

# Project Estimation and Risk

- practical software estimation
  - general principles of estimation
  - estimating size, complexity, test cases, bugs
  - estimating productivity
  - reasonable estimates and how to give them
- project risk
  - assessment
  - mitigation, monitoring, and management
  - common risk and management approaches

# Project Estimation

*“Predictions are tough,  
particularly when they involve the future.”*  
Yogi Berra

- project estimation involves imponderables
  - we aren’t exactly sure what we will do
  - we don’t know what problems will arise
  - we don’t know what distractions we will have
- inaccurate estimates can be disastrous
  - projects fail because of bad estimates

## estimates vs. schedules

- estimates
  - time and resources required for each task
  - usually prepared by engineering
- schedules
  - which tasks will be performed when
  - which resources will be used when
  - when will each task be completed
  - usually prepared by management
- schedules are based on estimates
  - but incorporate much additional information

## Estimation Principles

- estimates are not guesses
  - good estimate = good data + good analysis
- estimates are not precise or deterministic
  - it is not a number, but a confidence range
  - estimates start out very “rough”
  - they are revised throughout life of project
- get estimates from multiple sources
  - ask different people to make the estimates
  - use multiple techniques to develop estimates

## Estimation Principles - detail

- estimate at a low level of detail
  - for each component, or sub-component
  - for each activity, step, task, and sub-task
- low level estimates invite you to consider
  - the full range requirements to be satisfied
  - the design of the components to be built
  - the methodology to be used to build them
  - the kinds of problems that are likely to occur
- planning and estimating go hand-in-hand

## Development Estimate Factors

- Program size
  - LOC, function points, routines, classes, ...
- Program complexity
  - algorithmic, data-structures, interactions, ...
- Programmer productivity
  - (ready-to-integrate) LOC / day
- Required test cases
  - test cases, LOC, time to get them working
- Bugs we will have to fix at each stage
  - with associated productivity rates

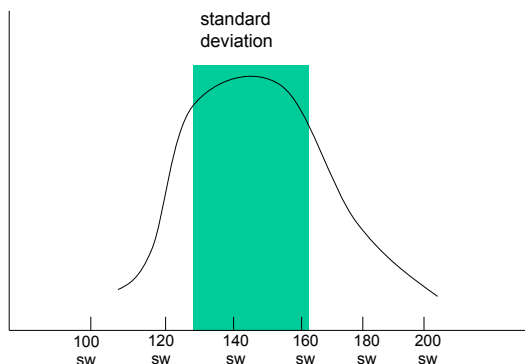
## bugs, tests and project size

- simple logic & coding errors tend to be flat
  - for a particular person, problem, technology
- other errors scale with the problem
  - misunderstood specifications
  - misunderstood interfaces
  - unanticipated interactions
- test cases should scale with the risk
  - it will take more tests to find these problems
- interaction problems take longer to debug

## Giving Estimates

- you are not allowed to say “I can’t”
  - it is part of deciding if a project is viable
- don’t just make up a number
  - it will be both wrong and indefensible
- **do what you’ve been trained to do:**
  - gather data, make assumptions, do analysis
    - put assumptions and analysis out for review
  - present results honestly (as confidence band)
    - be able to provide a basis for every number
    - have a plan for narrowing confidence bands

## a reasonable estimate



## a back-up slide

Requirements	Architecture	Integration	System test
8-12sw	12-15sw	8-12sw	12-20sw

Design	Coding				Testing			Debugging			total
	sw	LoC	rate /sw	cost (sw)	#	rate /sw	cost (sw)	#	rate /sw	cost (sw)	sw
U/I front end	3	1500	150	8	150	20	6	25	8	3	20
	4	2000	180	13	200	25	10	35	10	5	32
Task Applets	2	1000	100	6	150	40	3	15	5	3	14
	3	1200	150	12	150	50	6	25	6	5	26
Access Clients	2	500	120	3	75	40	2	5	4	1	6
	3	700	150	6	140	50	4	10	5	2	15
Storage Server	4	900	100	8	90	25	3	25	3	5	20
	5	1200	120	12	150	30	6	40	5	13	36
Content	1	6000	600	7	N/A	1200	5	100	40	2	15
	2	8000	900	13			6	120	50	3	24

Grand Total: 115-192 sw = 25-43sm

## Agile and Estimation

- agile methods are not date driven
  - they strive for customer satisfaction
  - flexible requirements demand flexible dates
- agile methods don’t believe in big plans
  - requirements and plans will change
  - refactor as need for change is recognized
- design and estimation are progressive
  - progressive requirement/task refinement
  - design the next few tasks on the list
  - estimates are based on detailed designs

## Estimation: Agile vs. Planned

- they completely agree on basic principles
  - based on a detailed design
  - seek consensus from multiple sources
  - estimates are imprecise and inaccurate
  - be honest, rational, and clear
- they differ on prognostication
  - “Always in motion the future is”
  - “Do the best you can w/available information”
- these represent different project models

## Different Models and Goals

- Agile ... planning the next sprint
  - the team and sprint cadence are givens
  - how much can we accomplish in this sprint?
- Big Waterfall ... product planning
  - how many people will we need?
  - when will we be ready to ship?
- Does project demand full up-front plan?
  - even so, agile processes may still apply
    - continuous integration, regular customer feedback, focused small team sprints

## On Risk

“If you’re not managing risk, you’re managing the wrong thing!”

Rear Admiral Bill Carlson

“Risk Management is Project Management for adults.”

Tim Lister

## Risks: poor technical planning ●

- incorrect or incomplete requirements
- designs that can’t be built or won’t work
- schedule based on inadequate analysis
- team lacks experience w/tools, techniques
- problems that prove harder than expected ●

## Risks: poor management

- schedules imposed without commitment ●
- external dependencies with no back-ups ●
- failure to assign required resources
- doing unnecessary/less important work
- failure to monitor progress and problems
- delayed or ineffective problem response
- poor inter/intra-group communication

## Risks: unexpected events

- requirements changes
  - customer changes his/her mind
  - evolving competitive landscape
- staffing problems
  - illness, resignations
  - hiring difficulties
- distractions
  - customer emergencies
  - legacy support work-load

## risk assessment

- like software failure mode enumeration
  - enumerate all plausible sources of risk
    - unclear/unstable requirements
    - poorly understood technical problems
    - staff size, skills, experience, tools
    - complexities of the domain and platform
  - describe each in as much detail as possible
  - rate each for likelihood and impact
  - order them by risk exposure (likelihood \* impact)
  - decide which warrant inclusion in the plan

## risk management

- for each high exposure risk, formulate
  - proactive mitigation measures
    - what can we do to reduce its likelihood
    - what can we do to reduce its expected impact
  - reactive monitoring and management plan
    - what danger signs should we watch for
    - how will we respond when problem happens
- cost-benefit comparison of alternatives
  - determine the most cost-effective approach
- incorporate into plans and schedules

## Wrap-up on Risk

- this entire course has been about risk
  - problems that cause projects to fail
    - their nature and causes
    - the elements of good (and bad) solutions
  - processes to mitigate these risks
    - to prevent them
    - to monitor and manage them
- this is only an introduction/overview
  - **you** must gain facility with the concepts
  - **you** must develop skill in the techniques
  - **you** must develop disciplined approaches

## For Next Lecture

- McConnell section 34.7 (risk awareness)
- Wiegers: Successful Project Management
- Brooks: Mythical Man Month (digest)
- Kampe:
  - Project Management Concepts
  - Project Milestones
  - Putnam Norden Rayleigh curves
- Wiki: Earned Value Analysis
- Kampe: SCRUM points and velocity

## Supplementary Slides

## Estimation Principles - other stuff

- be sure to include other process activities
  - recruiting and training new personnel
  - participation in reviews
  - training for customers, sales and support
  - customer driven travel and meetings
  - hand-holding alpha customers
- be sure to include waiting time
  - for hires, feedback, turn-around time

## other project risks

- business risk
  - our business strategy will change (*executive*)
  - the market will change (*inbound marketing*)
  - we won't be able to sell it (*sales*)
- project risk
  - failure to fund or staff (*executive*)
  - external dependencies (*program management*)
  - failure to monitor and manage (*project management*)
- these risks are assumed by other people
  - this is why there are project approval processes