Lecture 9: Estimation and Prioritization

- → Project planning
- → Estimating Effort
- → Prioritizing Stakeholder's needs
- → Trade-offs between stakeholder goals

Project Planning

Given:

A list of customer requirements

E.g. a set of use cases, a set of change requests, etc.

Estimate:

How long each one will take to implement (cost)

How important each one is (value)

Plan:

Which requests should be included in the next release

Complication:

Customers care about other stuff too:

E.g. quality, performance, security, usability,…

Key Principles of Management

A manager can control 4 things:

- **Resources** (can get more dollars, facilities, personnel)
- **Time** (can vary the schedule, delay milestones, etc.)
- **Product** (can vary the amount of functionality - e.g. scrub requirements)
- **Risk** (can decide which risks are acceptable)

Approach (applies to any management)

- Understand the goals and objectives
  - quantify them where possible
- Understand the constraints
  - if there is uncertainty, use probability estimates
- Plan to meet the objectives within the constraints
- Monitor and adjust the plan
- Preserve a calm, productive, positive work environment

Note:

- You cannot control what you cannot measure!

Strategies

Fixed Product

1. Identify customer requirements
2. Estimate size of software needed to meet them
3. Calculate time required to build this much software
4. Get customer to agree to the cost & schedule

Fixed schedule (a.k.a. Timeboxing)

1. Fix a date for next release
2. Obtain prioritized list of requirements
3. Estimate effort for each requirement
4. Select requirements from the list until the "box" is full

Fixed Cost

1. Agree with customer how much they wish to spend
2. Obtain prioritized list of requirements
3. Estimate cost of each requirement
4. Select requirements off the list until the "cost" is used up

Estimating Effort: COCOMO

Source: Adapted from van Vliet, 1999, section 7.3.2

COnstructive COst Model (COCOMO)

- Predicts cost of a project from a measure of size (lines of code)
- Basic model is: $E = aL^b$ (effort; project specific factors; lines of code)

Modeling process

- Establish type of project (organic, semidetached, embedded)
  - this gives sets of values for a and b
- Identify the component modules, and estimate L for each module
- Adjust L according to how much is reused
  - COCOMO has a model for adjusting according to how much design, code and integration data is reused
- Compute effort for each module using $E = aL^b$
- Adjust E according to difficulty of the project
  - COCOMO identifies 15 effort multipliers to take into account
  - **Product attributes:** eg required reliability, complexity, database size
  - **Computer attributes:** eg execution time constraints, storage constraints, etc.
  - **Personnel attributes:** eg capability & experience of analysts and programmers,
  - **Project attributes:** eg use of CASE tools, programming language, schedule
- Compute time using $T = cE^d$
  - c and d provided for different project types like a and b were

Estimating Size: Function Points

Source: Adapted from van Vliet, 1999, section 7.3.5

Function Points

- used to calculate size of software from a statement of the problem
- tries to address variability in lines of code estimates used in models such as COCOMO
  - e.g. because SLOC varies with different languages
- Originally for information systems, although other variants exist
- Basic model is: $FP = a_1 I + a_2 O + a_3 E + a_4 L + a_5 F$ (metric from problem statement; weighting factor for this metric)

Example

- Sets of weightings (a_i) provided for different types of project
- Measure properties of the problem statement:
  - I = number of user inputs (data entry)
  - O = number of user outputs (reports, screens, error messages)
  - E = number of user queries
  - L = number of files
  - F = number of external interfaces (to other devices, systems)
- Example calculation:
  - $FP = 4I + 5O + 4E + 10L + 7F$

Simpler approaches to estimating

Estimation in Practice:

- People tend to underestimate effort needed
- Most estimates are made to please the {boss, customer, …}
- Easier to estimate small chunks of work than large ones

Three-point estimating

- Provides better estimates than just asking for a range
- w = worst possible case
- m = most likely case
- b = best possible case

$$E = \sum_i \frac{w_i + 4m_i + b_i}{6}$$

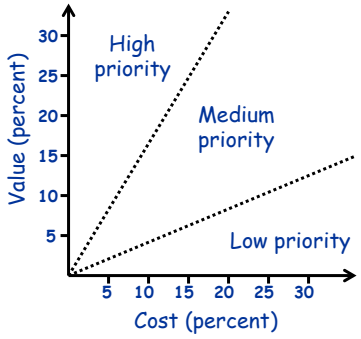
…and don't forget: effort < duration !!

Not All Requirements Are Equal!

Source: *Adapted from Karlsson & Ryan 1997*

Perform Triage:

- Some requirements *must* be included
- Some requirements should definitely be excluded
- That leaves a pool of "nice-to-haves", which we must select from.



Some complications

Hard to *quantify* differences

- easier to say "x is more important than y"…
- …than to estimate by how much.

Not all requirements comparable

- E.g. different level of abstraction
- E.g. core functionality vs. customer enhancements

Requirements may not be independent

- No point selecting between X and Y if they are mutually dependent

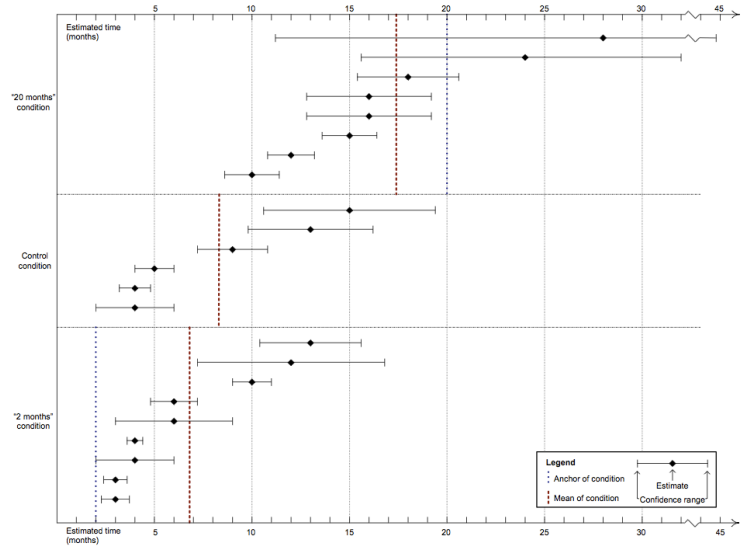
Stakeholders may not be consistent

- E.g. If X > Y, and Y > Z, then presumably X > Z?

Stakeholders might not agree

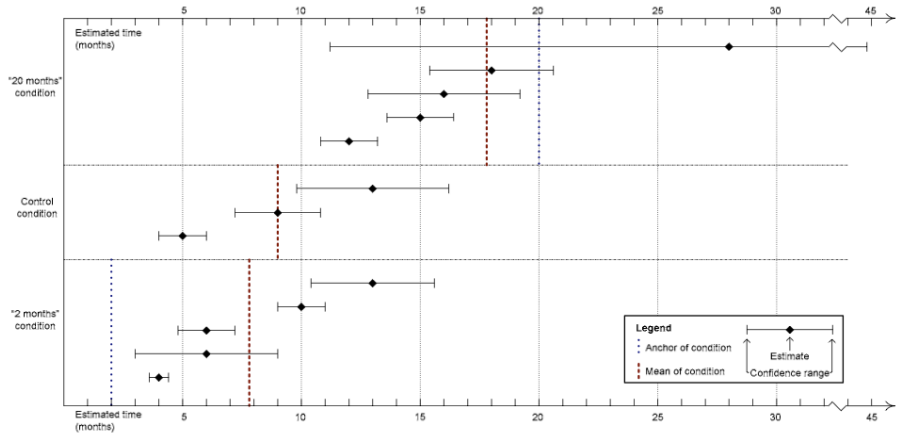
- Different cost/value assessments for different types of stakeholder

Biased Estimates…



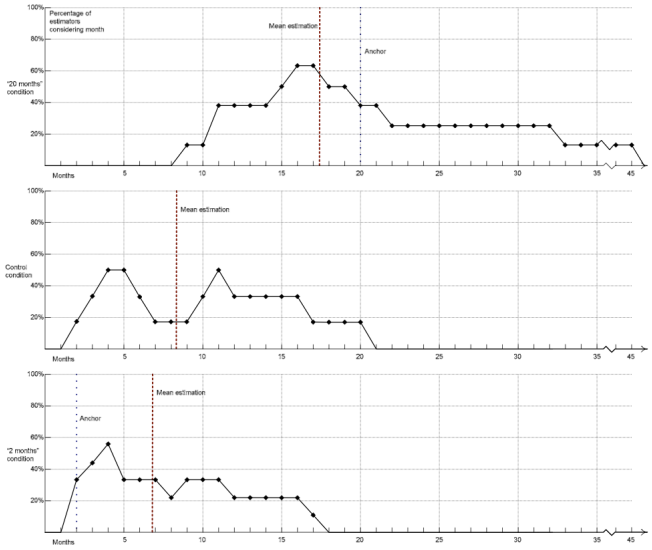
Source: Aranda & Easterbrook 2005 "Anchoring and Adjustment in Software Estimation"

Even with experienced estimators…



Source: Aranda & Easterbrook 2005 "Anchoring and Adjustment in Software Estimation"

By confidence interval



Source: Aranda & Easterbrook 2005 "Anchoring and Adjustment in Software Estimation"

Agile Approach

Measure progress of small tasks

- Reduces regions of no visibility
- Use the issue tracking system!

Estimate effort for each task

- adjust estimate of remaining effort as iteration progresses

Only working code counts

- Task complete only when it delivers tested, working code
- *(okay, training material, manuals, etc also count - but nothing else!)*

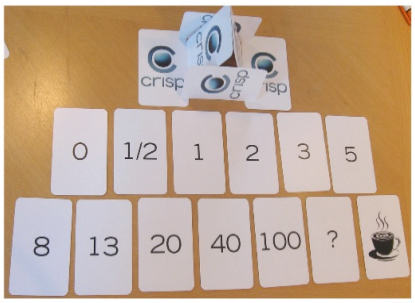
Visualize Progress

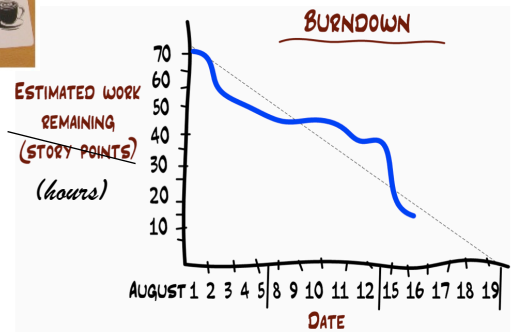
- E.g. use a burndown chart

Use your experience!

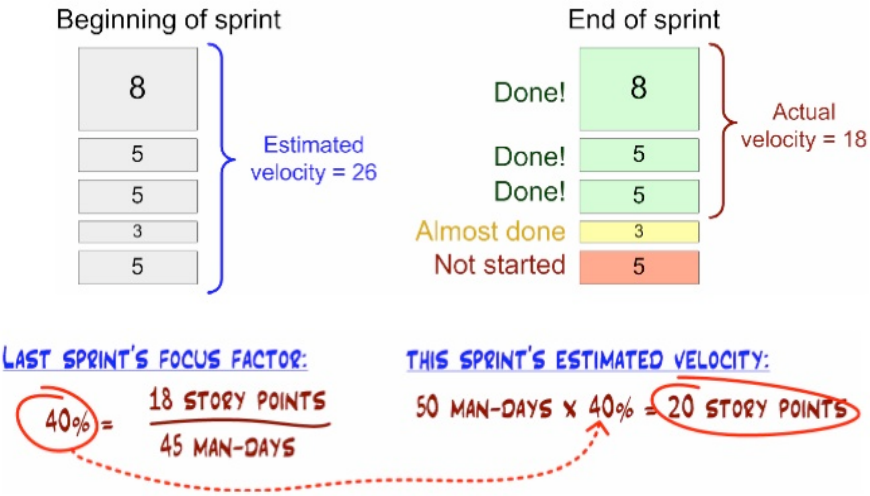
- E.g. if estimates are out on this iteration, adjust estimates for the next

Agile Estimation

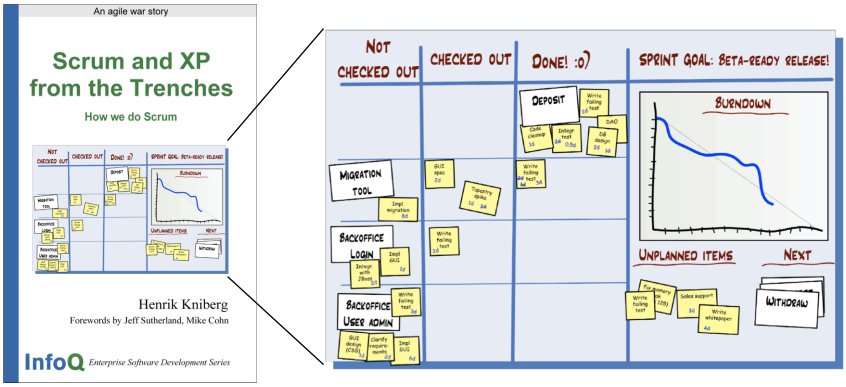




Continuously re-estimate velocity



Good read for intro to Agile…



Henrik Kniberg
http://infoq.com/minibooks/scrum-xp-from-the-trenches

Advice from ICONIX

Plan at appropriate detail

Negotiate the scope (when faced with fixed deadline)

Customer dictates priority

Adjust the plan to fit reality (small release cycles help)

Get feedback on progress and risks

Try to get it right first time (rather than fix it later)

Use 3 types of release: internal, investigative, production

Plan to re-factor when necessary (avoid rot)

Consider high impact decisions during early iterations