Session 2: A Phased Life Cycle for a modern system development project

Concepts and Terminology
- We already know what these are:
  - Project
  - Successful project
- This session will look at these:
  - Task / Task specification
  - Phase / End-of-phase decision
  - Project life cycle
  - Life-cycle methodology
  - Life-cycle system
  - Methodology

Task:
- A unit of project work that:
  - Can be assigned to a single individual
  - Is worth keeping track of
  - Produces a well-defined result ("task deliverable")

  The only official unit of project work.

Task specification:
- A plan for performing a task, including:
  - Specification of task deliverables or tangible results
  - List of prerequisite tasks.
  - Estimates of resources needed / cost
  - Estimated duration

Predecessor (or prerequisite) relationship
- What does it mean for task P to be a prerequisite to task Q?
  - Task Q cannot be started until task P is finished.
    - usually because task Q needs the deliverables produced by task P
  - A milestone event has prerequisites but 0 duration

  What's that for?
A simple task network

(assuming numbers are the number of days required to perform the task)

How long is the critical path?

The critical path

(11 days if nothing goes wrong)

Project plan:

- A network of task specifications.

What do we mean by a network?

- A single final (end) task or milestone (major intermediate result) has no successors
- No other task has no successors.
- The start task has no predecessors. Why?
- There may be more than one start task.
**Weisert's rule #1 on project failure**

- **Insufficient detail in the project plan** is one of the two most common reasons why projects fail. That is:
  - The tasks on the plan are too big (too long duration) for accurate estimating and tracking.
  - There aren't enough tasks to provide visibility over the work to be done.
  - Task deliverables are only vaguely specified

  *How can we avoid such failures?*

---

**How big should a task be?**

- A task is *too big* if:
  - It isn't clear exactly how it can be done.
  - Its estimated duration exceeds 2-4 weeks.
- A task is *too small* if:
  - The effort required to administer (assign and keep track of) it is comparable to the effort to do it.
  - It is the only successor to another small task that is always assigned to the same person.

  *Example?*

  - When in doubt, too small is better. *Why?*

---

**Aggregating the task estimates**

- **Cost**: sum for all tasks of resource cost \( \times \) resource quantity
- **Duration**: sum of duration along longest ("critical") path in the network.
  - This is the *minimum duration* even if infinite resources were available.

  *This will be covered in detail later*

---

**Task-level status reporting**

- Which of these status reports contains more information?

  - An eight-week task is 50% done
  - 4 (of 8) one-week tasks are complete
Non-binary task status

- What does it mean to say that a task is:
  - "50 percent done"?
  - "95 percent done"?
  - "complete (except for a couple of minor loose ends that we'll take care of this weekend)"?

Binary task-status reporting

- Once the project manager assigns a task to an individual project team member, its status must be tracked, i.e. reported periodically.
- Only 2 possibilities: a task is either:
  - complete (task deliverables on file and available)
  - incomplete, with definite estimates of target date and remaining cost. Those estimates may either:
    - confirm (or improve upon) previous estimates
    - or (slippage!) be supported by an explanation

Task-level status reporting frequency

- How often should team members report on the status of tasks assigned to them?
  - Often enough to assure early warning of slippage or overrun
  - but not so often as to constitute unreasonably burdensome overhead
- For typical information system development projects weekly status reporting is a good balance
  - Exception: At the end of a project (installation, conversion, and start-up) durations may have to be stated in hours!

Avoiding loss of control

- The combination of:
  - Short task durations and
  - Weekly binary task status reporting forces each team member to face reality now rather than hoping for a miracle later.
Corrective action

- Even with early warning, what can a project manager do after discovering that tasks are slipping?
- Can we correct the problem by
  - adding more people to the project team?
  - compressing the schedule for later tasks?
  Both are popular pseudo-remedies!

Brooks's Law

- "Adding manpower to a late project makes it later."
  - Frederick P. Brooks, jr.: *The Mythical Man-Month*, 1982, Addison-Wesley

Do we agree? Why?

Making up time in later tasks

- Consider this argument:
  1. Our project is now 3 weeks behind schedule.
  2. We originally allocated 5 weeks for final system acceptance testing.
  3. Therefore, if we do the acceptance testing in 2 weeks, we'll make up the lost time and deliver the system on schedule.

- What's the obvious question?
- What are the likely consequences?

Project Phase:

- A major sequence of project activity that:
  - Represents a manageable commitment and an acceptable risk.
  - Can be planned, scheduled, and controlled in full detail
  - Produces a cohesive set of results ("phase deliverables") upon which the project's sponsors can make a rational decision:
  a. To approve / fund the next phase and continue, or
  b. To re-do part of this or an earlier phase, or
  c. To change the project scope or objectives, or
  d. To abort the project.
End-of-phase decisions

- Examine phase deliverables
- Satisfactory?
  - Y: Review budget & schedule for next phase
  - N: Assess costs, benefits, & risks
- More funds available?
  - N: Abort project
  - Y: Justified?
  - N: Re-do part of phase, or...
  - Y: Next phase

More terminology . . .

Project life cycle:
- The sequence of phases in a project.

Life-cycle methodology (LCM): (or model)
- A standard life cycle for almost all projects of a particular kind, such as:
  - application system development projects (SDLC)
  - major software enhancement projects
  - conversion (platform, language, DBMS, . .) projects
  - course development projects
- Is any uniform sequence practical and realistic?

Estimating
- At the start of each phase, the project manager prepares:
  - **Firm** estimates for the next phase:
    - budget
    - schedule
  - **Rough** estimates for the rest of the whole project:
    - range of costs
    - range of target dates
- At the completion of each phase we know more about the scope of work and can therefore narrow the range.

Moving-Window estimates of total project cost

- Maximum estimate for rest of project
- Minimum estimate for rest of project
- Range of uncertainty
- Actual cumulative project cost

COMP 377 / 477 21-24 Spring, 2016
II. Characteristics of a modern results-oriented life-cycle model

- Packaged life-cycle systems
- A reasonable 7-phase model
- Activity-oriented versus results-oriented LCM's
- Multiple layer presentation
- Deliverables specifications
- Model task specifications
- Phase cost ratios

Life cycle system (LCS):

- A packaged (how?) LCM often sold as a product. Examples?
- Such products were popular in the 1970's
- Some remain today, often built into a C.A.S.E. tool.
- Some have as few as 3 phases:
  - Analysis
  - Design & construction
  - Installation
- One has 14!

Reasonable and typical LCM phases for system development

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

Phase 1. Project Initiation

Representatives of prospective users and developers agree that:
- some definite problem might be solved or
- some definite opportunity might be met using information technology.

They establish a project, which then determines the initial project scope.
Phase 2. Business System Requirements

Systems analysts work with the prospective users:
- to specify what capabilities a new system must support
  and
- to assess the feasibility of developing (or buying) such a system

This phase is poorly understood and is rarely performed well.

Phase 3. External System Design

Systems analysts specify exactly what the proposed system will do in terms that both users and developers will understand.

They may also recommend (or specify constraints on) equipment, software, development tools, or design approaches.

What are some other common names for this phase?

Phase 4. Computer System Architecture

Technical specialists establish a structure for a computer-based system to implement the ESD.

In parallel with the above, systems analysts and users assess organizational impacts, and begin preparing plans for testing, training, and installation

What are some other common names for this phase?

Phase 5. Construction

Technical specialists build the system according to the overall architecture.

- For in-house (or contract) development, they:
  - design, code, and test programs
  - establish data bases
- For purchased components, they:
  - order, test, and install software products
- For all systems they:
  - prepare operational documentation,
  - package the system for routine operation, and
  - conduct integrated system testing.

What are some other names for this phase?
Phase 6. Installation

**Systems analysts, programmers, and users** cooperate to bring the already tested and documented components into operational ("production") status. Their activities include:

- **Training**, user personnel
- **Acceptance testing**
- **Conversion** (or "start up") to initialize the live database and lead into routine operation.

*What are some other names for this phase?*

---

Phase 7. Project review

The **developer organization** assesses the project, both improve its own future effectiveness and to propose improvements to the recently-installed system. They evaluate:

- How well the new system is meeting the **user organization's needs**
- the operational cost, performance, and reliability of the system.
- the cost and duration of the project.

This phase is often omitted. *Why?*

---

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

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

  *More about this later.*

- 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.
  - that is (or at least should be) independent of:
    - operating platform(s)
    - make or buy choice
    - development tools and methodologies
    - etc.
**Weisert’s Rule # 2 on project failure:**

- Inadequate external design is one of the two most common reasons why projects fail.

  **What was the other one?**

---

**What do we mean by "inadequate External Design"?**

- The deliverables contents are one or more of:
  - incomplete,
  - erroneous, inconsistent,
  - incomprehensible to the intended audiences,
  - mixed up with internal design.

  Especially this Common with inexperienced analysts.

---

**Typical phase cost ratios**

---

**Enlightened project failure**

- We know the two most common cause of project failure.
- We know how to avoid them.
- So why do so many projects still fail?
Management by wishful thinking

I agree with you in principle, but this project is so urgent that we just can’t afford the luxury of doing it right.

Methodology choices during a project

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

The tools and techniques we use in these analysis phases are (can be / should be) largely independent of the tools and techniques we use in these implementation phases.

Packaging and documenting an LCM -- alternative philosophies

- **Activity-oriented LCM:** (“cookbook”)
  - "Just follow the procedures and everything will turn out all right."
  - Most (not all) commercial LCS’s were of this type.
  - Below average or inexperienced staffs produce better quality systems under this approach. (Why?)

- **Results-oriented LCM:**
  - "Just produce phase deliverables that satisfy rigorously stated criteria."
  - Above average, highly skilled professional staffs produce better-quality systems under this approach. (Why?)

A 4-layer results-oriented LCM

1. **Concept** layer
   - Procedures, guidelines, etc.
   - Overview of phases

2. **Phase description** layer presents:
   - Purpose
   - Authors and their responsibilities
   - Audiences and their responsibilities
   - Deliverables overview

   1-3 pages for each phase
4-layer results-oriented LCM
(continued)

1. **Concept** layer
2. **Phase descriptions** layer
3. **Phase deliverables** layer
   - Specification or description of each required (or common known optional) result from this phase.
     *(often all the experienced professional needs)*
4. **Model tasks** layer
   - Specifications of tasks that could produce the deliverables.
     *Optional guidance*

---

**Level 3:**

**Phase deliverable specifications**

*6-20 pages for each phase*

- Specifications of **results** expected at the end:
  - Items needed for the end-of-phase reviews
  - Items needed in a later phase
  - Items for the permanent system documentation
- **What** is to be produced, not how.
  - Criteria for the *content*, not the form.
  - Hard copy or computer based deliverables.
- Built-in flexibility at several levels.

---

**Level 4:**

**Model task specifications:**

*1-2 pages for each task, 10-25 tasks for each phase*

- An optional (more activity-oriented) layer for project teams who want more guidance.
- A suggested network of task specifications for producing the phase deliverables.
- Each model task specifies:
  - Responsible **role**
  - List of **prerequisite tasks**
  - List of **task deliverables**
  - Cost and time **estimating guidelines**
- **What, when, and by whom,** but still not how.

---

**Model tasks versus real tasks**

- Some activity-oriented LCS's confuse the two (especially where an LCS is tied to related project management software), leading to seriously flawed project plans.

- Note that one model task may give rise to multiple actual tasks on the project plan, especially in the common sequence:
  - **Identify** (e.g. the entities, the classes, the modules, …)
  - **Design** or specify (each of the things identified)
  - **Develop** (a thing that was designed or specified)
Alternatives to the phased life cycle

- Although the benefits of a phased life cycle have been demonstrated many times over, some current writers promoting newer methodologies disparage it.

- Alternatives proposed include:
  - Incremental or iterative development
  - Rapid application development ("RAD")
  - Visual application design

A canard: the waterfall model of system development

- A "straw-man" term of derision, used mainly to attack the phased life-cycle concept.

  by whom?

- The reference is to the inability of water to flow uphill; i.e. once a phase is done, you're committed and can't go back.

- But no enlightened SDLC advocate seriously proposes such extreme inflexibility.

Anti-life-cycle criticisms

Claims put forth in recent articles and presentations

- "You have to realize that [...] is a new paradigm and the old approaches no longer apply."

- "When we start a [...] project no one can predict where it will lead or how long it will take. After each incremental step, we just decide where to proceed next."

- "With [...] development goes so quickly that there's no need to take extra time to define the problem or agree on specifications. If we get it wrong, we just do it over."

  What new methodologies/technologies can we substitute in each sentence?

  Are any of those assertions valid?

21st Century System development

*We've adopted an incremental approach to developing our applications!*  
*Then why do all our systems still turn out to be excremental?*