projects, teams and estimation

- 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

Elements of an Estimate

- 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

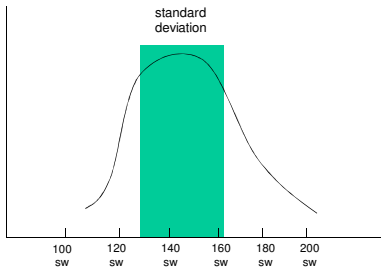
bugs, tests and project size

- 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 real engineers 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 4	1500 2000	150 180	8 13	150 200	20 25	6 10	25 35	8 10	3 5	20 32
Task Applets	2 3	1000 1200	100 150	6 12	150 240	40 50	3 6	15 25	5 6	3 5	14 26
Access Clients	2 3	500 700	120 150	3 6	75 140	40 50	2 4	5 10	4 5	1 2	6 15
Storage Server	4 5	900 1200	100 120	8 12	90 150	25 30	3 6	25 40	3 5	5 13	20 36
Content	1 2	6000 8000	600 900	7 13	N/A	1200	5 6	100 120	40 50	2 3	15 24

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

Risk Managment

"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

technical project risk

- Planning failures
  - incorrect or incomplete requirements
  - schedule based on inadequate analysis
  - schedules imposed without commitment
  - external dependencies with no back-ups
- Management failures
  - doing unnecessary work
  - assigning the wrong resources to a task
  - failure to monitor and respond to problems
  - poor inter/intra-group communication

technical project risk

- Changes to the problem
  - requirements changes
  - resource or schedule changes
- Unanticipated technical difficulties
  - team lacks training and experience
  - issues with new tools & techniques
  - designs that can't be built or won't work
  - problems that prove harder than expected
  - unexpectedly low productivity

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

Course Wrap-up

- 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