rule")

# Work Breakdown Structure (WBS)




- Work not in the WBS is outside the scope of the project
- When the scope of the project changes, the WBS usually changes.

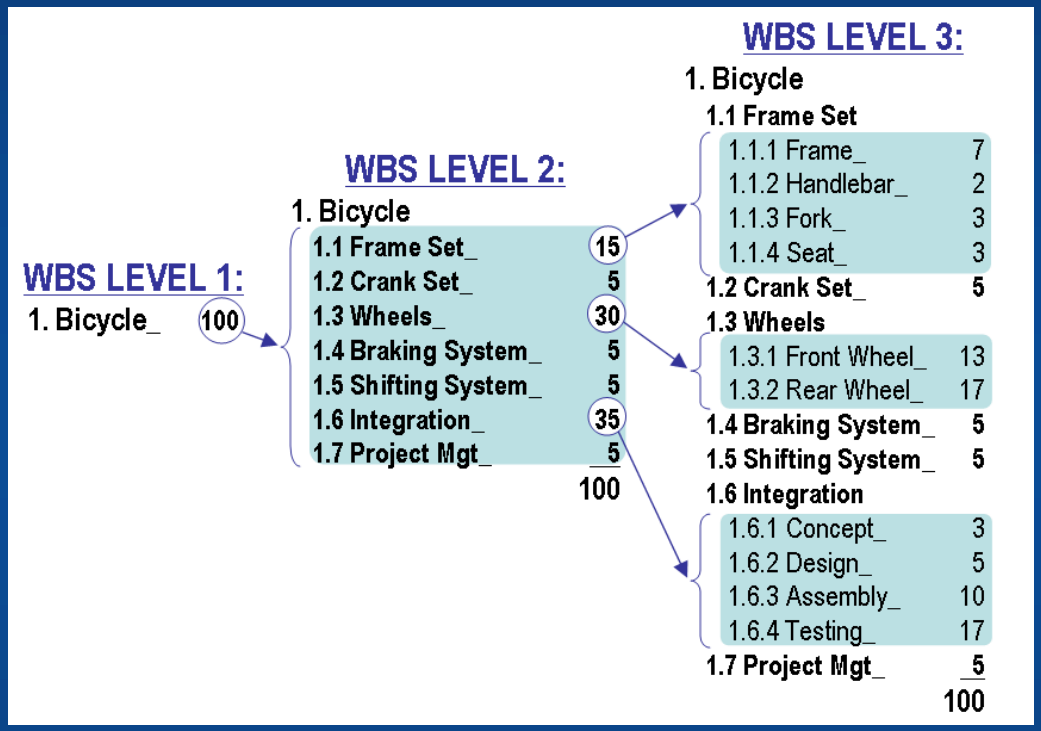
# Work Breakdown Structure (WBS)




* The lower-level components should be both necessary and sufficient for completion of the decomposed upper level item

What is the implication of this last comment and, what could cause it not to be true?

# Work Breakdown Structure (WBS)




- Constituent elements of WBS should be described in terms of tangible, verifiable results (services or products, components… ).

# Calculating Cost Estimates

Once the whole project scope and tasks are understood, we can price the project:

- Costs are based on effort and the cost of tasks are well known on the IT market:
  - Per/hour figure for analyst/designer etc.
    - + overhead figure
    - x utilisation factor (allowing for holidays, illness, training etc.)

Utilisation figure of > 80% is suspect – why?

# “Price to Win” estimates

- Unfortunately estimates in practice are often done on a price-to-win basis,
  - meaning that the cost and time expectation of the sponsor are accepted without validation as the actual estimate.
- The price the customer is ready to pay, the price to win the contract…
- Price-to-win means we need to calculate the number of hours we can afford to develop for, *after* we’ve chosen the price
  - – depends on the development team’s hourly rate of pay.

# “Price to Win” estimates

- How much would you bid to develop Asteroids from scratch?
  - Must be delivered exactly 2 weeks from now.
  - Financial penalties for late/substandard delivery
- A combination of lowest bidder and best established reputation wins the contract!

# Estimation is a continuous process

Even in waterfall projects:

- Costs and durations will be re-estimated at key dates: end of phases and milestones; whenever a major change is introduced; when unexpected technical problems occur…
- Lecture on "project control" will discuss this further.
- Estimates are more and more accurate when advancing in the development process

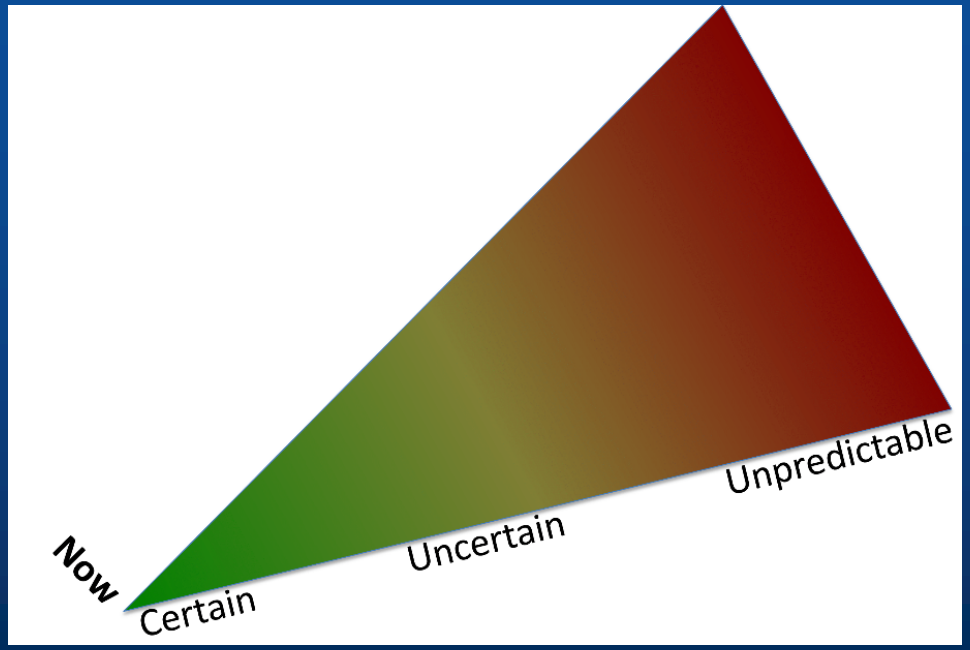
# This Lecture

- Agile: Scrum Versus Kanban
- Techniques for Estimation
  - Estimating whole Projects
    - Pricing Projects
    - The Work Breakdown Structure
  - Estimating Tasks
    - Story Points
- Duration, Effort + Resources

# Estimating Individual Tasks

# Estimating Individual Tasks

1. Delphi Method
2. Splitting into increasingly finer subtasks
3. Story Points and Velocity
   * Planning Poker



# 'Delphi' Method of Estimating

Delphi method of estimation is as follows:

- Get a wide number of experts (or non-experts) with relevant experience (database, application, communications, operations, package, development environment, industry…) to make an initial estimate.
- All have the same information concerning the project
- Individual estimates are discussed in turn with all experts including the reasons and assumptions made by each individual
- Second round where each expert makes a revised estimate to incorporate others findings and discussion
- Take the median, extremes excluded

# 'Delphi' Method of Estimating

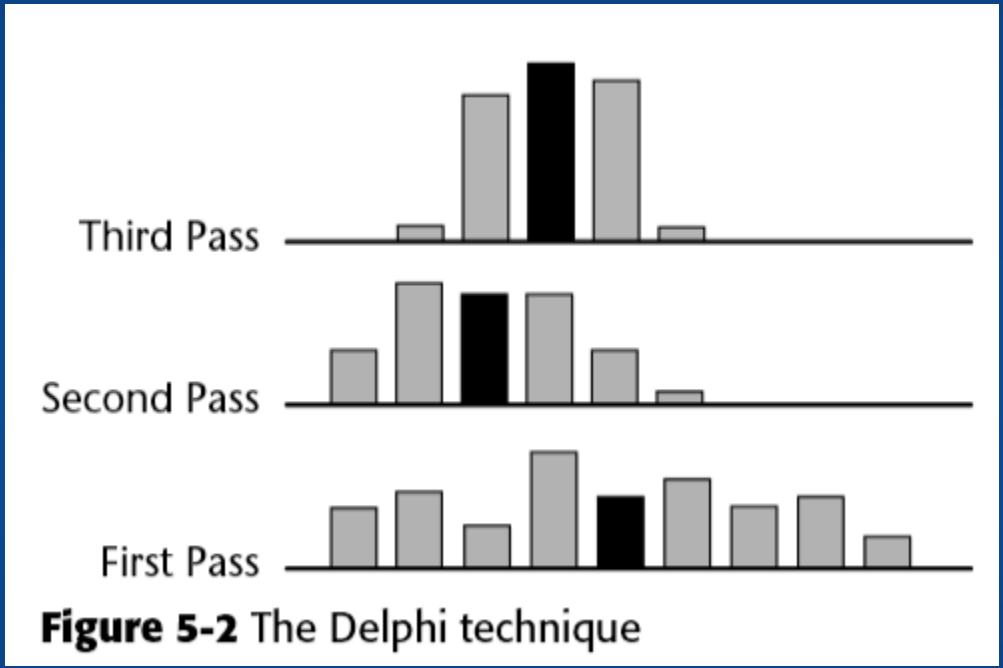



**Figure 5-2** The Delphi technique

Fig 5.2 from Wysocki "Effective Project Management : Traditional, Adaptive, Extreme"

# 'Delphi' Method of Estimating


- You were asked: "*How long do you think it would take for yourself to program the classic arcade-game Asteroids from scratch?*"
- I will review the Asteroid Estimation Survey in a live presentation.
- For the Delphi method to be applied, after the survey, we should all discuss our answers and argue who is right and why, and then survey again

# 'Delphi' Method of Estimating

Related to using a panel of experts to make estimates*:

* In machine learning, we can *improve the accuracy* of machine predictions by using several different predictors and average, rather than relying on one.
  * Ensemble methods
* Likewise, the Bank of England uses a panel of experts who all have to agree when to set/change interest rates
* Bankers collate to rig/set interest rates (Libor scandal)

# Splitting finer subtasks can raise estimation precision

- What is our estimate for programming Asteroids, if we break the task down into finer sub-tasks:
  1. Get spaceship moving around a blank screen (with keyboard control)
  2. Create some drifting asteroids (no collision detection)
  3. Collision detection between spaceship and asteroids
  4. Collision detection between bullets and asteroids, and implement asteroid-splitting
  5. Finish game off (Game-over after 3 lives lost; score display, level restart after clearing level; UFOs)

- Take the refined asteroid survey on Moodle when released in Unit 2.
- Is using this method better than just plucking a figure out of thin air for estimating Asteroids?

# Story points: Measure complexity

- Instead of estimating in hours/days, we can estimate in relative task complexity: story points
- Rate how complex a task is *relative to* the tasks rated so far

# “Planning poker” with story points

- We are going to choose a complexity from the following options: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?, ∞
- Can buy/make cards for your team to do this (need one pack each)

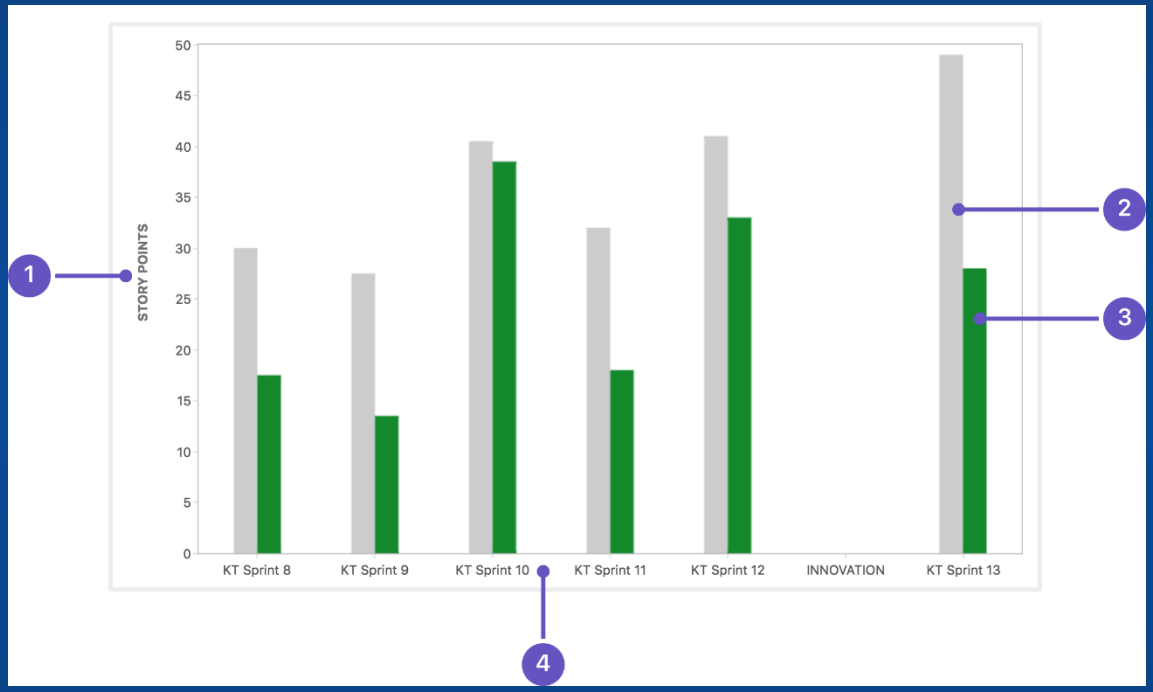

- Backlog of tasks to estimate:
  - Choose recipe
  - Find ingredients
  - Cook meal
  - Serve meal
  - Eat meal

# “Planning poker” with story points


- All team sits round and votes on tasks one-by-one using cards
  - After you’ve seen each other’s cards, discuss the highest and lowest cards
  - “Why did you think *Find ingredients* took so long?”
  - Then everyone votes on the same task again
    - (Delphi method again)
- Very popular in Agile
- You should do this in your team meetings
- See https://www.mountaingoatsoftware.com/agile/planning-poker for more details

# Story points: Average Velocity

The number of average story points a team completes per sprint is called “velocity”.

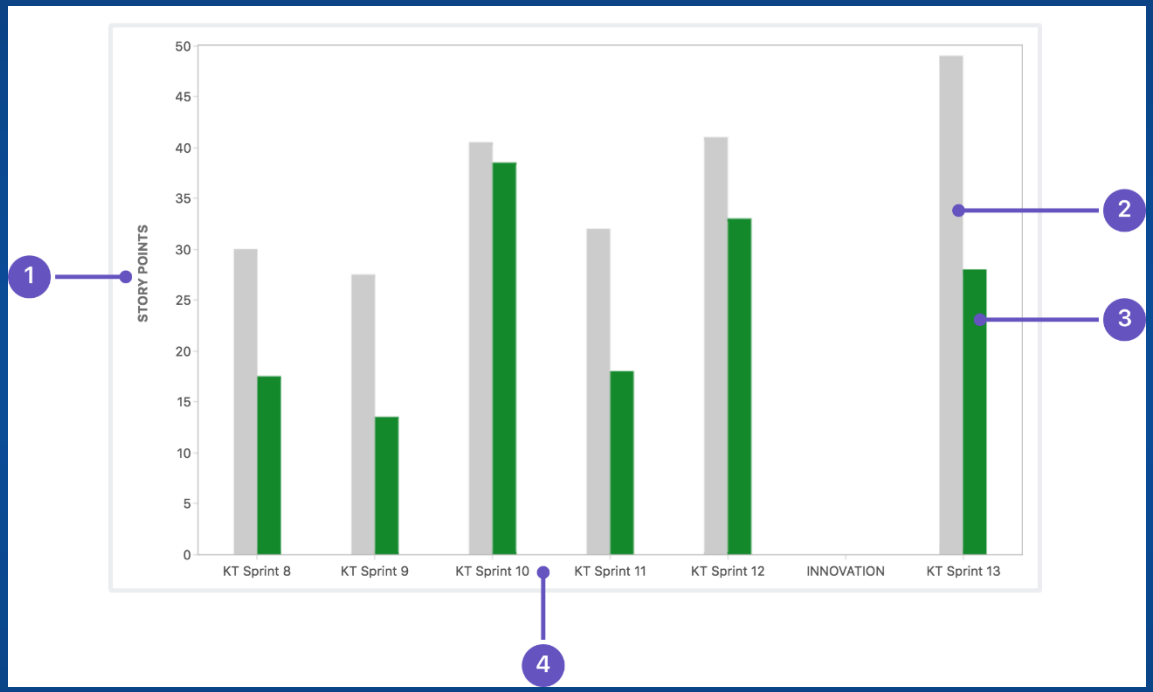



2 (Grey) = what the team promised each sprint (total story points)

3 (Green) = what the team delivered each sprint (total story points)

# Story points: Average Velocity

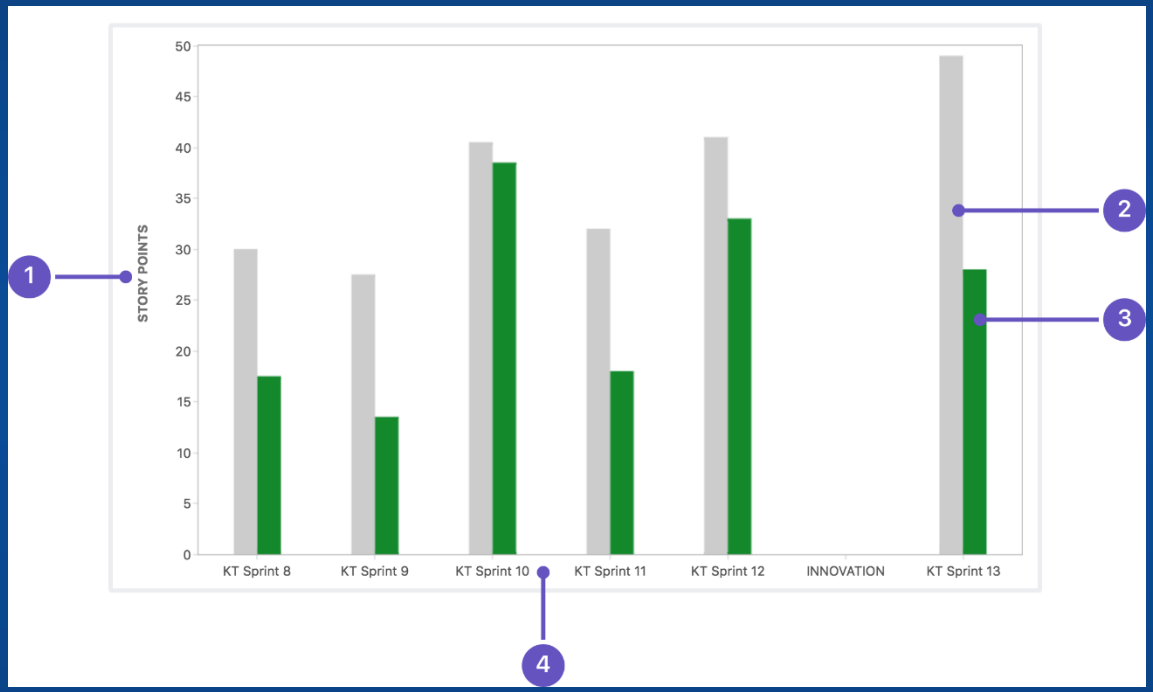
The number of average story points a team completes per sprint is called “velocity”.




- This team is averaging 24.75 story points per sprint (average of green bars)
- Can use this to plan what they can deliver next sprint
- Need a consistent way of assigning story points for velocity to make sense

# Story points: Average Velocity

The number of average story points a team completes per sprint is called “velocity”.




* Using velocity allows us to automatically rescale to a team’s rate of work and scale of story points

See https://confluence.atlassian.com/jirasoftwarecloud/velocity-chart-777002731.html for further details

# Duration, Effort + Resources

# Duration and Effort

- Duration is : How long it will take to complete the activity:
  - it is an elapsed time, not an effort estimation.
  - E.g. "this task will take 5 programmers 2 days to implement."

- Effort is the total time spent by all resources on a specific task

  E.g. the above task takes how many "programmer days" to complete?

# Duration, Effort + Resources

- "Resource" is the number of people (or machines) working on a task. E.g. Resource = 5 builders + 1 JCB digger
- Therefore:
  - Duration = Total Effort / Resource

    e.g. duration = total "programmer days" / number of programmers available per day.
  - NB – the same units of measurement (e.g. days) must be used for each component

Whole days is generally a sensible minimum unit of duration.
What is the risk of using hours?

# Resource Example

- Resource dictates the amount of useful work that can be done per day.
  - 6 builders get 6 times the amount of work done compared to 1 builder

E.g. Three painters work two days on a task, they work 8 hours per day.

- The total resources available per day is
  - 3 people
- The total effort for the task is:
  - 2 (days) * 3 (people) = 6 person days
- The duration is
  - effort / resource = 6 / 3 = 2 days.

# Duration, Effort + Resources

- The relationship between effort estimates and elapsed time is often not quite that simple
- A task with duration=2 days could take 4 calendar days if the task started on a Friday
  - We can't talk about calendar days until the final schedule is published (because start dates float around until then)
- Here are a just few things that might influence the elapsed time:
  - Calendars (staff has booked holiday)
  - Number of personnel
  - Skill sets
  - Availability of other resources (e.g. tools)

# Resource, Effort and Duration

- E.g. Adding a second programmer might not halve the duration of a task:
  - It might be that the second programmer can't start developing until the other has finished their bit
  - How should the Project Manager account for that?

# Impact of Dependencies on Duration

- Task dependencies have an impact on the duration of tasks
- E.g. Effort = 6 days.  Resources = 3 painters.
- Dependencies could mean the 3 painters can't work on the same days
- For instance, 2 of them will do the walls and ceiling, the 3rd will do the stencils.  Stencilling cannot start until walls and ceiling completed.
- How should we account for this in the planning?
  - We need to introduce three separate tasks
  - The wall and ceiling tasks can be run concurrently
  - But the stencilling task can only begin after the tasks made by the two others is completed
  - Now the 3 painters need longer than 2 days to complete the job.

# Summary of Lecture

- Looked at Estimating Projects and Tasks
  - Whole project estimation
    - Work Breakdown Structure
    - Algorithmic / Analogy methods
  - Delphi Method
  - Using finer subtasks
  - Planning Poker (Story Points)
- Pricing Projects
- Defined Effort, Duration and Resources

  + their estimation techniques

Further reading: Wysocki "Effective Project Management : Traditional, Adaptive, Extreme", chapters 4+5.

Practice quizzes for Unit 2 will be available on Moodle. Look out for them in Unit 2.