CS446
Software Engineering

Richard Bartels

University of Waterloo
Faculty of Mathematics
Department of Computer Science

September 13, 1996

CS446
Software Engineering Process

Richard Bartels

University of Waterloo
Faculty of Mathematics
Department of Computer Science

September 13, 1996

Outline

- Life Cycle Models
- Waterfall
- Evolutionary Development
- Spiral
- Risk Management
- Visibility
- Planning and Orgainzation

Reference:

This is background material for Chapter 1 of Sommerville

Paradigms

Basic term: *Software Development Life Cycle (SDL)*

◇ ◇ ◇

Models often mentioned are

- Waterfall
  - Strictly sequential
- Evolutionary Development
  - Interleaved building phases
- Spiral
  - Many, regularized building and evaluation phases
- Formal Transformations
  - Mathematically logical specifications migrated carefully into program code
- Reuse-Oriented Development
  - Concentrate on building systems from off-the-shelf components

◇ ◇ ◇

Focus on the first three

Waterfall

The oldest model – phases in strict sequence

- System engineering
- Followed by analysis
- Followed by design
- Followed by implementation
- Followed by testing
- Followed by delivery and maintenance

At each phase it is possible to backtrack to any previous phase, in extreme circumstances, in order to recover from a difficulty.

Evolutionary Development

Preliminary exploration followed by development
Interleaved with customer interaction
Event driven
Early stages may be built as throw-away prototypes

- Gather requirements
- Follow by quick design
- Build a prototype
  - Possibly with a special system
    interpretive or different language or ...
  - Possibly incomplete
    Simulate functionality or only user interface or ...
- Evaluate and assess
- Develop the product
  - May shift from prototyping to production system
  - Or prototype system may be suitable
    - ▹ Refine the prototype

After assessment, it is possible to backtrack to the quick design, refine it, and rework the prototype for another assessment.
4GL addresses the transformation from easy prototyping system to production system (automatic generation).

Spiral

Gradual, iteratively refined progress from prototype to product.

`while(` not satisfied `) do`

- Gather user input/feedback
- Design/plan the next refinement
- Analyse risks and plan resources
- Implement/modify the product

Each time risk analysis is carried out, may abort the project.
All but (possibly) initial iterations use production system.

Risk Management

No matter what model is used, one should

- Estimate risk
- Carry on activities so as to minimize risk

The most fundamental wisdom

- Look before you leap

The more advanced preparation and preparation review employed, the more probable that risks will be anticipated early

The most fundamental truth

- Better sooner than later

Problems get more expensive the later they are discovered and addressed in any life-cycle model

Remark

The spiral model is notable in providing explicit activities to consider risk.

Visibility

For the purpose of management we need oversight and tracking

Where are we now?
What comes next?
Are we on track?
Are we meeting specifications?
Are we within budget?
Is something starting to go wrong?
If so, precisely what?
Do we have corrective measures?

Conventionally handled through documents, reports, reviews

Deliverable-oriented approach
Every activity ends with a document and legacy

The organization of some life-cycle models lends themselves more readily than other models to visible records

Typical Deliverables and Legacy

- Requirements analysis
  - Feasibility study
  - Outline of requirements
- System specification
  - Functional specification
  - Acceptance test plan
  - Draft of user manual
- Testing
  - Test scaffolding
  - Test automation control
  - Test reports
- Etc.

Visibility of Models

- Waterfall
  - High visibility
  - Clearly separate activities
  - Deliverables are defined
  - Resources allocated for deliverables
- Evolutionary development
  - Poor visibility
  - Unstructured progress
  - Rapid activity
  - Deliverables hard to plan
  - Rapid pace inconsistent with documentation
- Spiral
  - Moderate to high visibility
  - Documents scheduled for each trip around the loop

Remarks

- Models not always applied in pure form.
- Model used is distinct from many other aspects
  - Management/corporate style
  - Testing plan/structure
  - Quality/progress tracking
  - Delivery/service
  - Team organization and guidance
  - Technicalities of risk/resource assessment
  - Maintenance staffing
  - Legacy management and version control
  - Information/archive access/provision
  - and on, and on . . .

Planning and Organization I

"The major benefit of any software engineering ritual is that it probably forces you to think about what you are doing so you may catch problems early. It might not matter so much what you do so long as you at least do something."

Five example rituals (documents); usage of any or all depends upon the size of the project, the size of the company, demands of the customer, certification requirements, etc.

1. Requirements specification

   Customer and company systems analyst
   (precise description of the system's goals;
   proposes a contract between company and customer)

2. Risk assessment

   Company management
   (estimation of effort, pitfalls, and costs;
   supports decision to enter the contract or not)

3. Design specification

   Software project leader and systems analyst
   (high-level organization of system;
   serves as a bridge from requirements to implementation)

Planning and Organization II

4. Test plan

   Test/verification project leader and software project leader
   (assessment procedures for acceptability;
   determines whether the system meets the requirements)

5. Development schedule

   Company management and project leaders
   (layout of times, resources, responsibilites, deliverables;
   serves to track progress, catch difficulties, prioritize goals)

Even advanced SE tools don't circumvent such rituals. Software, at whatever level of abstraction, still has to be planned and written.

Recently, independent standards organizations have been set up to evaluate various rituals, suggest norms for them, train, audit and certify companies;

IEEE, ISO, SEI.

Certain customers may demand such certification;

European contractors (ISO 9000), US Government (CMM).