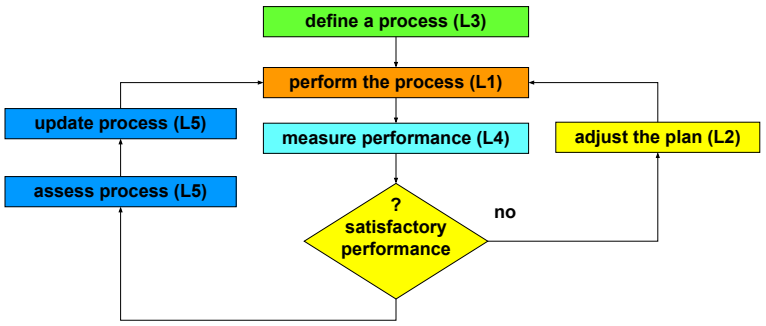
Basic Project Skills

- Project Post-Mortems
  - motivations and their use in this course
  - examine the Diablo II post mortem
- Project Planning
  - work break-down, dependency analysis
  - risk assessment and mitigation
  - estimation and scheduling
- Version Control
  - motivation and its use in this course
  - git and GitHub

Process Improvement 1A
(the Capability Maturity Model)



Post Mortems/Retrospectives

- complex skills must be developed/refined
- every project is a learning opportunity
  - improve our skills with existing methodology
  - try new techniques, confront new problems
  - learn from our mistakes
- post-mortems are pro-active learning
  - reflect and discuss as a group
  - what worked and what didn't work?
  - what should we do differently next time?

Post Mortems/Retrospectives

- there are many techniques and formats
- all have the same basic requirements:
  - honesty: willing to acknowledge mistakes
  - introspection: willingness to analyze them
  - safety: no penalties for admitting mistakes
- in this course
  - you will develop post mortem processes
  - they will help you learn from the projects
  - you will learn to use them as a learning tool

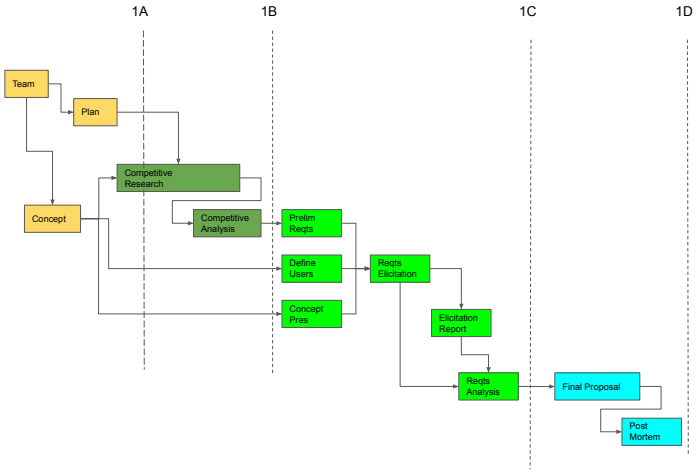
Diablo II Post Mortem

- Gamasutra article by the D2 design lead
  - brief history leading up to this project
    born from D1 wish-lists and complaints
  - organization, methodology, tools
  - discussion of things that worked
    D2 is Diablo, hiring/development process, new skill trees, massive Q/A effort, world-wide release
  - discussion of things that didn't work
    the old battle.net, # of users, non-state-of-the-art graphics, weak tools, a new game-save feature

Common Themes (prior PMs)

- problems
  - communication and pace management
  - insufficient tool/API/service mine-sweeping
  - insufficient planning for impl/integration
- successes
  - shared board/screens for brainstorming
  - timezone/schedule aware mtgs and pairing
  - clarity gained from competitive research
  - social networks for requirements gathering
  - team reviews of all deliverables
  - reviewing other projects improved their own
  - benefits of reviews, pair-programming, TDD

Project 1: Concept and Proposal



Teams and Planning

- Share your thoughts on
  - general product concept
  - relevant experiences and strengths
  - implementation tools & technologies
- Consider
  - how well it will meet P2-4 requirements
  - team skill composition (for the work)
  - is this the right team for you
- wrap-up at 10:35

Plan Check List

- task descriptions:
  - everyone believes the list to be complete
  - all task owners understand:
    - what their tasks mean, and how to do them
    - what they will deliver, when, in what form
- risk exposures and mitigation:
  - everyone agrees w/assessments & plans
- schedule:
  - achievable w/adequate room for problems

Regular Team Status Meetings

- a good meeting
  - has clear goals ... and accomplishes them
  - is brief … avoids tangents and rat-holes
- Daily SCRUM or XP *Stand-Up*
  - what have you completed since last meeting?
  - what do you plan to do by the next meeting?
  - what is getting in your way?
    - if there is a problem, assign people to work it

version control: GitHub

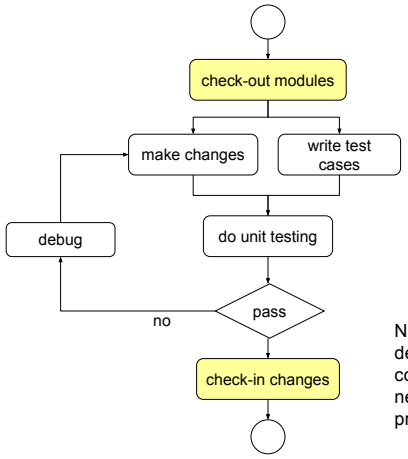
- version control is critical to any s/w project
  - track changes, previous versions, backup
  - work products come from version control
- centralized control is the old paradigm
  - new projects are distributed collaborations
  - distributed version control is more powerful
  - "git" is today's dominant open-source tool
- all code work products should be on GitHub
  - suggest you also use issues and project board

For next lecture

- McConnell 3-3.2
  - the importance of having a plan
- McConnell 34.2
  - process models
- Kampe: S/W Process Models
  - introduction to project phases and models
- Boehm: Spiral Development
  - iterative development: what, why, and how
- Ambler: "Big Requirements Up Front"
  - the agile critique of the classic Waterfall

Back up slides

software development process



The checked-out modules may include program source code, test cases, make files, configuration databases, etc.

Note that what development delivers is often not executable code. Rather they check in new versions of their work products.

the laws of version control

- All of our work products are versioned
  - we can tell what version we are dealing with
- All official changes are tracked
  - we know exactly what changes were made
  - we know who made each change, when, why
- We can reconstruct any version at any time
  - not just the current version, any prior version
- Files exist in multiple parallel branches
  - each of which has its most current version

version control procedures

- never deliver a work product directly
  - rather, deliver a version-controlled file
  - ensures proper recording of all work
- build from the version controlled files
  - extract specific (or current default) versions
- associate versions with deliverables
  - release has a list of all versions used to build it
  - test/bug reports associated w/specific releases
  - bug fixes are associated w/new file versions
  - work product approvals specify a version

change control

- who can change what, where, when?
- sometimes, some change is good
  - it represent progress as work is completed
  - such changes should be facilitated
- sometimes, some change is bad
  - changes can be disruptive to the product
  - we need processes to detect & prevent these
- hopefully these processes are adaptive
  - adjusting the burden in response to the risk

change control mechanisms

- may be performed by version control tools
  - may control who can modify which files
  - may notify interested parties of changes
  - these features are usually configurable
- may be managed by human processes
  - publication and objection
  - designated component reviewers
  - change control boards
- should have mechanism/policy separation

Conflicting Updates

- People may work at cross purposes
  - independent changes to same module
  - different understandings of how things work
- File Change Notifications
  - subscribe to notifications for selected files
- File Locking
  - at check out time, or independently
  - locks can be advisory or enforced
- Change merge assistance
  - automatic difference analysis, proposed merge

Deeper

- Why would there be so many different models for Post Mortems?
  - projects happen in different cultures
  - participants have different backgrounds
  - some situations are more emotionally charged
  - some situations demand greater formality

Deeper

- What do you think of having managers present at a post-mortem?
  - Why is it a good idea?
    they may be responsible for implementing changes
  - Is there another way to get that benefit?
    they could just read the summary report
  - Why is it a bad idea?
    their presence could stifle frank discussion
  - Is there any way to mitigate that problem?
    maybe with a strong culture and trusted managers

Deeper

- How do you know if you need to break a task/deliverable into sub-tasks?

  If you already know how to do it, it probably does not need to be further broken down.

  If you are not yet sure how to do it:

  perhaps it needs to be more completely described
  perhaps it needs to be broken into sub-sub-tasks
  perhaps you need to do some research first

Deeper

- Why do we need to specifically identify the whole team tasks?

  Meetings with other people are harder to schedule and need to be planned well in advance.

Deeper

- How can you know what “might” go wrong?
  - What parts of the problem or proposed solution do we not yet clearly understand?
  - What difficult and necessary skills do we not yet have?
  - What investigations/experiments might not yield the needed results?

Deeper B_4

- How do you decide whether to prevent, monitor and respond, or just deal with it when it happens?
  - ease of detecting and dealing with it later
  - cost of prevention v.s. cost of dealing with it

Deeper B_5

- Why should <u>potentially</u> serious problems be "provoked" as early as possible?
  - some problems may be so difficult that solving (or avoiding) them requires a different plan.
  - understand the problem and its solution while there is still time to deal with it.