Iterative Models

CS-370: Software Design and Development

Robert Bruce

Predictive versus iterative model

- **Problem:** Predictive models cannot handle unexpected changes in a project. Furthermore, there's no room for adaptability.
- **Solution:** Utilize an iterative model instead.

Source: page 446, *Beginning Software Engineering* (2nd edition) by Rod Stephens

What is an iterative model?

- Iterative models build applications incrementally.
- Each increment is a short duration and generates a new prototype.
- Each increment incorporates new features.
- Each increment allows for requirements adjustment (e.g. due to a vague requirement).

Source: page 447, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Prototypes: two important facts

- “[Prototypes] do not need to work the same way that the final application will work.”
- “[Prototypes] don’t need to implement all the features of the final application.”

Source: page 449, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Three types of prototypes

- **Throwaway prototypes:** a temporary prototype created for evaluative purposes. This prototype (and the software to create the throwaway prototype) is discarded once the evaluation is complete.
- **Evolutionary prototypes:** demonstrates some of the features in the product and is refined over-time until the project is completed.
- **Incremental prototypes:** a modular approach in which each feature is built into a separate prototype. The functionality of the prototypes – the software – is then combined to produce the finished application with all of the features combined.

Source: page 451, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Prototyping: advantages

- **Improved requirements:** gives customers an opportunity to provide feedback to the developers.
- **Common vision:** allows one to assess how well the customer and developers vision align.
- **Better design:** provides alterative methods to assess and improve the product.

Source: page 451, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Prototyping: disadvantages

- **Narrowing vision:** The prototype can distract developers by focusing on the problem at hand rather than alternative or better solutions in general.
- **Customer impatience:** Customers can easily confuse the prototype with the finished product.
- **Schedule pressure:** Customers may believe the prototype signifies the project is close to completion and become impatient with the present completion schedule.
- **Raised expectations:** The prototype may include features the customer didn't ask for but now expects the developer to include in the final product.
- **Attachment to code:** Developers may be tempted to re-use low-quality prototype software in the finished product even if improved software exists.
- **Never-ending prototypes:** Iteratively refining a prototype with new features that aren't actually necessary.

Source: page 452, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Spiral model development cycle: four phases

- Phase 1: “Determine objectives, alternatives, and constraints”
- Phase 2: “Risk analysis. Evaluate alternatives. Identify and resolve risks. Build a prototype.”
- Phase 3: “Use the prototype to perform simulations and model problems. Fix problems and produce a result.”
- Phase 4: “Plan the next iteration.”

Source: page 453, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Spiral model development cycle: clarifications

- The spiral model is not a waterfall approach.
- The activities in the spiral model need not be unified in one master spiral. Instead, each activity can be in its own separate spiral.
- The ordering of activities can be changed for each spiral as well, depending on the project needs.

Source: page 453, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Unified Process

- An iterative and incremental development framework
  - Phase 1: Inception: define the project idea.
  - Phase 2: Elaboration: define the project requirements
  - Phase 3: Construction: software codification (write, test, and debug)
  - Phase 4: Transition: project maintenance

Source: page 457, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Cleanroom model

- Emphasis: “defect prevention rather than defect removal”

Source: page 460, *Beginning Software Engineering* (2nd edition) by Rod Stephens

Cowboy coding

- A carte blanche approach which maximizes developer autonomy.

Source: page 461, *Beginning Software Engineering* (2nd edition) by Rod Stephens