COMP354

Software Engineering

Lecture 7

Software Process Models

Software Process Models

A **process model** is a description of a way of developing software.

A process model may also be a methodology for software development.

- code-and-fix (obsolete — only for small scale)
- waterfall — documentation driven
- evolutionary — increment driven
- transformational — specification driven
- spiral — risk driven

A process is divided into **phases** or stages

Each phase has

- *activities* to be performed, including
  quality reviews,
  collecting cost and time data,
  collecting defect data from quality reviews
- *input* products, including
  process description, and standards
- *output* products (internal and external) to be produced
  (NB the audience for each document)
- *entry condition*
  under what circumstances can phase begin
  i.e. do you have enough reliable input to begin
- *exit condition*
  under what circumstances is this phase finished
  i.e. do deliverables pass quality control

Waterfall Model

The process is a sequential linear flow among the phases

deliverables produced by a phase are input to following phase

(may allow limited feedback between adjacent phases)

Advantages

- simple to understand
- **phases** are important even if their sequence is not
- works for well-understood problems
- tangible milestones — keeps managers happy
- forces one to complete requirements before design, design before code

Disadvantages

- limited iteration
  does not allow for changing requirements
  forces use of inaccurate or incomplete documents
- does not work for novel or poorly understood problems
- does not allow for overlap in the phases
- plethora of documents lead to "bureaucratic" project management with more concern for the existence/size of documents than their content

Evolutionary Model

Evolutionary model is **increment driven** and **cyclical**:

1. deliver something
   (this is the *increment*)
2. measure added value to customer
3. adjust design and objectives based on the observed realities

An evolutionary process model is one whose phases consist of expanding increments of an operational software product, with the direction of evolution being determined by operational experience.

Evolution often requires **prototypes**.

A *prototype* is a preliminary instance of a system that serves as a model for the final, complete version of the system.

More on Prototypes

Throwaway Prototype

A throwaway prototype is not part of the final product.
Throwaway prototypes should:

- be fast to build
- help to clarify requirements and prevent misunderstanding
- warn implementers of possible difficulties

Once prototype has done its job discard the prototype:
it was quickly done and does not meet quality standards
it may have been written in another language (eg 4GL, Prolog, Smalltalk)

Evolutionary Prototype

Evolutionary prototypes become part of final product.

1. Develop a system that meets a well-understood (and possibly small) subset of the requirements.
2. Deliver the system and obtain feedback from the client.
3. Choose next-best understood requirement and work on that.

Incremental Prototype

Requirements are known in advance, but system construction is done as a sequence of prototypes.

- fewer surprises
- early delivery of some functionality
- more customer involvement

Spiral Model

The **spiral model** is Barry Boehm's formalization of the evolutionary model.

The spiral model envisaged by Boehm has four phases:

1. Identify objectives, alternatives, and constraints.
2. Evaluate alternatives and assess risks.
3. Develop according to established objectives and verify that these objectives are met.
4. Review results obtained during the current cycle. Plan another iteration if required.

Waterfall Model is roughly "once around the spiral".
A typical industrial-scale project requires from three to six iterations.

The spiral model is based on **risks**.

A *risk* is a potentially adverse circumstance which may impair the development process or the quality of products.

In **risk management** we identify risks and respond to them before they endanger the whole project.

1. identify the risk
2. assess their likelihood and potential impact
3. address the risk by devising a plan to deal with it
4. eliminate the risk by implementing the plan

Comparison on Models

- Waterfall development provides: good management of the process; poor response to clients; a large final product; a short test phase.
- Spiral development provides: short development time; good response to changes in requirements; a small final product.
- Consensus: the spiral is better than the waterfall, especially for products that are not well understood.
- "In the old days, we **wrote** software; then, for a while, we **built** software; nowadays, we **grow** software" (Brooks, 1978)

OO methodologies adopt a spiral process model based on either evolutionary prototypes (for novel products) or incremental prototypes (for well-understood products)