Wednesday June 1, 2016

Software Engineering Introduction

* Good understanding of the Software Engineering process
* Useful tools, techniques (methods), and experience
* Academic and Industry value

What is Software Engineering?

* Understanding the problem and the requirements
* Testing and insuring that the product works (and it’s truly what the client wants)
* Maintenance

*“The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.”*

Systematic – There is a step by step process

Disciplined – It’s controlled

Quantifiable – It’s measurable

How is Software Engineering different from Computer Science?

Software vs. Enterprise Software

Enterprise Software

*Software that is built for a specific business process. Can be large, medium, or small software systems.*

* Persistent data (usually via a database)
* Accumulate a lot of data (database size can grow quite large)
* Multiple, concurrent users and lots of screens
* Integrates with other enterprise systems and data
* Involves behavior that is specific to the business process

Software Systems

* Software products usually do not operate in isolation (i.e., without interaction with external entities)
* **Entity** – A human or other software/system that interacts/interfaces with a software program.
* **Boundary** – Determines what is included in the development of the software product (entities outside the boundaries not part of the development effort but produce input for the system and/or consume its output)
* To better understand software, describe its parts
  + **Behavior** – Something the software does (function, use case, etc)
  + **Objects** – data used by the activities; can be records, classes/templates, fields, etc
  + **Relationships** – which objects are used in which activities
  + **Boundary** and **external entities**

Breaking Down Complexity

* **Abstraction** – A simplification that allows focus on some pertinent aspect of the software; a model (e.g., a blueprint of a home showing electrical wiring)
* **Modularization** – Divide and conquer; break entire behavior into independent, logical parts; design/build/test/integrate each part (sometimes in parallel). Make things sensible.

Software Engineering Activities

**Activity** – A type of work performed during software development

1. Requirements analysis and definition
   1. Creation of specifications
   2. Understanding what the software needs to do
2. System design
   1. Understanding which parts of the system are going to go where
   2. High-level design
3. Program design
   1. Identifying which functional goes into which functional units (classes, libraries, moduls)
   2. Little lower-level design (start to compose functionality)
4. Programming
5. Unit/Integration/System Testing
   1. **Unit** – Testing at the functional level
   2. **Integration** – Start to bring components together
   3. **System Testing** – Normally when you take the software system and test it in an environment as similar as possible as to when it goes live. Includes performance testing (Does the application respond in a reasonable amoutn of time).
6. Delivery
   1. Once it goes live and real-end users get there hands on it
7. Maintenance
   1. Bug fixes, enhancement, patches, new updates, switching platforms
   2. What will the next version look like? (you can’t get too comfortable with your tools)

Software Engineering Process

* A **process** is an approach for the engineering of a particular software product (i.e., a specific configuration of the SE Activities)
* Encompasses the approach AND the tools/methods that are within the approach to build the software
* The process should suit the project, stake holders, budget, etc.
  + Budget – Wanting high quality software (what are the costs?)

Stakeholders

*Anyone who benefits from the software being produced (i.e., has a stake in the success of the software)*

Can be:

* End-users (customers)
* Client
* Developers:
  + Programmer
  + Testers
  + Analysts
  + Designers/Architects
  + Managers
  + Trainers
* Anyone who is involved in the production of the software
* Investors (client and developer)

Quality

* A very important characteristic of good Software Engineering
* Lots of different ways to judge software quality
  + User View
    - How well it satisfies the requirements and is it defect free
  + Manufacturing View
    - How well the software conforms to it’s specifications and it’s production process
    - Does the software do what its specs say it does?
    - Assumes the specifications are correct
  + Product View
    - Gages product on level of documentation, maintainability, performance
  + Value Views
* Ultimately, quality is not Boolean and should be a customer-dictated requirement.
* Not Perfect, but ultra-high quality (NASA space launch example)
  + 1/3 of development occurred before any code
  + 40,000 pages of specifications
  + Adding new GPS feature caused 2,500 more pages
  + Specifications are almost pseudo-code
  + **Remember** – a good design leads to good implementation

Bug Terminology

* **Fault/Defect/Bug** - A human error while performing a software engineering activity
  + **Example**
* **Failure** – When the program is doing something it is not supposed to be doing

Does every fault cause a failure? Vice versa?

No, not every fault causes a failure. If there is no observable change of incorrect behavior, it might not be a failure.