

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

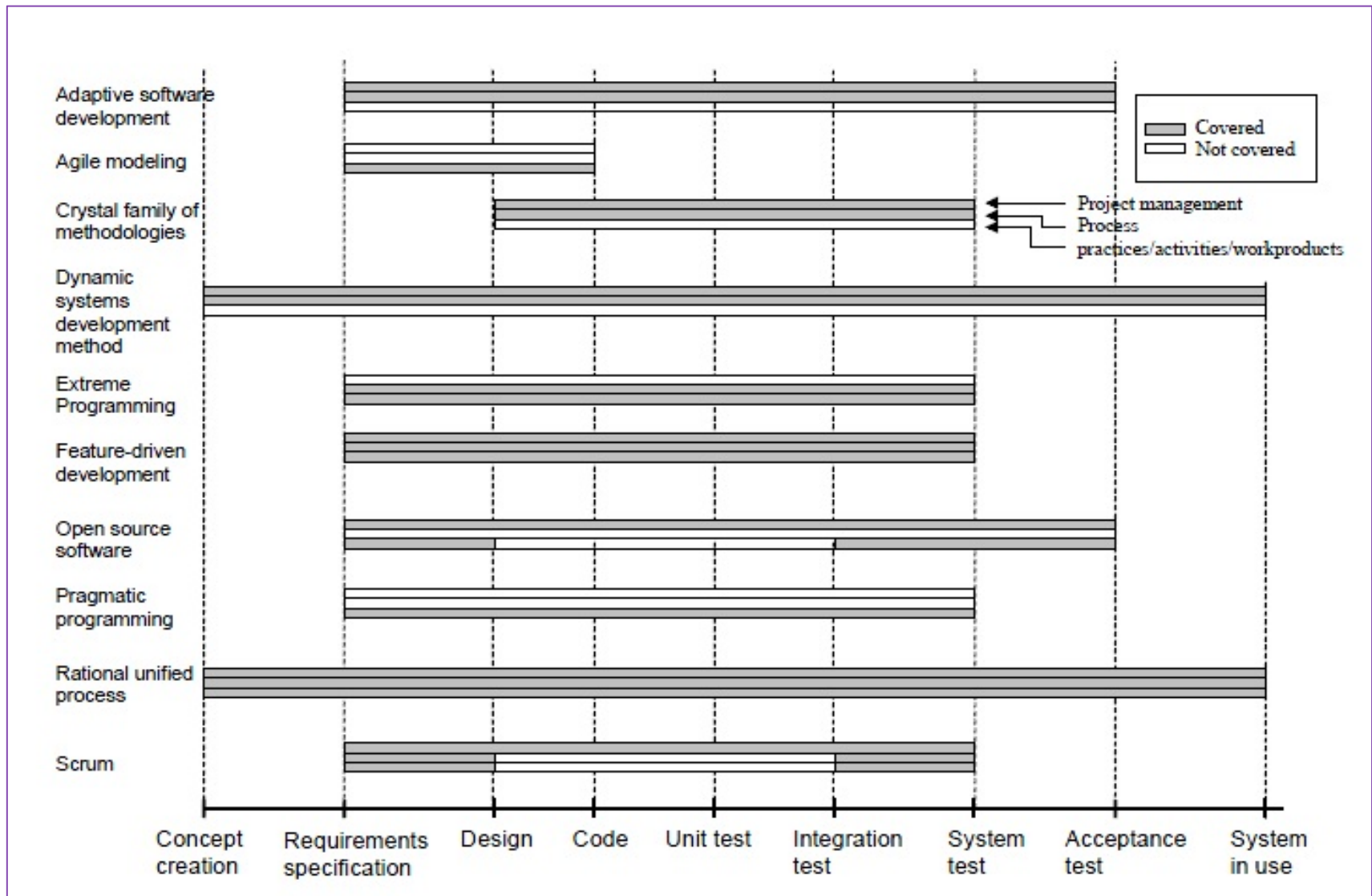
That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Genuine differences among agile approaches in scope and in content – but the commonalities are more important for now



A core issue – rarely explicit

- Developers write programs – static *descriptions*
- Customers care about the *behavior* of those static descriptions
- This is a huge gap, especially when we modify a program with the objective of achieving a new behavior
 - Might change the program to fix bugs
 - Might change the program to meet new or better understood customer needs
- Software is discrete – a small change in the program can cause a disproportional change in the behavior

Feedback loops in agile

Customer-team

- Frequently interactions and iterations between the customer and the team decrease the potential for big jumps in expected behavior
- Such jumps in behavior would, of course, often demand big changes in the program

Team-team

- Ensuring that changes to a program do what is intended is hard
- A tighter loop between the changes to the program and the changes to behavior decreases the potential for them to become significantly out of whack

Bounding divergence

- In other words, a lot of agile – and indeed a lot of software engineering techniques, methods, tools, and approaches in general – try to bound divergence
 - The divergence between the behaviors provided by the software and the behaviors desired by the customers [building the right system]
 - The divergence between the behaviors provided by the software and the behaviors the development team thinks (wants) the program will provide [building the system right]
- Another example from around the mid-1990's was Microsoft's nightly builds (or sync-and-stabilize) – at the very least, don't get too far from an executable program!

Concrete example of the “past”

- DOD-STD-2167A, MILITARY STANDARD: DEFENSE SYSTEM SOFTWARE DEVELOPMENT (29 FEB 1988) [S/S BY MIL-STD-498]., This standard establishes uniform requirements for software development that are applicable throughout the system life cycle. The requirements of this standard provide the basis for Government insight into a contractor's software development, testing, and evaluation efforts.
 - This was the basis for all US government procurement of software
 - Thousands of pages of specifications, documentation, diagrams, etc. – valuable in some cases, but not in all

<http://www.agile-process.org/>

- “We realize the way a team works together is far more important than any process. While a new process can easily improve team productivity by a fraction, enabling your team to work effectively as a cohesive unit can improve productivity by several times. ...
- “The most brilliant programmers alive working competitively in an ego-rich environment can’t get as much done as ordinary programmers working cooperatively as a self disciplined and self-organizing team. You need a process where team empowerment and collaboration thrive to reach your full potential.”

Note: agile and flexible does not equate to code-and-fix

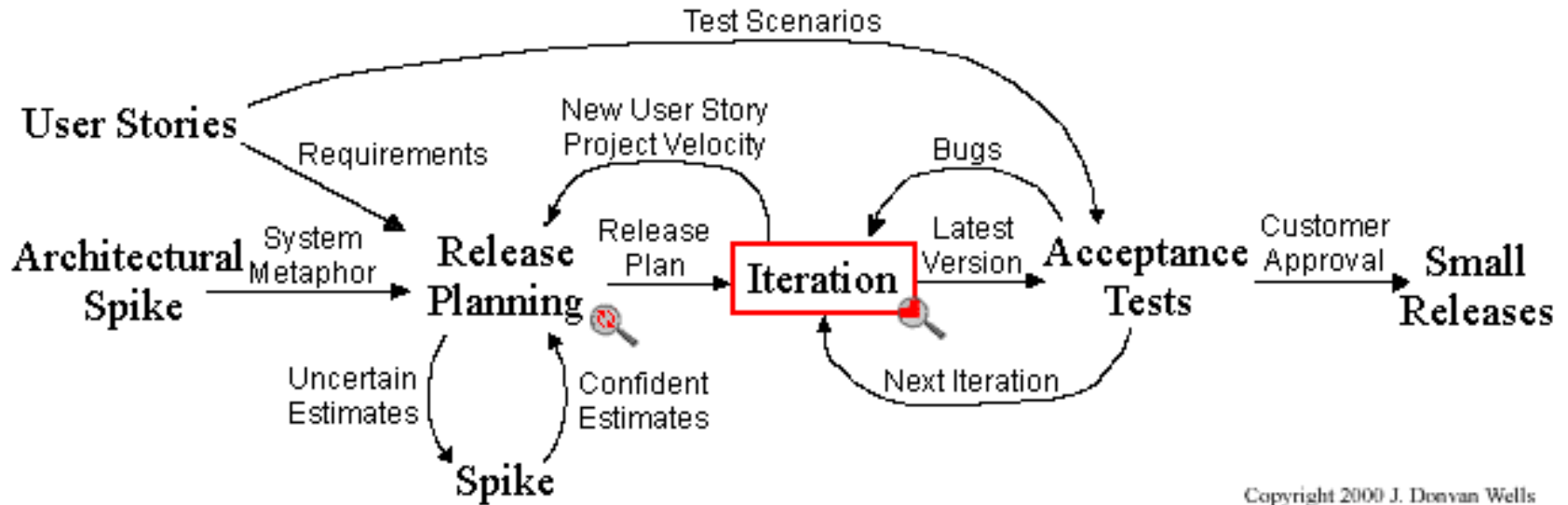
Extreme Programming (XP)

Kent Beck, mid-1990's

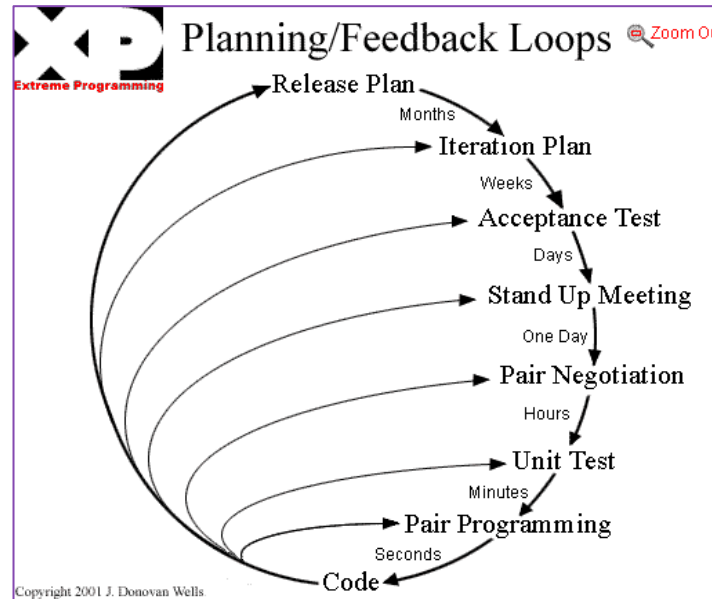
- Focuses on the customer, the development team, and the need for change
- A set of short development cycles instead of a long one
- Changes are desirable and unavoidable, so plan around that notion rather than pretending to define a stable set of requirements *a priori*
- Four activities: coding, testing, listening, and designing



Extreme Programming Project



Copyright 2000 J. Donovan Wells



Copyright 2001 J. Donovan Wells

XP: Coding

- “It’s the source, Luke!”
- It’s a way to communicate both with the computer and also with other developers in a more precise way

XP: Testing

- If a little testing is good, a lot of testing is better
- Unit tests essentially represent the specification of a feature works
- When all the tests pass, the coding is complete
- All code is tested as it is developed

XP: Listening

- The developers have to listen, carefully, to the customers
- Not only to figure out what to build, but also to provide clear information about the technical aspects and costs of how the problem might be solved (or not)
- User stories (roughly, use cases) are a key vehicle for this
- The Planning Game – meeting once/iteration (often weekly) with both release planning – customers and developers determine what features should go in upcoming releases – and iteration planning – developers defines the tasks and activities to achieve the releases
 - Intent of planning is to steer the project in the right direction

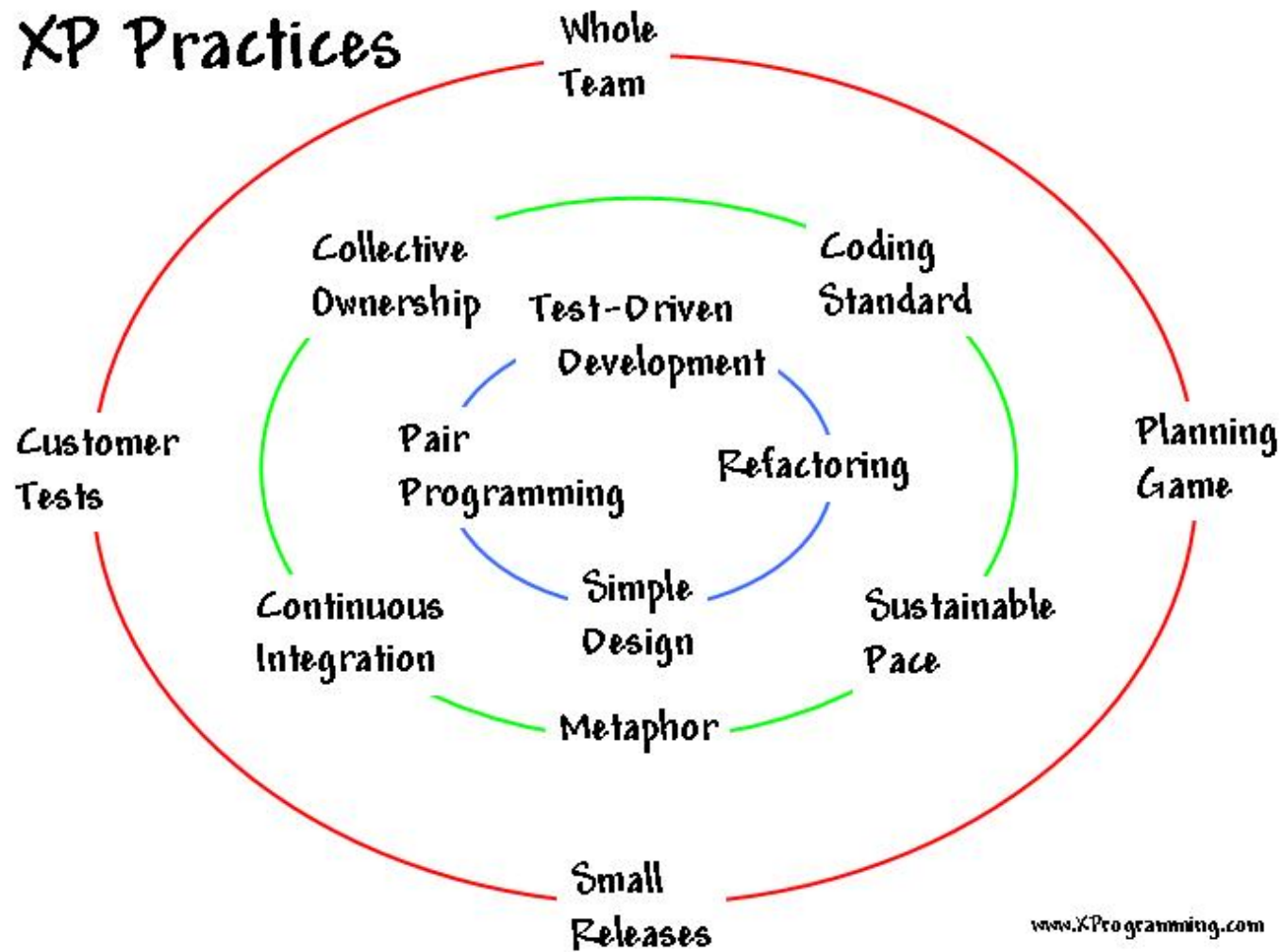
XP: Designing

- “Good design will avoid lots of dependencies within a system; this means that changing one part of the system will not affect other parts of the system.”
 - In XP, this goes with the notion of simplicity – don’t anticipate change ♦ but when you see an actual need for abstraction, do it
- ♦ We’ll look next week, and in Reading III, at the more common notion of information hiding and anticipating change

XP Values

- Simplicity: “We will do what is needed and asked for, but no more. ... We will take small simple steps to our goal and mitigate failures as they happen. We will create something we are proud of and maintain it long term for reasonable costs.”
- Communication: ...
- Feedback: “We will take every iteration commitment seriously by delivering working software.”
- Respect: “Everyone gives and feels the respect they deserve as a valued team member. Everyone contributes value even if it's simply enthusiasm. Developers respect the expertise of the customers and vice versa. Management respects our right to accept responsibility and receive authority over our own work.”
- Courage: “We will tell the truth about progress and estimates. ...”

XP Practices



12 in four categories

[Previous slide has 13, including refactoring]

- Fine scale feedback
 - Pair programming
 - Planning game
 - Test driven development
 - Whole team
- Continuous process
 - Continuous integration
 - Design improvement
 - Small releases
- Shared understanding
 - Coding standard
 - Collective code ownership
 - Simple design
 - System metaphor
- Programmer welfare
 - Sustainable pace

Is it XP unless it follows all practices?

Comparison to SCRUM?

Extra credit

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

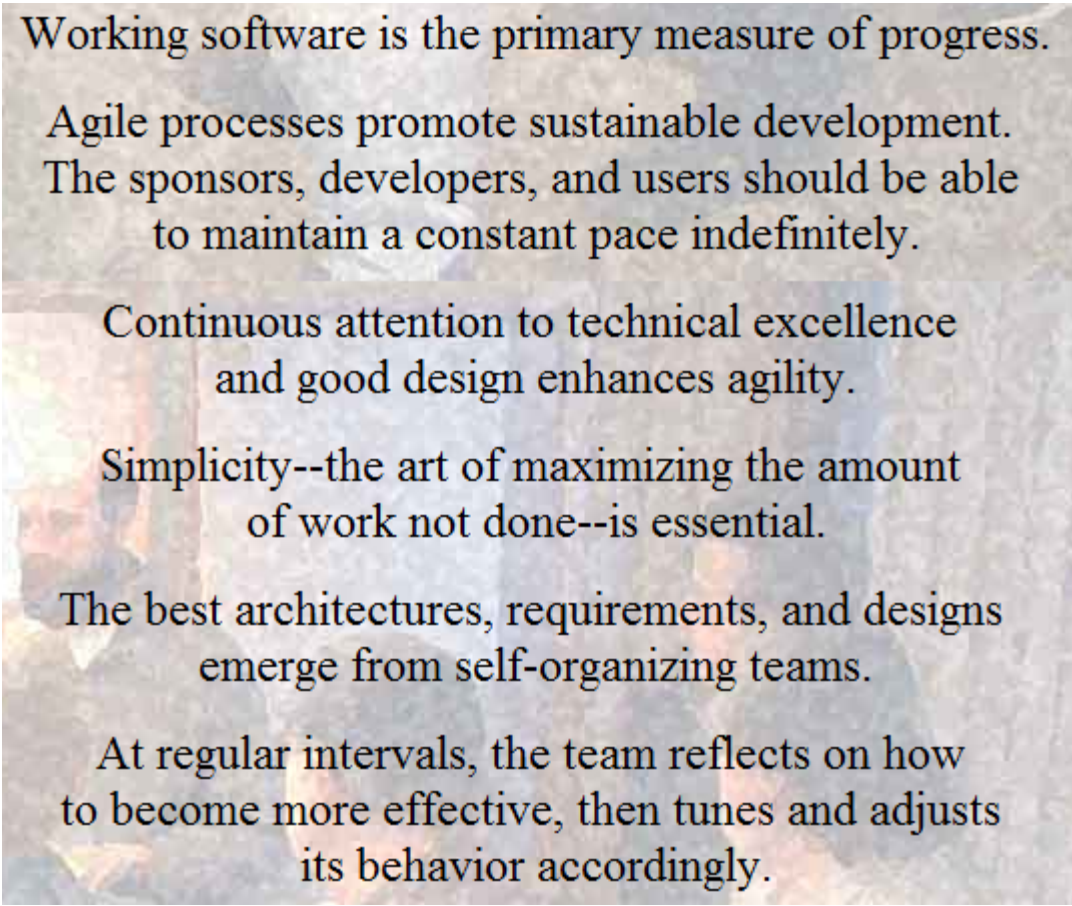
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.



Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity--the art of maximizing the amount
of work not done--is essential.

The best architectures, requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects on how
to become more effective, then tunes and adjusts
its behavior accordingly.

WebQ feedback: easy comments

I think it's too early to really tell.

meeting with TA, teaming up with people

The website is pretty poorly formatted-- that's probably the most/only annoying thing so far.

It is okay.

Fun times with a big project

WebQ feedback: other comments

I have a way to manage project *code*, but I'd like an easy way to keep track of tasks for members in my group. I'd like to be able to assign tasks to members, track due dates, keep a work log history, etc. And it should have a sexy UI. And it should be free and come with lots of awesome. Thanks.

I am a little bit confused so far about what the exams will be like. Is it going to be about the readings? Is it going to be about the topics discussed in lecture? Is it going to be multiple choice?

I don't have a rough idea about what I am gonna learn throughout this course.

The readings are way too long and thorough, and because of that it is less useful. A more concise article would be more useful, as one would be reading only the important points, which would be easier to understand, as we read exactly what is important.

WebQ feedback: other comments

So far the class seems to be going fairly well. The lectures are moving a little slowly though. Other than that, I have no complaints or suggestions.

I am a little bit confused so far about what the exams will be like. Is it going to be about the readings? Is it going to be about the topics discussed in lecture? Is it going to be multiple choice?

I am concern about the midterms and how we will be tested.

Week 2 -3

Monday	Tuesday	Wednesday	Thursday	Friday
<ul style="list-style-type: none">• <i>Requirements</i>• ...	<ul style="list-style-type: none">• Group meetings	<ul style="list-style-type: none">• Team work and structure	<ul style="list-style-type: none">• SRS information	<ul style="list-style-type: none">• Agile
<ul style="list-style-type: none">• Design	<ul style="list-style-type: none">• Group meetings	<ul style="list-style-type: none">• Design	<ul style="list-style-type: none">• Section	<ul style="list-style-type: none">• Design

- Weekly team summary: due Friday @ 11PM on DropBox by each PM
- SRS: due Tuesday April 10 @ 11PM on DropBox by each PM

Any questions?