Session 7a: Managing Change

- The critical point revisited
- Freezing the specifications
- Reasons for change
- Decision process

Overview of phases (review)

- 1. Project definition
- 2. Business requirements specification
- 3. External design
- 4. System architecture
- 5. Construction
- 6. Installation
- 7. Review

What’s the significance of the dividing line between phases 3 and 4?

The critical point (review)

- The end of phase 3 (ESD, "Functional Spec.", "Detailed User Requirements", or . . ) is the most important point in a project's life cycle.

Why is that?

- It is the last point:
  - at which we can make changes without huge cost.
  - where the sponsoring end users can be expected to understand the deliverables in complete detail.
  - that is (or at least should be) independent of:
    - operating platform(s)
    - make or buy choice
    - development tools and methodologies
    - etc.

Development Contract

- Since the ESD defines exactly what the proposed new application system will do:
  - We can consider the ESD to be a contract between the developing organization and the sponsoring users.
    - If the developers deliver a system that satisfies the ESD, they will have met their obligation.

What’s wrong with that?
**Contract Terms**

- For the first time in the project, we (an in-house I.T. organization or an outside contractor) now know enough to be able to bid a **fixed price** and **fixed date** for the remaining project phases.
- Provided that the developing organization is
  - highly skilled and experienced
  - willing to take a little risk

- If not, it's common to agree on a **time and materials** contract with an estimate.

**So, do we freeze the specifications?**

- Only on a very short-duration project!
- Reasons for amending the specification
  - further insights
    - "We thought it over, and . . . ",  "By the way . . .")
  - correcting mistakes
  - changes to
    - laws & regulations
    - competitive business environment
    - technology
- What do we do in those cases?

**Giving up**

- Some recent fad methodologies advise that since change is inevitable, there's no point in preparing rigorous specifications (ESD) at all!

- The most extreme abandon the whole notion of a phased life cycle

  *What are the likely consequences of that?*

**Requesting a change**

- The project must establish a systematic way of receiving, evaluating, and resolving requests (from whom?)
  - Many organizations use a standard form (paper or online)
  - Don't make it so easy that you encourage a continuous flood of change requests.

- Resolution can be:
  - Accept change with no impact on cost or schedule
  - Accept change with budget increment and later target date
  - Reject change
  - Defer change to a release 2.
A rational approach

Q: Who should make the final decision whether to accept or reject a requested change?

A: The *sponsoring end-user* who's paying for the project.

The project must provide to the customers sufficient information to allow them to make that decision rationally.

- Benefits of the change / penalty for not making it
- Impact on the schedule
- Impact on the cost

Change management overhead

- Whether or not a requested change is accepted, the act of just responding to it consumes time of the project manager and appropriate technical specialists.

- A reasonable level of ongoing overhead should be planned and budgeted.
  - You must never have to say: "We would have met the target date if we hadn't had to respond to so many change requests."

Avoid bureaucracy

- Some experts advise setting up a change control board
  - Staffed by both user and I.T. people.
  - Discusses change requests and issues resolution decisions.

- Advantage:
  - Team members doing productive project work aren't distracted.

- Disadvantage:
  - Slow to respond.
  - Added cost.
  - Still need team members to estimate impacts.

Zero-impact changes

- Some requests can be handled without additional cost or schedule delay
  - Trivial changes
    - parameter values
  - Very early changes
    - We haven't built (or even documented) anything yet

- Requesting users will be pleased by the project team's response, but they mustn't be allowed to conclude that *all* changes are like that
Changes that are harder than they sound

- A user may assume incorrectly that a requested change is trivial, and may be unsympathetic to being told that it isn't:

  "Instead of yearToDateTotalSales we need the current rolling total of the last 12 month's sales."

A possible problem

- Sometimes a project team member will exaggerate the impact of a requested change, because we just don't feel like doing it:
  - The work would be tedious and boring
  - It would invalidate a really interesting technique that we've already committed to.

- Make sure senior people understand their responsibilities for honest evaluation

- Get a second opinion.

Adding functionality

- Requests will often call for something that wasn't part of the original specification:
  - Another report or inquiry transaction
  - Interface to some existing other application

- YAGNI principle tries to discourage that.

- Putting it on the queue for release 2 is tempting.

- But we still need to respond.

Relative cost of making a change during the project

What accounts for this?

What strategy does it suggest?
Does incremental development help to flatten the curve

- Sometimes, but not necessarily in the way you'd like!
  - As soon as the project starts building a base of program code, the potential cost of change is high.
  - And if we start coding before we have detailed requirements (ESD), the likelihood that something will have to be changed rises.

- That may be tolerable
  - on a very small project,
  - on a project that entails unpredictable research.

The change log

- The project workbook should contain a chronology of change requests with their resolutions.
- A change that alters something already in the workbook should be made to the affected section.
- Unless auditors or lawyers require it, in-line change history is just confusing clutter that make the documentation hard to read.