CSC 171 – Module 5
Estimation

Yu Chen

TRADITIONAL ESTIMATION – COCOMO MODEL

COCOMO Suite

- Constructive Cost Model (COCOMO)
  - Predicts effort, time, schedule for software projects
- COCOMO 81
  - For Waterfall process model
  - Retired
- COCOMO II
  - Published in 2000
  - For Waterfall or spiral process models
  - Two models
    - Early design model
      - High-level
      - To explore architectural alternatives or development strategies
    - Post-architecture model
      - Most detailed

Nominal-Schedule Formulas

- Nominal-schedule (NS) formulas exclude the cost driver for required development schedule
  - PM_{NS} = A x $Size^E$ x $\prod_{i=1}^{16} EM_i$
  - E = B + 0.01 x $\sum_{j=1}^{5} SFj$
  - $TDEV_{NS}$ = [C x $PM_{NS}^{(D+0.2x(E-B))}$]
  - A = 2.94, B = 0.91, C = 3.67, D = 0.28
- An example
  - Given Size =100 KSLOC, E = 1.15, effort multipliers are all equal to 1.0
  - PM_{NS} = 586.61 person-months
  - $TDEV_{NS}$ = 29.7 months
  - The number of stuff required
    - PM_{NS} / $TDEV_{NS}$ = 19.75

Sizing

- Code size
  - KSLOC, logical source statements
  - Excludes comments lines and blank lines, code generated with source code generators, commercial libraries, not delivered code, unintentionally present code, …
- Unadjusted Function Points (UFP)
  - Follows definition in [2]
- Converting UFPs to SLOC
  - Published SLOC / UFP for Java is 53, which can be replaced by historical data in local environments
- Aggregating new, adapted, and reused code
  - Reused code, code reused without modification
  - Adapted code, code reused with modification
  - Non-linear reuse effect
    - DM: percentage of design modified
    - CM: percentage of code modified
    - IM: percentage of integration effort required
    - SU: software understanding increment
    - AA: assessment and assimilation
    - UNFM: relative unfamiliarity with the software
    - AAM: adaptation adjustment modifier

Sizing - Continued

- Requirement Evolution & Volatility (REVL)
  - Percentage of the code discarded due to requirements evolution
    - A project delivers 100 KLOC but discards 20 KLOC, REVL = 20
- Automatically translated code
  - AT: percentage of the code that is re-engineered by automatic translation
  - Automated translation is considered to be a separate activity from development, and its effort is calculated separately
- Software maintenance
  - Size of modified code
  - Excludes
    - Major product rebuilds changing over 50% of the existing software
    - Development of sizable (over 20% changed) interfacing systems requiring little rework of the existing system
  - $Size_M$ = ((Base code size) x MCF) x MAF
  - Maintenance Change Factor, MCF = $\frac{Size\ Added + Size\ Modified}{Base\ Code\ Size}$
  - Maintenance Adjustment Factor, MAF = 1 + ($\frac{SU}{100}\ x\ UNFM$)

$$\text{Size} = \left(1 + \frac{\text{REVL}}{100}\right) \times (\text{New KSLOC} + \text{Equivalent KSLOC})$$

$$\text{Equivalent KSLOC} = \text{Adapted KSLOC} \times \left(1 - \frac{\text{AT}}{100}\right) \times \text{AAM}$$

$$\text{where AAM} = \begin{cases} \dfrac{\text{AA} + \text{AAF} \times (1 + [0.02 \times \text{SU} \times \text{UNFM}])}{100}, & \text{for AAF} \leq 50 \\ \dfrac{\text{AA} + \text{AAF} + (\text{SU} \times \text{UNFM})}{100}, & \text{for AAF} > 50 \end{cases}$$

$$\text{AAF} = (0.4 \times \text{DM}) + (0.3 \times \text{CM}) + (0.3 \times \text{IM})$$

• **AA** Assessment and Assimilation • **AAF** Adaptation Adjustment Factor • **AAM** Adaptation Adjustment Modifier • **CM** Percent Code Modified • **DM** Percent Design Modified	• **IM** Percent of Integration Required for the Adapted Software • **REVL** Requirements Evolution and Volatility • **SU** Software Understanding • **UNFM** Programmer Unfamiliarity with Software

Effort Estimation – Post-Architecture

- Post-Architecture Model
  - PM = A x Size^E x $\prod_{i=1}^{17} EM_i$, where A = 2.94
  - $PM_{Auto} = \frac{Adapted\ SLOC\ x\ (\frac{AT}{100})}{ATPROD}$
    - ATPROD: productivity value for automated translation
  - E = B + 0.01 x $\sum_{j=1}^{5} SFj$, where B = 0.91
    - E ranges between 0.91 and 1.226, for a project with 100 KSLOC, PM ranges from 194 to 832
- Early Design Model
  - PM = A x Size^E x $\prod_{i=1}^{7} EM_i$ + PM_{Auto}
- Scale factors
  - Precedentedness (PREC), Development Flexibility (FLEX), Architecture / Risk Resolution (RESL), Team Cohesion (TEAM), Process Maturity (PMAT)

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC **SF_j:**	thoroughly unprecedented 6.20	largely unprecedented 4.96	somewhat unprecedented 3.72	generally familiar 2.48	largely familiar 1.24	thoroughly familiar 0.00
FLEX **SF_j:**	rigorous 5.07	occasional relaxation 4.05	some relaxation 3.04	general conformity 2.03	some conformity 1.01	general goals 0.00
RESL **SF_j:**	little (20%) 7.07	some (40%) 5.65	often (60%) 4.24	generally (75%) 2.83	mostly (90%) 1.41	full (100%) 0.00
TEAM **SF_j:**	very difficult interactions 5.48	some difficult interactions 4.38	basically cooperative interactions 3.29	largely cooperative 2.19	highly cooperative 1.10	seamless interactions 0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
SF_j:	SW-CMM Level 1 Lower 7.80	SW-CMM Level 1 Upper 6.24	SW-CMM Level 2 4.68	SW-CMM Level 3 3.12	SW-CMM Level 4 1.56	SW-CMM Level 5 0.00

Effort Multipliers – Post-Architecture Model

- Product Factors
  - Required Software Reliability (RELY)
  - Data Base Size (DATA)
  - Product Complexity
  - Developed for Reusability (RUSE)
  - Documentation Match to Life-Cycle Needs (DOCU)
- Platform Factors
  - Execution Time Constraint (TIME)
  - Main Storage Constraint (STOR)
  - Platform Volatility (PVOL)
- Personnel Factors
  - Analyst Capability (ACAP)
  - Programmer Capability (PCAP)
  - Personnel Continuity (PCON)
  - Application Experience (APEX)
  - Platform Experience (PLEX)
  - Language and Tool Experience (LTEX)
- Project Factors
  - Use of Software Tools (TOOL)
  - Multisite Development (SITE)
  - Required Development Schedule (SCED)

Effort Multipliers – Early Design Model

– Personnel Capability (PERS)
– Product Reliability and Complexity (RCPX)
– Developed for Reusability (RUSE)
– Platform Difficulty (PDIF)
– Personnel Experience (PREX)
– Facilities (FCIL)
– Required Development Schedule (SCED)

Schedule Estimation

- TDEV, time to development in months
- TDEV = [C x $PM_{NS}^{(D+0.2x(E-B))}$] x $\frac{SCED\%}{100}$
  - C=3.67, D=0.28, B=0.91, can be calibrated
  - PM_{NS}, estimated PM without SCED effort multiplier
  - SCED, required schedule compression
- Applicable process models
  - Between LCO and IOC for MBASE/RUP process model
    - MBASE: Model-Based (System) Architecting and Software Engineering
    - RUP: Rational Unified Process
    - LCO: Life Cycle Objectives
    - LCA: Life Cycle Architecture
    - IOC: Initial Operational Capability
  - Between SRR and SAR for Waterfall process model
    - SRR: Software Requirements Review milestone
    - SAR: Software Acceptance Review milestone

Software Maintenance

- SCED (Required Development Schedule) and RUSE (Required Reusability) are not used in effort estimation
- RELY (Required Software Reliability) has a different set of effort multipliers
- E is applied to number of changed KSLOC (added and modified, not deleted)
- Maintenance effort
  – $PM_M = A \times (Size_M)^E \times \prod_{i=1}^{15} EM_i$

AGILE ESTIMATION - SIZE

Estimating Size with Story Points

- Story points
  - "Are a unit of measure for expressing the overall size of a user story, feature, or other piece of work." [3]
  - Are relative
    - A story that is assigned a two should be twice as much as a story that is assigned a one.
  - Estimated by entire team
- Velocity
  - A measure of a team's rate or progress for iteration-based agile approaches
  - Sum of completed story points by the team in an iteration
    - A team planned 4 user stories with story points as 8, 3, 3, and 5 for an iteration, and completed the first three at the end of the iteration. The team's velocity for that iteration is 14.
- Project duration
  - Total user story points / average (or estimated) velocity
  - Example
    - Project total user story points = 150, average velocity = 15, one iteration = 2 weeks
    - Duration is estimated to be 20 weeks
- Related agile practices
  - Relative Estimation
    - Avoids some common pitfalls, including seeking unwarranted precision, confusing estimates for commitments

Estimating Size in Ideal Days/Hours

- How long is a game of American football?
  - Ideal time
    - Amount of time that something takes when stripped of all peripheral activities
    - 4 x 15 = 60 minutes
  - Elapsed time
    - The amount of time that passes on a clock
    - About 3 hours
- It is almost always easier and more accurate to predict the development time for a user story in ideal time than in elapsed time
- Why ideal time differs from elapsed time for software development?
  - Supporting the current release, sick time, meetings phone calls, special projects, training, email, reviews and walk-throughs, interruptions, multitasking, …
- Ideal days (hours) as a measure of size
  - The number of days (hours) of effort that it would take the team to get a story done without interruptions.
  - Assumption
    - No interruptions, no external dependencies, spend 100% time on the user story

Techniques for Estimating

- Be aware of diminishing return on time spent estimating
  - Expending more time and effort to arrive at an estimate does not necessarily increase the accuracy of the estimate.
  - No amount of additional effort will make an estimate perfect.
  - Vary the estimating effort according to purpose of the estimate
    - Is the estimate going to be used to make buy versus build decision?
    - Is the estimate going to be used to select user stories for the current sprint?
- Estimating should be a team activity
  - Do not rely on a single expert to estimate
  - It is important to have input from everyone in the team, because anyone may work on anything
- Estimate should be on a predefined scale
  - People are best at estimating things that fall within one order of magnitude
  - Nonlinear sequences work well, such as 1, 2, 3, 5, 8, 13, 20, 40, 100
  - Including 0 is often useful
  - User stories that will be worked on in the next few iterations need to be small enough to be completed in a single iteration
  - User stories or other items that are likely to be more distant than a few iterations can be left as epics or themes

Techniques for Estimating – Cont.

- [Planning poker]
  - It is best to include entire team (e.g., scrum team), product owner participates but does not estimate
  - Equipment
    - A deck of (physical or virtual) cards with numbers for each player.
  - Procedure
    - A short overview of a user story is provided by the product owner. The team is given an opportunity to ask questions and discuss to clarify assumptions and risks.
    - After the discussion, everyone lays a card face down representing his/her estimate.
    - The moderator asks everyone to turn his/her card face up at the same time.
    - People with the highest and lowest estimates offer their justification for their estimate and discussion continues.
    - Repeat the process until a consensus is reached.

Should You Re-estimate?

- Case 1
  - 4 user stories, each was estimated with 3 story points.
  - The team estimated to finish 12 story points per iteration, but only finished 6 story points after the first iteration. They want to re-estimate the finished user stories as 6 points each.
- Case 2
  - 6 user stories were estimated with story points, 3, 5, 3, 3, 3, 5, respectively
  - The first 3 user stories each has to do with displaying chart
  - The team picked the 1st, 2nd, and 6th user story for a sprint, but at the end only finished 1st and 6th user story
  - After finishing the 1st user story, the team feel it was twice as hard as previously estimated, due to underestimated difficulty of displaying chart, they want to re-estimate the 2nd and 3rd user stories
- Case 3
  - 6 user stories were estimated with story points, 3, 5, 13, 3, 3, 5, respectively
  - The team picked the first three for a sprint, finished first two user stories and partially completed the third user story, the team want to re-estimate the third user story.

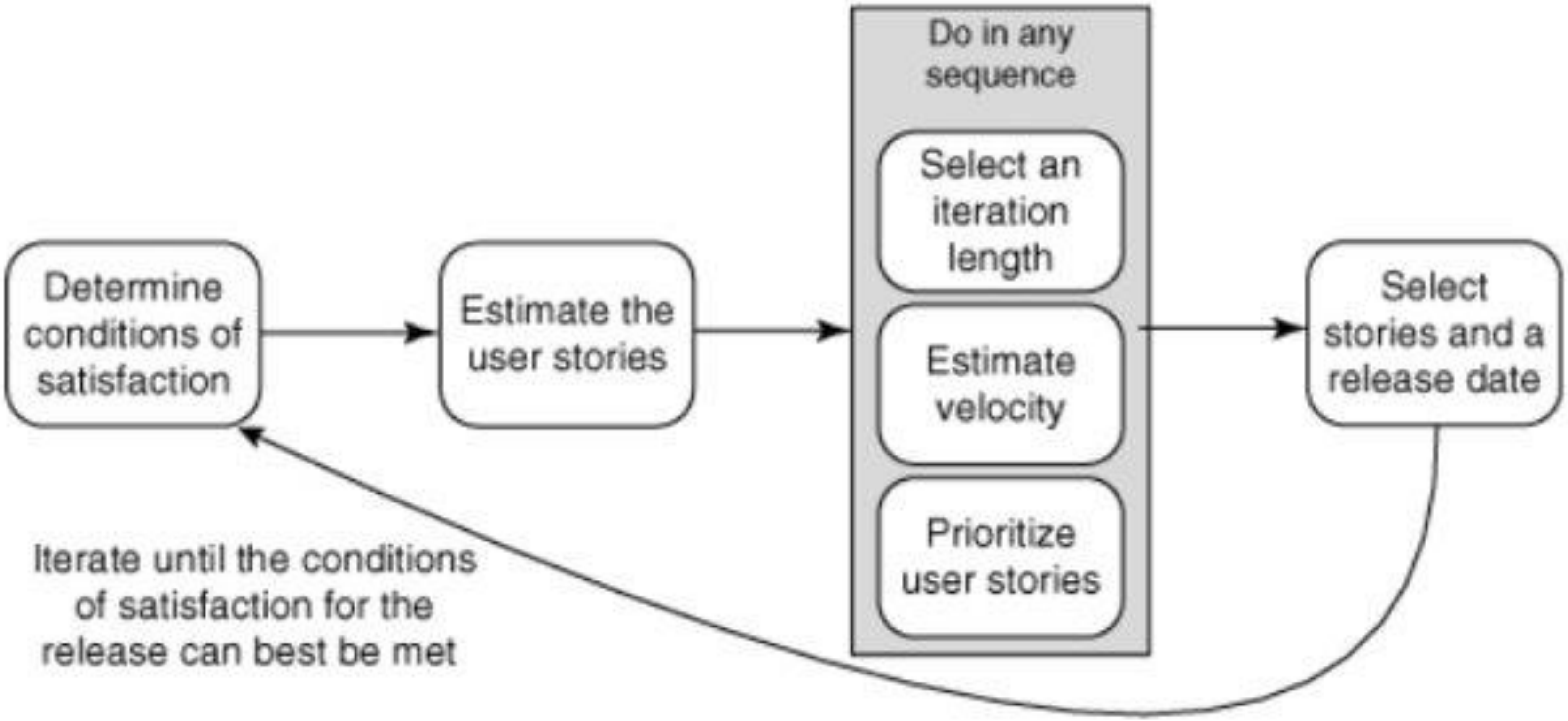
AGILE ESTIMATION - SCHEDULE

Release Planning Essentials

- Release plan
  - A typical release covers three to six months
  - Can include additional information such as start and end date, people on the team, iteration length etc.
- Steps in planning a release
  - Determine conditions of satisfaction
    - Are normally defined by a combination of schedule, scope, and resource goals
  - Estimate relevant user stories
  - Select an iteration length
    - Typically, 2-4 weeks
  - Estimate velocity
    - Use past data if they are available
  - Prioritize user stories
  - Select stories and a release date
    - feature-driven project: #iterations = story points of required features / expected velocity
    - date-driven project: #story points = #iterations x expected velocity
- The release plan should be revisited and updated with some regular frequency

Question: Is your current project date-driven or feature driven?

Steps in Planning a Release



[2]

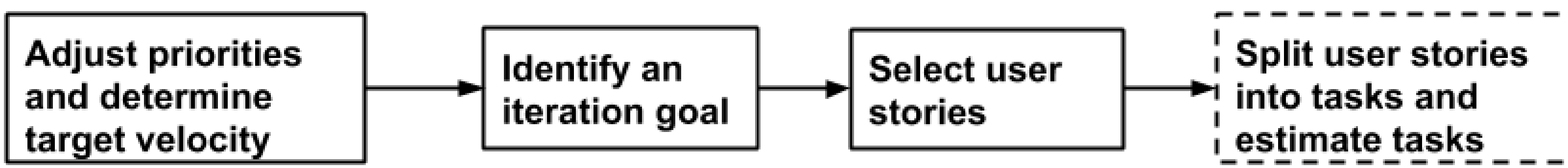
Release Planning Example

- A start-up must release a new product in three months
- Determine the conditions of satisfaction
  - Date-driven project, fixed budget, included features are flexible
- Estimate user stories
  - Relevant user story estimates are shown in the table on the right
- Select iteration length
  - 2 weeks
- Estimate velocity
  - 8 story points
- Prioritize user stories
  - The prioritized user stories are shown in the table on the right
- Select user stories
  - 3 months has 6 iterations
  - Total story points: 6x8=48
  - Select top 8 user stories with 46 story points

Story ID	Estimate
US01	5
US02	5
US03	8
US04	5
US05	5
US06	5
US07	5
US08	8
US09	8
US10	3
US11	8
US12	3

Velocity-Driven Iteration Planning

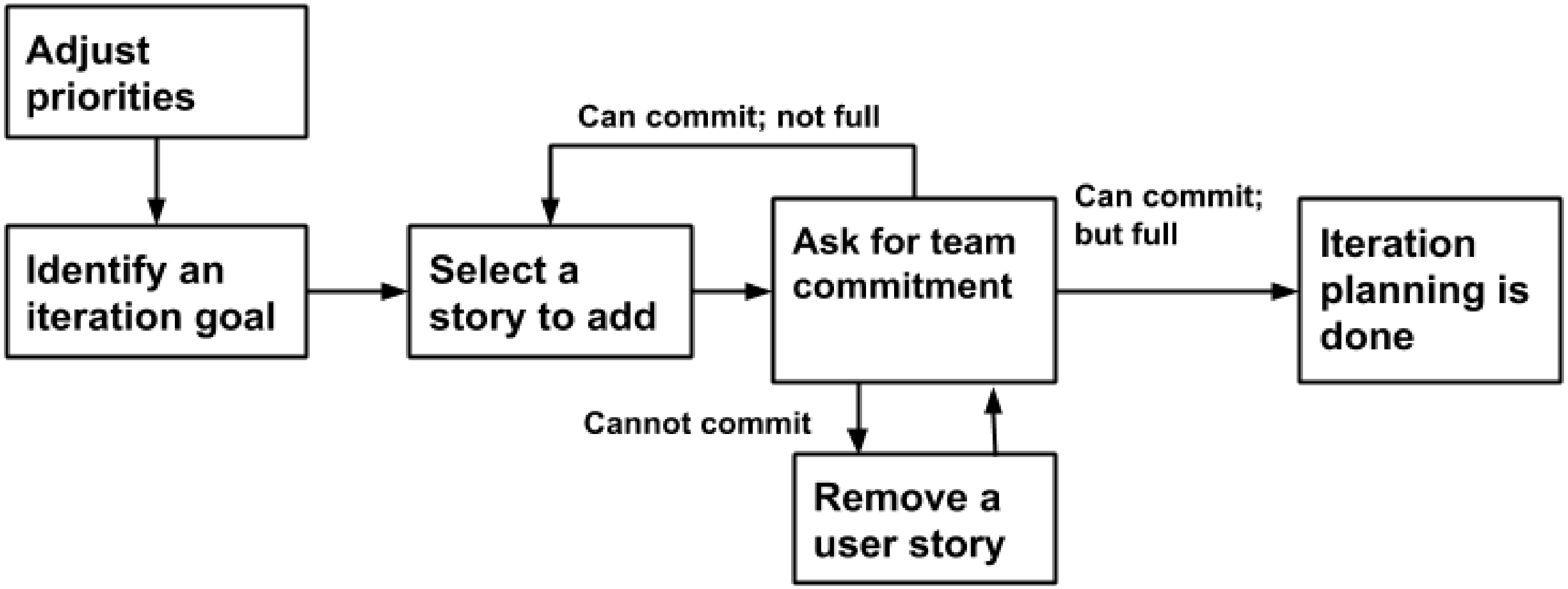
- Priorities for user stories can change over time
- Velocity of recent iteration, the moving average velocity, or the forecasted velocity can be used for target velocity
- User stories are picked using team velocity
- User story estimates should include time for fixing bugs found in the current iteration
- Optionally, a user story can be broken into tasks, task estimates are often expressed in ideal time



[2]

Commitment-Driven Iteration Planning

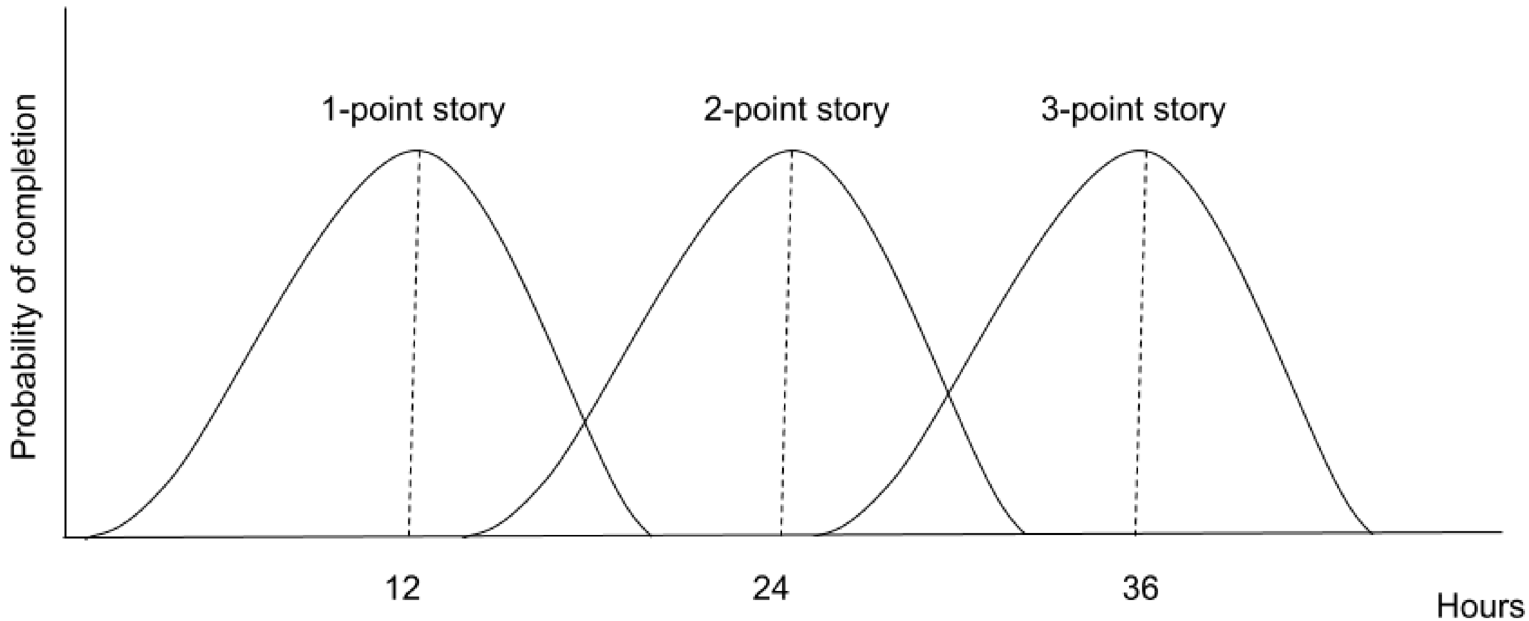
- The team is asked to add stories to the iteration one by one until they can commit to completing no more.



[2]

Question: What kind of iteration planning have you had?

Task Estimates & Story Points



The relationship between ideal hours and story point is a probability distribution

Selecting an Iteration Length

- Most teams use iteration lengths of 2-4 weeks
- The length of iterations determines
  - How often the software can be shown in potentially shippable form to users and customers.
  - How often the progress can be measured.
  - How often the team can refine their course due to changes.
- Factors in selecting an iteration length
  - The overall length of the release
    - One-month iteration only gives 2 review and feedback opportunities for a projects needs to be release in 3 months
  - The amount of uncertainty
    - The more uncertainty there is, the shorter the iterations should be
  - How long priorities can remain unchanged
    - It is better to keep priorities unchanged during an iteration
    - The time from new idea to working product will take on average 1.5 x iteration length
  - The easy of getting feedback
    - Iteration length should be chosen to maximize the amount, frequency, and timeliness of feedback
  - Willingness to go without outside feedback
    - The less often a team receives outside feedback, the more likely it goes astray
  - The overhead of iteration
    - Planning, regression testing, ...
  - How soon a feeling of urgency is established
    - If an iteration is too long, there is a tendency to relax a bit at the start of the iteration, which leads to panic and longe r hours at the end of the iteration.
    - Select an iteration length that evens out the pressure the team feels

Iteration Length Case Studies

The Napa Project

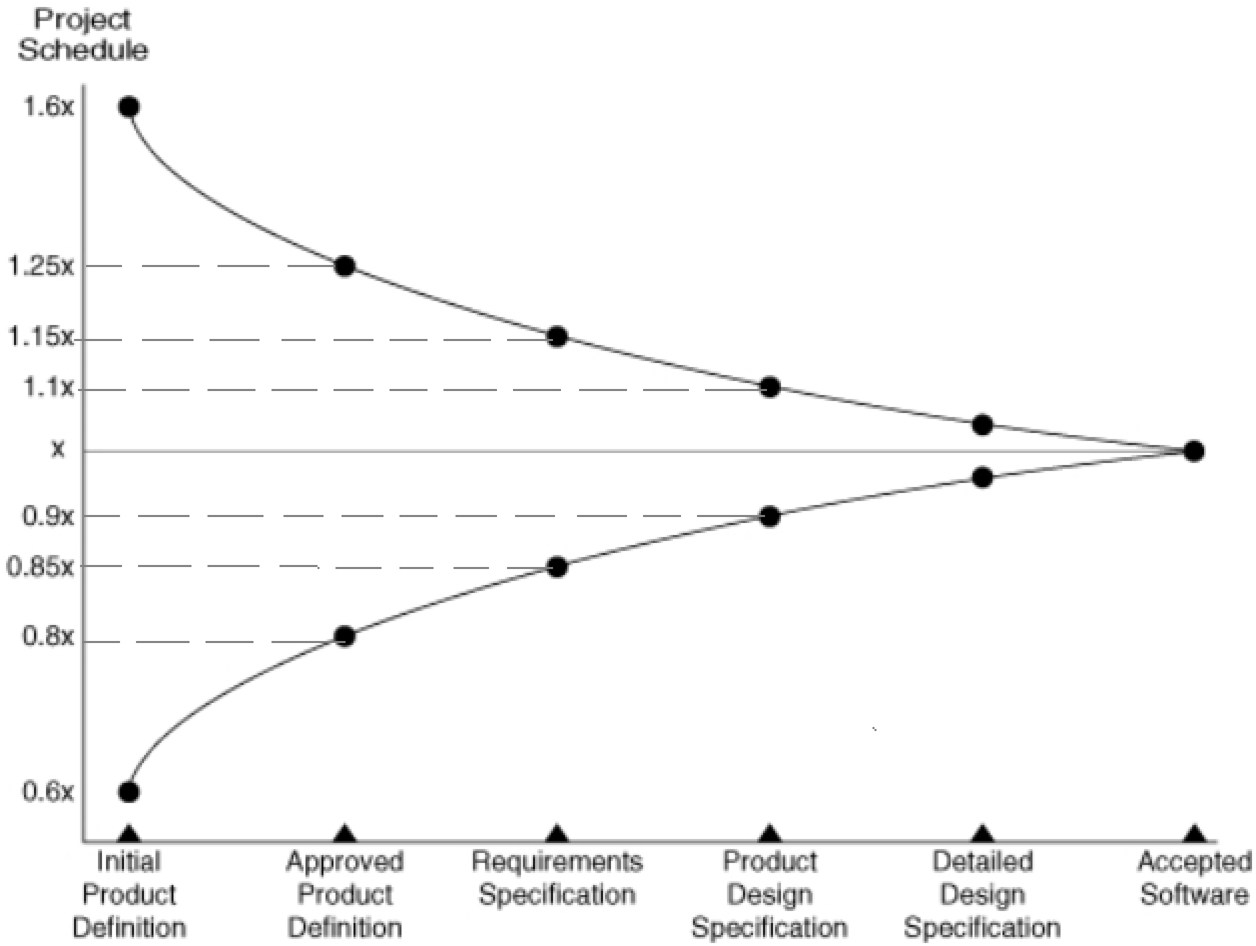
- 7-person team
- Client-server desktop applications, internal project
- Estimated to take 13 months
- Access to users are partially restricted
- Critical project with high visibility
- 4-week iteration was chosen

The Goodman Project

- Two 9-person teams
- Enterprise application, commercial project
- Estimated to take a year
- Preliminary release was planned after 6 months
- There was tremendous amount of uncertainty
- 2-week iteration was chosen

Question: What is the appropriate iteration length for your current project?

Cone of Uncertainty [2]



Estimating Velocity - Run an Iteration

- The best way to predict velocity is to observe velocity
- If one iteration is run and its velocity is 15, use cone of uncertainty to provide estimate
  - Estimated velocity = [15x0.6 .. 15x1.6] = [9 .. 14]
- If three iterations are run and velocities are 12, 15, 16
  - Use range of observed values to provide estimate
    - Estimated velocity = [12 .. 16]
  - Use cone of uncertainty to provide estimate
    - Average velocity = (12 + 15 + 16)/3 = 14.3
    - Estimated velocity = [average velocity* 0.85 .. average velocity*1.15] = [12.2..16.5]

Iteration Completed	Low Multiplier	High Multiplier
1	0.60	1.6
2	0.80	1.25
3	0.85	1.15
4 or more	0.90	1.10

Estimating Velocity - Use Historical Values

- They are of greatest value when very little has changed between the past and current project and team
- Check list
  - Is the technology the same?
  - Is the domain the same?
  - Is the team the same?
    Is the product owner the same?
  - Are the tools the same?
  - Is the working environment the same?
  - Were the estimates made by the same people?
- If answer to each of the above question is yes, use historical values and express estimated velocity as a range
- Otherwise, use cone of uncertainty to provide estimate
  - [average velocity* 0.6 .. average velocity*1.6]
  - Example
    - A team finished 150 story points in 10 iterations for a past project
    - Average velocity = 150/10 = 15
    - Estimated velocity for the new project = [15x0.6 .. 15x1.6] = [9 .. 14]

Estimating Velocity - Make a Forecast

- When there are no historical data, and it is infeasible to run a few iterations to observe velocity
  - e.g., Start a new project only after the clients signs a contract
- Steps
  - Estimate the number of hours that each person will be available to work on the project each day.
    - Studies found most individuals spent between 55% to 80% of their time on project activities
    - e.g., 6 hours per day
  - Determine the total number of hours that the team will be spent on the project during the iteration.
    - e.g., 4-person team, each person 6 hours per day, 2-week iteration: 4 x 6 x 10 = 240
  - Randomly select stories and expand them into tasks. Repeat until enough tasks have been identified to fill the number of hours in the iteration.
    - e.g., 221 task hours for 9 user stories, with total 25 story points
  - Convert the velocity determined in the preceding step into a range.
    - e.g. [25x0.6 .. 25x1.6]

Questions: How many hours per iteration does your team spend on the project?

Buffering Plans for Uncertainty

- Can be used when there is significant uncertainty or the cost of being wrong is significant
- Feature buffers
  - Project includes mandatory and optional features
  - Optional features will be included only if time permits
  - DSDM (Dynamic Systems Development Method)
    - Requirements are categories into: Must Have, Should Have, Could Have, and Won't Have.
    - No more than 70% of the planned effort for a project can be targeted at Must Have requirements.

Buffering Plans for Uncertainty – Cont.

- Schedule buffers
  - Add the schedule buffer to 50% estimate
  - avgCase: user story estimate with 50% confidence level
  - worstCase: user story estimate with 90% confidence level
  - Approach 1
    - Provide avgCase and worstCase estimates for each user story
    - bufferSize = $\sqrt{\sum(worstCase_i - avgCasei)^2}$
  - Approach 2
    - Limitation
      - not influenced by the actual uncertainty around specific user stories
    - bufferSize = $\frac{1}{2} * \sum avgCase_i$
  - Guidelines
    - Approach 1 is most reliable if there are at least 10 user stories
    - Schedule buffer should represent at least 20% of the total project duration
- Combing buffers
  - E.g. use both feature & schedule buffers
  - A project might use other buffers, such as budge buffer
- Some caveats
  - If precise deadline with a precise set of delivered functionality is not needed, do not take on extra work of adding buffers
  - If buffers are used, do not hide their existence or how they are used

Buffering Plans for Uncertainty - Schedule Buffer Example

- Assume average velocity is 9 story points per iteration
- Approach 1
  - bufferSize
    - $\sqrt{90} \approx 9$
  - projectTotal
    - 17 + 9 = 26
  - numIterations
    - 26/9 ≈ 3
- Approach 2
  - bufferSize
    - 17/2 ≈ 8
  - projectTotal
    - 17 + 8 = 25
  - numIterations
    - 25/9 ≈ 3

User Story	Average Case (50%)	Worst Case (90%)	$(\text{Worst–Average})^2$
As any site visitor, I would like to see the personal records of any swimmer.	5	13	64
As any site visitor, I need to be authenticated before given access to sensitive parts of the site.	1	3	4
As a swimmer, I want to see when practices are scheduled.	3	5	4
As a swimmer or parent, I want to know where league pools are located.	5	8	9
As any site visitor, I want to see the national records by age group and event.	2	5	9
As any site visitor, I would like to see the results of any meet.	1	1	0
Total	**17**	**35**	**90**

Estimate with Confidence Level

Feature Set	# Stories	Story Points	Total Story Points	Confidence Level
1	16	2 or 3 for each story	40	High
2	8	1 for each story	8	High
3	12	2 or 3 for each story	30	High
4	15	5 or 8 for each story	120	Medium
5	12	13 for each story	156	Low
Totals				
5 feature sets	63 stories	Unclear	354	Medium

If the team has a day for one-point story, it will take the team about 70 weeks to finish the project with about 50% confidence.

Alternatives

Story	Story Start Day	Story End Day	Story Duration
1	Day 1	Day 3	2 days
2	Day 3	Day 4	1 day
3	Day 4	Day 7	3 days
4	Day 7	Day 9	2 days
5	Day 8	Day 10	2 days
Average Cycle Time = Total Duration / total number of stories = 10 / 5 = 2			

- Count user stories for iteration/release planning
  - When all the stories are roughly the same size
- Use cycle time and story count for iteration/release planning
  - The average cycle time helps people know the average duration of a story.
  - If a project has 20 user stories, with average cycle time 2 days per user story, it takes about 40 days (8 weeks) to finish the project.
- #NoEstimates

References

- [1] Center for Software Engineering, USC, COCOMO II Model Definition Manual, Version 2.1
- [2] Function Point Counting Practices: Manual Release 4.0, International Function Point Users' Group, Blendonview Office Park, 5008-28 Pine Creek Drive, Westerville, OH 43081-4899.
- [3] Agile Estimating and Planning 1st Edition; Author: Mike Cohn; ISBN-13: 978-0131479418; ISBN-10: 9780131479418
- [4] Create Your Successful Agile Project, Johanna Rothman, Pragmatic Programmers LLC, 2017. ISBN:9781680502602