Monday June 6, 2016

The Software Engineering Process (software lifecycle)

*“A series of steps involving activities, constraints, resources that produce an intended output”*

* **Activity** – A type of work performed during the process
* **Constraint** – A restriction imposed on the process (i.e., deadlines, a schedule, money, target platform)
* **Resource** – Input consumed during process activities (i.e., labor, time, and money)
* **Output** (product) – A result created during one or more process activities (i.e., specifications, executable software)

Process Characteristics

* Prescribes the major activities and when each occurs
* Can produce intermediate product and final product
* Can be composition of mini-processes
* Each activity has entry and exit criteria (clear when it starts and stops)
* Sequential (know which activity is next)
* Each activity has goals
* Constraints can apply to one or more activities, resources, products

Why use a software engineering process?

* **Repeatability** – Consistent products of desired quality.
* **Efficient Training** – Easier to train new people with a well-documented process (and again, it’s more repeatable)
* **Room for Improvement** – Easier to measure, analyze and improve parts of the process to yield better results.

Process Activities

1. Persistent data (usually via a database)
2. Requirements analysis and definition
3. System Design
4. Program Design
5. Programming
6. Unit/System/Acceptance testing
7. Delivery
8. Maintenance

Work Product

* Requirements
  + What the system is supposed to do in terms of behavior, data and constraints.
  + Language: plain and anything
  + Produced by clients, end-users, etc
* Specifications
  + Developer interpretation of what the system is supposed to do
  + Extracted from the requirements
  + Language: structured plain
  + Produced by developer, requirements engineer, etc
* Design
  + Blueprints of how the system will be built
  + Language: model notation, diagrams, prototypes
  + Produced by developer, architect
* Implementation
  + The working software product
  + Handwritten and/or COTS (Component off the shelf)
  + 3rd party libraries, frameworks, databases
  + Language: executable
  + Produced by developer, programmer, integrator, testers

Process Model

*“A specific configuration of process activities that can guide real software development.”*

* Provides a roadmap for development
* Documenting creates common understanding
* Helps find bugs in the process, even before dev starts
* Model should match the special needs/circumstances of the real project.
  + *No one model is an answer to all software development*

Waterfall Model

* Linear sequence of activities
  + Proceed one activity at a time and no going back
* The prescriptive process for several decades
* Part of DoD standard for Software Engineering
* Use when all requirements are known (not realistic)
  + Can be extremely efficient (minimal context switching)
* Cons:
  + Clients rarely know all requirements upfront
  + Doesn’t allow any change
  + Cannot use it until software is done
* Risk:

*The chance that something bad will happen (i.e., an activity’s work product is not 100% correct)*

* + Waterfall requires an activity to be 100% correct/finished before moving to next activity
    - How do you know if an activity is done and done right?

Quality Terms

* **Validation** – check if all requirements are accounted for (Is this what the customer wants)
* **Verification** – everything works correctly (i.e., according to the specifications)

V Model

* Variation of the Waterfall Model
* Close couples different levels of testing with analysis and design
* Divided into levels and sides

Other Models

1. Prototype Model
2. Operational Specifications Model
3. Incremental Model
4. Spiral Model