Agile Process and XP

- the Waterfall model and formal process
  - goals, strengths, and weaknesses
- the Agile Process Rebellion
  - motivations and principles
  - eXtreme Programming
- Comparative Religion (agile vs. prescriptive)
  - similarities, differences, complementarities

The Wisdom of the Waterfall

- Basic Premise of the Waterfall Model
  1. understand the requirements
  2. develop a plan to satisfy them
  3. execute and manage against the plan
- Goals and Advantages
  - predictable functionality
  - predictable schedule and cost
  - minimal waste and minimal surprises

On Planning

The general who wins a battle makes many calculations in his temple ere the battle is fought. The general who loses a battle makes but few calculations beforehand.

Sun Tzu

No battle plan ever survived contact with the enemy.

Dwight Eisenhauer

There is no great art to devising a good plan of operations. The entire difficulty lies in this: To remain faithful in action to the principles we have laid down for ourselves.

Carl von Clausewitz

Fallacy of the Waterfall Model

- Assumptions
  - we are executing a well crafted plan
  - plan is designed to satisfy correct requirements
- Good requirements may be unattainable
  - in which case the plan is wrong
  - it will deliver the wrong functionality
- Even an incremental model may fail
  - because each incremental release is a waterfall

Formal Process - the good news

- it institutionalizes best practices
  - techniques to avoid common mistakes
- it enables more effective management
  - better planning and measurement
  - clear responsibilities and milestones
- it is necessary for process improvement
  - following a defined process
  - capturing permanent records of what we did
  - enables process evaluation and evolution

Formal Process - the bad news

- it *can* place form over substance
  - people are goaled on process deliverables
  - real goals are customer satisfaction, R.O.I
- one size may not really fit all
  - bureaucracy may greatly burden small projects
  - it makes assumptions that may not be true
- it is a lowest-common-denominator solution
  - it can improve the work of weak teams
  - it can greatly limit strong contributors

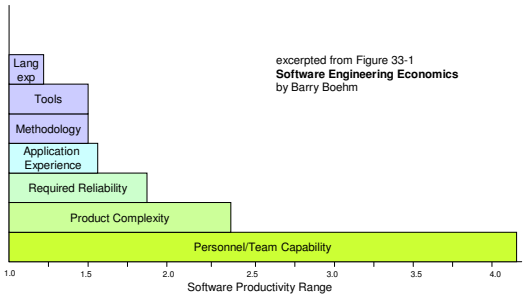
The Agile Process Rebellion

- Don't put too much faith in paper process!
  - our only deliverables are working software
  - our key goal is customer satisfaction
- Prescribed processes are, per force, flawed!
  - continuous change is a given – deal with it!
  - the best process is collaboration
    - all stake-holders communicate regularly
    - frequent, small, updates and good feedback
- Overemphasis on task definition is myopic!
  - people, not processes, solve problems

People v.s. Process

- Focus on creating good teams
  - ensure they have the right skill/perspective mix
  - ensure all the stake-holders are involved
  - ensure they are dedicated to the project
  - ensure they will work well **as a team**
- Then, let them do their job
  - don't tell them how to organize themselves
  - don't tell them how to best solve the problem
- Good teams beat good process, every time

The High Order Bits



excerpted from Figure 33-1
Software Engineering Economics
by Barry Boehm

Agile methods ...

- address people and teamwork issues
  - which are far too important to be ignored
- focus more directly on real goals
  - which is always a good thing
- put principles & methodology over process
  - which is almost surely right
- still enumerate required process activities
  - but avoid over-specifying tasks/deliverables
  - good for simple or poorly understood projects

eXtreme Programming

- an agile development philosophy
  - embracing change, short feedback-driven cycles, high stake-holder involvement, communication intensive
- a set of techno-moralistic principles
  - personal courage, mutual respect, responsible behavior, sustainable pace, minimalism, incrementality, continuous integration
- a collection of development practices
  - requirements gathering, design, coding, testing

eXtreme Programming Practices

- Planning Game
  - functional scenarios that fit on a 3x5 card
  - these are prioritized and mapped into tasks
- Pair Programming
  - two people working on a single task
  - one coding, one thinking, reviewing, planning
- Small, Frequent Releases
  - regular updates of incremental functionality
  - features as a measure of content & velocity

eXtreme Programming Practices

- Test Driven Development
  - implement features as they become testable
  - tests and features developed in parallel
- Continuous Integration
  - integrate code to product daily
  - get feedback and discover problems ASAP
- Design for Today and Refactoring
  - if you won't run the code today, don't write it
  - redesign code as problems are discovered

Comparative Religion - similarities

- They all follow the same basic process
  - understand the problem
    - start w/concept, gather & prioritize requirements
  - plan the solution
    - move from high level to more detailed design
    - prototype to reduce risk
  - execute the plan
    - implementing and testing proceed in tandem
    - monitor progress, look for problems, re-plan
- Agile approaches are intrinsically iterative

Comparative Religion - differences

	Planned approaches	**Agile approaches**
Stable Requirements	A *sine qua non* for a successful project	Somewhere between a myth and a *canard.*
Predictable Budget and/or Schedule	A primary goal	Get real!
User Satisfaction	Should be achieved if the requirements are right and we correctly implement them	The primary goal
Progress Measurement	Process related - project milestones	Customer related – delivered story cards

Complementary Religions

- There are different types of projects
  - with different goals and constraints
  - some require definition and control
  - others require investigation and iteration
- They are not mutually exclusive
  - agile processes can benefit from best practices
  - XP practices can be used in waterfall projects
  - traditional models ignored crucial team factors
  - getting it right from the start isn't always wrong

What We Take Away

- Agile Development is a "reformation"
  - pointing out the evils of over-process-ism
  - reminding us of our real goals
- Agile Development takes a broader view
  - addressing key principles and people issues
- Agile Development may be the other pole
  - small vs. large, U/I vs. infrastructure
- We don't have to choose between extremes
  - Hegel's Dialectic: thesis, antithesis, synthesis

For the next lecture

- SEI: Definition of architecture
  - the scope of architecture
- Garlan: Introduction to S/W Architecture
  - evolution of S/W complexity management
- McConnel 3.5-6, 5-5.3
  - the elements of good architecture
- introduction to some key architectural principles
  - Wikipedia: Mechanism/Policy Separation
  - Kampe: S/W Testability
  - Kampe: Interface Stability

Supplementary Slides

Q: How much process do I need?

A: Just enough to ensure your success

- How large and complex is the problem?
  - one person-hour, or 40,000 staff years
- How well understood is the problem?
  - do it every day, clear, imponderable
- How critical is quality?
  - throw-away, product, mission/life critical
- How critical are cost and schedule?
  - not a problem, +/-50%, deal-breakers

Q: How formal a process?

A: Enough to prevent misunderstandings?

- How many stake holders are there?
  - two, twenty, two-hundred, twenty thousand
- How good is their communication?
  - inseparable, can't be in the same room together
- Have they successfully done this before?
  - regularly, a few times, never
- How complex are the responsibilities?
  - obvious, I don't even know what they are

Kampe's Advice on Process

- there will never be a perfect process
  - life is change, every problem is different
  - one size will never fit all
- but specified process is still important
  - check lists and clear expectations are vital
  - too much is bad, too little can be even worse
- specify "what", not "how"
  - enumerate the things that need to be done
  - don't try to tell people how to do their jobs

Kampe's Advice on Process

- only mandate things that are vital
  - like approval check-lists and authority
  - standard forms are good if they make sense
  - most things can be recommended practices
- always do cost-benefit analysis
  - process is expensive, try to keep it lean
  - watch out for "social engineering"
- prototype your process changes
  - prove them before instituting them