Programming in the large

Bertrand Meyer

Lecture 21:
Software lifecycle models

Arguments for the waterfall

(After B.W. Boehm: Software engineering economics)

- The activities are necessary
  - (But: merging of middle activities)
- The order is the right one.

Problems with the waterfall

- Late appearance of actual code.
- Lack of support for requirements change — and more generally for extendibility and reusability.
- Lack of support for the maintenance activity (70% of software costs?).
- Division of labor hampering Total Quality Management.
- Impedance mismatches.
- Highly synchronous model.

Quality control?

- Analysts
- Designers
- Implementers
- Testers
- Customers
Impedance mismatches

As Management requested it. As the Project Leader defined it. As Systems designed it. As Programming developed it. As Operations installed it. What the user wanted.

Pre-1970 cartoon; origin unknown.

The Spiral model (Boehm)

Figure from: Ghezzi, Jazayeri, Mandrioli, *Software Engineering*, 2nd edition, Prentice Hall.

The Spiral model

M.C. Escher: *Waterfall*

Tasks

Analysts
Designers
Implementers
Testers

Seamless development

Example classes

TRANSACTION, PLANE, CUSTOMER, ENGINE...

Example classes

TRANSACTION, PLANE, CUSTOMER, ENGINE...

STATE, USER_COMMAND...
Seamless development

Example classes

Example classes

Analysis classes

deferred class VAT
inhibit
feature
require
deferred
ensure
and
and
invariant

Seamless development

Use consistent notation from analysis to design, implementation and maintenance.

Advantages:
- Smooth process. Avoids gaps (improves productivity, reliability).
- Direct mapping from problem to solution, i.e. from software system to external model.
- Better responsiveness to customer requests.
- Consistency, ease of communication.
- Better interaction between users, managers and developers.
Single model principle

- Use a single base for everything: analysis, design, implementation, documentation...
- Use tools to extract the appropriate views.

The cluster model

- Feasibility study
- Division into clusters
- Cluster 1
- Cluster 2
- Cluster n

The cluster model: extreme variants (1)

- "Clusterfall"

The cluster model: extreme variants (2)

- "Trickle"

Generalization

- Prepare for reuse
- Possible tasks:
  - Remove built-in limits
  - Reorganize inheritance hierarchy
  - Abstraction (e.g. introduce deferred classes)
  - Improve documentation

Cluster development

- Bottom-up development: from the most general clusters (providing utility functions) to the most application-specific ones.
- Flexible scheduling of clusters – depending on resources, team experience, customer and management demands. Waterfall is one extreme; "trickle" is the other.
- Sub-lifecycle sequencing: specification, design and implementation, validation, generalization.
- Relations between clusters: each cluster may be a client of lower-level ones.
Quality goals: the Osmond curves

Other qualities

Desirable

Common

Functionality

Envisaged

Release

The advice

- Add functionality at constant quality

Complementary material

- OOSC2:
  - Chapter 28: The software construction process

End of lecture 21