Basics of Software Engineering

CS134

Chris Pollett

Aug. 30, 2004

Outline

- The Constraint Triangle
- Requirements and Specification
- Software Engineering Process
- The Software Lifecycle
- Managing a project
- Teams

The Constraint Triangle

- Cost -- number of programmer on team
- Time -- how long until project is finished
- Quality -- number of features and how extensively they are tested
- Usually can decrease two out of three

Requirements and Specification

- To start a project need the requirements of the program needed. Can think of requirements like requests for features
- Given these requirements we need to then produce a brief description or specification of what the software will be.
- Often communicate between various interested parties in a software project using some diagramming language. Ex UML. Use case diagrams.

What is a use case diagram?

- Program is a rectangle. Have actors who want to do things with this program. Represented as stick figure. Each of these stick figures activities with the program is drawn with a bubble with a word in it and is called a use-case.

Software Engineering Process

- Requirement and Specification (already talked about)
- Schedule
- Design
- Project document

Schedule

Some components of this include:

- Lifecycle -what to do when
- Milestones - some definite tangible goals which should be reached by some fixed date. Ex. Specification sketch by Sept 6. (Dates can be revised.) Often have sub-milestones or task lists.
- QA plan -- quality assurance. How things will be tested.
- Risk Management -- trying to anticipate how things might cause you to go off schedule. Need to monitor progress to know if going off schedule and need to have a mechanism for recovery if this happens

Design

- High level design -- the architecture of your project (might model the way the different classes are organized using UML)
- Detailed design -describes the order in which specific things happens. Ex: Sequence of events to update a character.

Project Documents

- Written specification
- Scheduling -- milestone list, risk list
- Design -- class diagrams, sequence diagrams, class headers
- User’s guide for your software

The Software Lifecycle

Model of what happens as project develops

- Classic model called Waterfall lifecycle - Requirement -> Specification -> Architecture-> Detailed Design -> Coding -> Testing and Debugging -> Ship
- Stage Delivery lifecycle Software concept -> requirement dev (n stages) -> Architecture and high level design-> Stage 1 design code test debug-> Stage n design code test debug ->Final release

More Lifecycles

- Inventor Lifecycle -- Requirements gathering -> architecture -> Spec n and Detailed design n -> Alpha n program and user’s guide -> final design and feature freeze -> beta N program and user guide -> test and debug beta n ->Ship

Managing Projects

- Need to be able to track builds so know what version you are working with and what version things have been tested for. Try using a revision control system if that’s possible
- Code should be easy to read and follow some common conventions for everyone on the project.
- Hard algorithms or subtle points about sections of code should be commented

Teams

- Each person should have contact info for other team members (phone and email addresses)
- Should practice merging each other code, so people will know what will happen during a merge/build
- Member roles should be decided upon.
- Meetings should be held where demos of things are done