Process: Goals, Risks, Estimation, Planning, and Metrics

15-413: Software Engineering Practicum

Jonathan Aldrich Charlie Garrod

Outline

- **Goals – Picture of Success**
- Risk Management
- Estimation
- Planning
- Metrics

Picture of Success

[source: Gil Taran]

- The minimum set of conditions that must be met for project members to consider the project a success
  - A standard against which to measure risks
  - Not the set of all goals

- Characteristics
  - Set at specific time in future
  - Measurable
  - Agreed to by team
  - Short – e.g. one slide with 4-5 statements

Common failure modes

- Your Picture of Success should NOT:
  - Discuss your current status
  - List vague goals that you cannot easily measure or evaluate
    - Measurability is hard for non-technical goals, but think creatively about this
- Your Picture of Success SHOULD:
  - Include personal and learning goals as well as objectives related to completing the course or project

Example Open Source Picture of Success

- **Become committers:** In our project, until we prove ourselves we are only able to submit patches. Once we prove that we can submit good code we will be promoted to committers.
- **Aid in design decisions:** As opposed to simply fixing bugs the whole time, we would like to quickly get a deep enough understanding to be able to actively participate in design decisions.
- **Fix a few hard bugs:** Bugs are labeled by expected difficulty. While we are not yet sure exactly what this entails, we aim to fix at least one, and hopefully more, of the bugs labeled hard.
- **Contribute and develop an original idea:** We would like to come up with an improvement or new feature that we will both initiate and bring to completion.

Example Bullets – Open Source Project

- Our team shall share the workload evenly based on the tasks assigned to each individual in our team meetings
- Learn and follow the development standards and contribution process used by the OpenBaz project, and become proficient in the tools used by the project
- Document progress by maintaining a blog with weekly progress posts and time tracking. Update design documents and carefully document design goals, algorithm details, and changes.

Example Bullets – Open Source Project

- Beat the other schools by contributing better and cleaner code
  - Good documentation
  - Clearly written commit message
  - Well thought-out designs using applicable design patterns
- Learn best practices for contributing to the open source community and committing some code within 2 weeks of the hackathon

Exercise: Meet and Discuss Goals

- Introduce yourselves by project
  - List on next slide
  - Say your name, something about yourself, and something you hope to get from this course
- Move to sit with your project partners
- Talk a bit more about your course goals

Picture Of Success – Course Project

[source: Gil Taran]

- We deliver the "must have" requirements as agreed by us and the client by the end of the semester with the levels of quality specified by the client.
- The team shares the workload evenly and collaboratively throughout the project and resolves conflict through timely team communication.
- We have a designated process that is thoroughly documented and followed throughout the project.
- We periodically, at a minimum once a month, review our actions and processes so as to identify actions that get implemented in the next phase.
- We are able to articulate core principles in the areas of people, process, and technology, and reflect on having used them in our project so as to understand our successes and failures and react accordingly.

Project List

- Kotlin
- MongoDB
- Mozilla Firefox O/S
- Mozilla Marketplace
- Review Board
- App Inventor
- JTS
- OpenStack
- Prediction.io
- SocketIO
- Umple
- KDevelop for Ruby

Picture of Success in 15-413

- Come up with a team Picture of Success
  - Due 1/20, follow the guidelines above

- Rationale
  - Structured way to think about goals beyond completing each assignment
    - What you get out of this course depends on your investment
  - Help focus your activities towards achieving your goals
  - Help each other and the instructors understand what you want out of the course

15-313 Review

- If you have taken 15-313, and are comfortable with:
  - Condition-consequence risk statements
  - Wideband Delphi estimation
  - Earned Value / Velocity

then you may leave now.

If you leave, please look at the slides with 15-413 in their title online, in order to see how we are using these concepts in this course.

The next lecture will be Tue, Jan 21

Outline

- Goals
- **Risk Management**
- Estimation
- Planning
- Metrics

An Example of Risk

[source: Gil Taran]

Risk Defined

[source: Gil Taran]

- The possibility of suffering loss
  - Webster's dictionary, 1981

- All definitions share the following characteristics:
  - **Uncertainty** - an event may or may not happen
  - **Loss** - an event has unwanted consequences or losses

Risk Management

www.dilbert.com

- Anticipating risks so they are not a surprise
- Plan response to risk
- Decreasing the magnitude of a potential loss
- Decreasing the chance of loss
- Increase control over the risk

Defining Risks

[source: Gil Taran]

- Condition
  - Phrase describing some factual condition of the project
    - May be positive or negative
    - Example: *There is water on the floor*
  - Should ***NOT*** be a hypothetical statement
    - *WRONG:* "Water might spill on the floor"
    - *RIGHT:* "There is a faucet in the room"

- Consequence
  - Potential negative consequence of the condition
  - Example: *Someone might slip in it and get hurt*

Risk Statements

- What are risks to your picture of success in this course?

Risk Analysis

- Impact: the severity of the loss
- Probability: the likelihood of the loss
- Exposure: Impact * Probability
  - A measure of priority
- Time frame: how long until risk materializes?
  - Urgency combines priority and time frame

- Mitigation
  - Approach to reduce impact or probability of a risk
- Periodic monitoring
  - Identify new risks
  - Increase or decrease in impact or probability

Risk Management in 15-413

- We will not have a formal process
- However, we will ask you to think about risks in our weekly meetings
- Use the ideas above to guide your thinking

Outline

- Software Engineering Minor
- Risk Management
- **Estimation**
- Planning
- Metrics

Estimation is Difficult

- Average project exceeds cost, schedule estimates by 100% - 1998-2000 Standish Report

- Factors
  - Complex systems are hard to estimate
  - Problems look easy until you see the detail
  - We never build the same thing twice
  - Management pressure affects estimates

Estimation Basics

- Cost = person-months * cost-of-person
  - cost-of-person includes benefits, taxes, equipment, support staff, and building
    - may be 2-3 times salary
  - Which factor is more uncertain?

Estimation Principles

- Ultimately based on experienced judgment
  - Structuring techniques may improve accuracy
- Principles
  - Use historical data
  - Divide and conquer
  - Many points of view
  - Correction over time

Estimation: Historical Data

- Find similar projects with cost data
  - Domain
  - Size
  - Architecture
- Adjust for differences
  - Project size/scope
  - Improved expertise
  - New/unknown problems
  - Reuse of old artifacts
    - Note: reuse is not free! Adaptation is required

Estimation: Divide and Conquer

- Work Breakdown Structure
  - Divide hierarchically into tasks
- Develop conceptual design
  - Break into parts
  - Only for estimation: should redo entirely in real design phase!
- Estimate tasks/parts separately
  - Combine estimates
  - Recognize costs for integration

Estimation: Wideband Delphi

[source: Mel Rosso-Llopart]

- Planning
  - Define the scope of the problem
  - Break large problems into smaller
- The Kickoff
  - Deliver problem to team
- Individual preparation
  - Everyone does individual estimates on problem parts
  - All assumptions are written down
- Estimation Meeting
- Assembling Tasks
  - Put together the estimates of team members
- Review Results as team

Wideband Delphi: Estimation Meeting

[source: Mel Rosso-Llopart]

- A moderator collects the estimates for the task being estimated
  - Present the average or a line with all estimates (anonymous)
- The estimate is discussed and assumptions presented
- Moderator calls for a new estimate from everyone
- Values are again presented to the team as average or line
- Continue process until:
  - Four rounds are completed
  - The estimates "converged" to an acceptably narrow range
  - The allotted meeting time is complete
  - All participants are unwilling to change their estimates
- 15-20 minutes per item discussed

Rounds in Delphi

[source: Mel Rosso-Llopart]

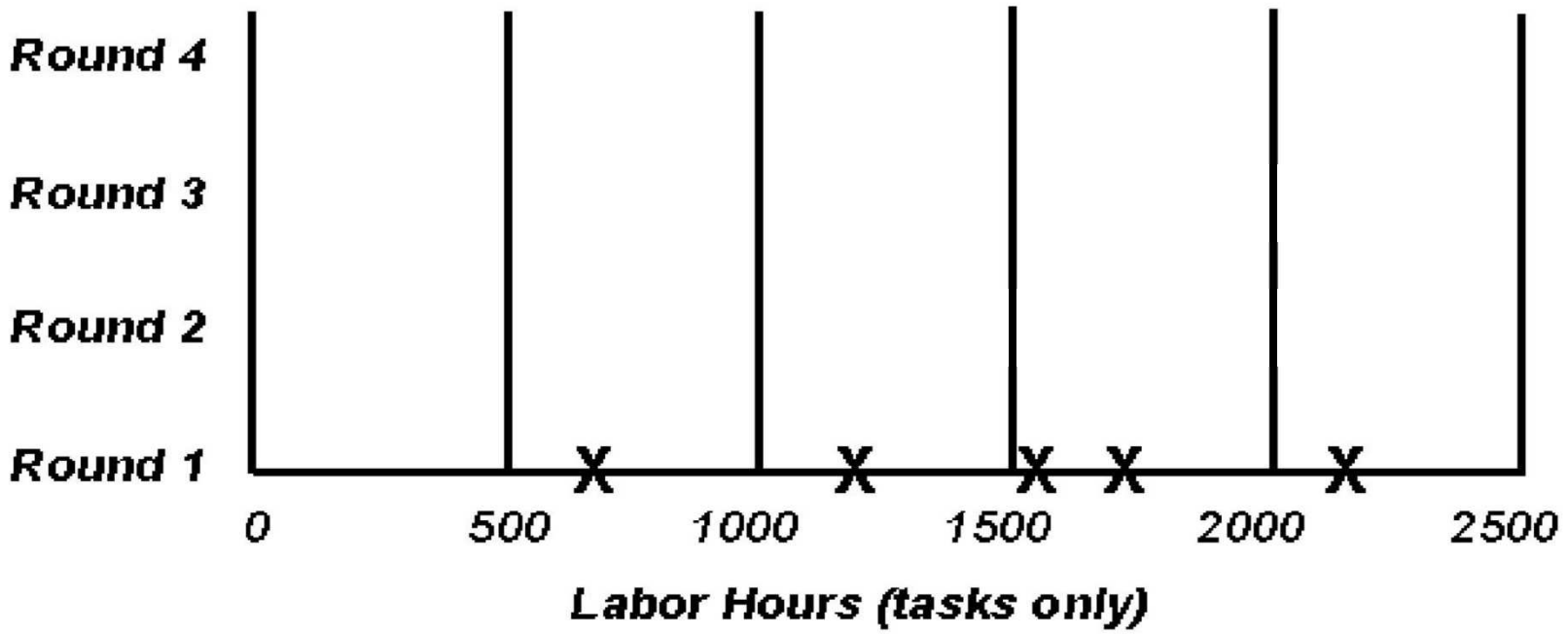

Figure 4. Estimation chart showing three rounds from a Wideband Delphi session.

Wideband Delphi: Best Practices

[source: Mel Rosso-Llopart]

- Gather a heterogeneous team
- Write down assumptions
- Make anonymous estimates
- Never estimate tasks larger than you are comfortable with

Estimation: Correction over Time

- Last task estimated cost: 10 hours
- Last task actual cost: 20 hours

- Next task estimated cost: 15 hours
  - How long will it actually take?

- XP: load factor
  - Developers estimate tasks in “ideal time”
    - Time with no meetings, questions from buddies, etc.
    - Multiply all estimates by empirically measured “load factor”
      - load factor = actual / estimate = 20 / 10 = 2.0
      - New task estimate = ideal * load factor = 15 * 2.0 = 30

Estimation: Function Points

[source: Alvin Alexander, devdaily.com]

- Standard unit of measure for software size
  - In terms of requirements, not code

- Data Functions
  - Internal logical files
    - Database table, file, preferences
    - # FPs depends on record, field structure
  - External interface files
    - Data the app needs but does not maintain
- Transactional Functions:
  - External Inputs
    - User data entry, automatic data feeds
  - External Outputs
    - Output of computed or processed data
  - External Inquiries
    - Output of retrieved data

From FPs to Effort

- Historical data
  - Cost per FP on similar projects
  - May need fudge factors to account for differences
- COCOMO
  - Adjust function points based on project characteristics
    - Product: reliability, complexity, reuse, …
    - Platform: performance, storage, change
    - Personnel: capability, continuity, experience
    - Project: tools, distributed dev., schedule
  - Convert to person-hours based on historical data

Estimation in 15-413

- Estimate each technical task before you do it
  - For small tasks, just make the estimate yourself
  - At least once, use Wideband Delphi
    - Ideally for a larger task done by >1 person
  - Adjust estimates according to history
    - Use load factor
    - Or simply compare to similar previous tasks
  - Once you start a task, ***do not change the estimate!***
    - Even if you know you under/over-estimated
    - Otherwise you will not have data to adjust future estimates

- Estimation is not necessary for course assignments

Outline

- Software Engineering Minor
- Risk Management
- Estimation
- **Planning**
- Metrics

Planning

- Choosing in what order to do things
- Inputs
  - Cost determined by engineers
  - Priority determined by customer
- Ordering Principles
  - Deliver a working system early
    - Biggest risk: not delivering anything
  - Deliver customer value early
    - Agile: customer can change his mind!
  - Mitigate risks early
    - Beware: New risks may come up
  - Respect dependencies & critical path

Planning in 15-413

- Determine how (and whether) to do planning in conjunction with your open-source mentor
  - May be informal in practice
  - Still useful to think of the criteria above

Outline

- Software Engineering Minor
- Risk Management
- Estimation
- Planning
- **Metrics**

Metrics: How are we doing?

- Scenario: contract work
  - Your company has agreed to produce program X
  - Program X has two parts. You've estimated effort, cost and time for each:
    - Part A: 160 hours, $3200, delivery at 2 week point
    - Part B: 120 hours, $2400, delivery at 3 week point
  - Now you are 3 weeks into the project. You have only delivered Part A. You have spent 200 person-hours on it. How are you doing?

Metrics: How are we doing?

- Earned value
  - A measure of how much value you have delivered to the client
- Back to our contract scenario
  - Estimates
    - Part A: 160 hours, $3200, delivery at 2 week point
    - Part B: 120 hours, $2400, delivery at 3 week point
  - Reality:
    - Part A: 200 hours, ???, delivered at 3 week point
    - Part B: no progress yet
- How much earned value?
  - 200 hours → $4000 (at $20/hour)?
    - But it's hardly fair to the customer if you are slow!
  - A better answer: $3200
    - That's what you originally estimated as the cost of the work you actually delivered
    - It's also how much your client promised to pay you for it!

Metrics: How are we doing?

- Earned value
  - A measure of how much value you have delivered to the client
  - The estimated cost (typically in hours, or dollars) of the work you actually finished
    - In some methodologies work that is incomplete is pro-rated—but this is risky. You may think it is 90% complete, but how do you know?

Metrics: How are we doing?

- Cost performance index (CPI)
  - Are we under or over budget? By how much?
  - Compare budgeted cost to actual cost

- Crunching the numbers
  - Budgeted Cost of Work Performed (BCWP)
    - BCWP is the sum of the costs in the budget for the tasks we've completed so far
  - Actual Cost of Work Performed (ACWP)
    - ACWP is the sum of the actual costs of the tasks we've completed so far
  - CPI = BCWP / ACWP = 160/200 = 80%
    - We've done 80% of the work we promised we could do in 200 hours
    - Good to be above 100% - indicates efficiency
    - Too far about 100% suggests inaccurate estimation

Metrics: How are we doing?

- Schedule performance index (SPI)
  - Are we ahead or behind schedule?
  - Compare how much work we actually did, to how much work we expected to do
- How to measure work fairly?
  - Use the budgeted cost – just as in earned value
- Crunching the numbers
  - Budgeted Cost of Work Performed (BCWP)
    - BCWP is the sum of the costs in the budget for the tasks we've completed so far
  - Budgeted Cost of Work Scheduled (BCWS)
    - BCWP is the sum of the costs in the budget for the tasks we were scheduled to have completed as of now
  - SPI = BCWP / BCWS = 160 / 280 = 57%
    - We've done 57% as much work as we planned to do
    - Again, good to be above 100%, but too far above is a red flag

Earned Value Exercise

(assume we have just finished Month 1)

Task	Budget	Month	Done?	Actual
1	20	1	N	
2	5	1	Y	10
3	10	1	Y	15
4	10	2	Y	5

- Earned Value =
- CPI =
- SPI =

15-413 Time Tracking and Metrics

- Each week, prepare a time report
  - List all tasks performed by the team
  - If a technical task, write the time estimate
  - Write how much actual time each person spent on each task (to nearest 0.25 hours)
  - Write what % complete you believe each task is right now
  - List tasks you know you will work on next week, with estimates
    - Note: new tasks may be added during the week. Estimate them as they come in, before starting.
- Compute summary information
  - Earned value, CPI, SPI *(see Metrics slides just above)*
  - Current load factor *(see Estimation slides earlier)*

Earned Value Exercise Answers

(assume we have just finished Month 1)

Task	Budget	Month	Done?	Actual
1	20	1	N	
2	5	1	Y	10
3	10	1	Y	15
4	10	2	Y	5

- Earned Value = 5+10+10 = 25
- CPI = 25 / (10+15+5) = 25/30 = 83%
- SPI = 25 / (20+5+10) = 25/35 = 71%