Agile Software Development

What is it?

- Agile software development is a conceptual framework for undertaking software engineering projects.
- There are a number of agile software development methodologies.

Common elements in Agile methods

- Attempt to minimize risk by developing software in short **iterations**.
  - 1-4 weeks.
- Each iteration is like a miniature software project of its own.
  - Planning, requirements analysis, design, coding, testing, and documentation.
- Intends to be capable of releasing new software at the end of every iteration.
- At the end of each iteration, the team reevaluates project priorities.

Common elements in Agile methods (cont’d)

- Agile methods emphasize real-time communication, preferably face-to-face, over written documents.
- Most agile teams are located in a bullpen and include all the people necessary to finish software.
  - At a minimum, this includes programmers and their "customers."
  - The bullpen may also include testers, interaction designers, technical writers, artists, and managers.
- They also emphasize working software as the primary measure of progress.

Manifesto for Agile Software Development

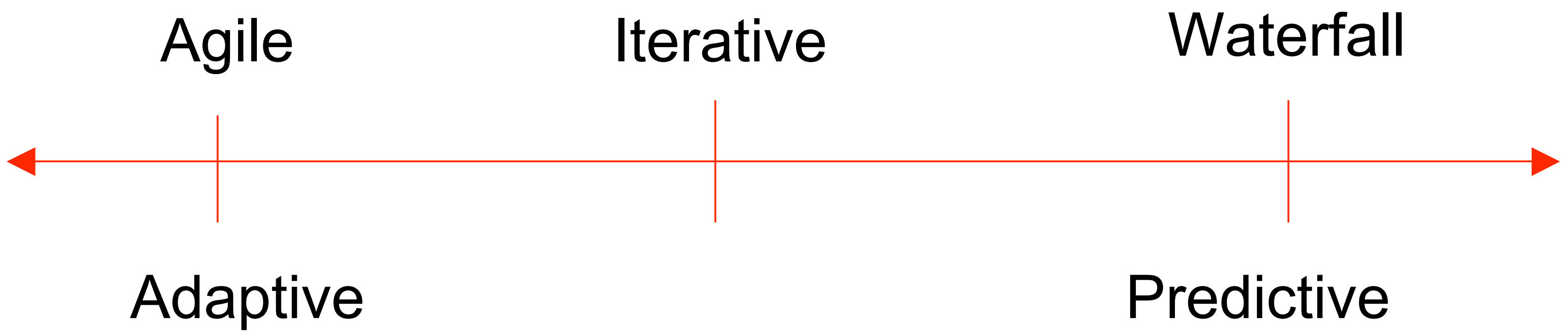
- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - 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

Comparison with other methods

- Agile methods are often characterized as being at the opposite end of a spectrum from "plan-driven" or "disciplined" methodologies.
  - This distinction is misleading, as it implies that agile methods are "unplanned" or "undisciplined."
- A more accurate distinction is to say that methods exist on a continuum from "adaptive" to "predictive."
  - Agile methods exist on the "adaptive" side of this continuum.

Methodologies Spectrum



Adaptive methods

- Focus on adapting quickly to changing realities.
  - When the needs of a project change, an adaptive team changes as well.
- An adaptive team will have difficulty describing exactly what will happen in the future.
  - The further away a date is, the more vague an adaptive method will be about what will happen on that date.
  - An adaptive team can report exactly what tasks are being done next week, but only which features are planned for next month.
  - When asked about a release six months from now, an adaptive team may only be able to report the mission statement for the release, or a statement of expected value vs. cost.

Predictive methods

- Focus on planning the future in detail.
  - A predictive team can report exactly what features and tasks are planned for the entire length of the development process.
- Predictive teams have difficulty changing direction.
  - The plan is typically optimized for the original destination and changing direction can cause completed work to be thrown away and done over differently.
- Predictive teams will often institute a change control board to ensure that only the most valuable changes are considered.

Contrasted with iterative development

- Most agile methods share iterative development's emphasis on building releasable software in short time periods.
- Agile methods differ from iterative methods in that their time period is measured in weeks rather than months.
- Most agile methods also differ by treating their time period as a strict timebox.
- Some agile teams use the waterfall model on a small scale, repeating the entire cycle in every iteration.
- Other teams, most notably Extreme Programming teams, work on activities simultaneously.

When to use agile methods

- Agile development has been widely documented as working well for small (<10 developers) co-located teams.
- Agile development is particularly indicated for teams facing unpredictable or rapidly changing requirements.
- Agile development's applicability to the following scenarios is open to question:
  - Large scale development efforts (>20 developers)
  - Distributed development efforts (non-co-located teams)
  - Mission- and life-critical efforts
  - Command-and-control company cultures

Boehm and Turner's risk-based approach

from: Boehm, B. and Turner, R. 2003. Using Risk to Balance Agile and Plan-Driven Methods. Computer 36, 6 (Jun. 2003), 57-66.

- They suggest that risk analysis be used to choose between adaptive ("agile") and predictive ("plan-driven") methods. The authors suggest that each side of the continuum has its own home ground.
- By analyzing the project against these home grounds, the risk of using an agile or plan-driven method can be determined.

Boehm and Turner's risk-based approach

- Agile home ground:
  - Low criticality
  - Senior developers
  - High requirements change
  - Small number of developers
  - Culture that thrives on chaos
- Plan-driven home ground:
  - High criticality
  - Junior developers
  - Low requirements change
  - Large number of developers
  - Culture that demands order

Agile methodologies

- Some of well-known agile software development methodologies include
  - Extreme Programming (XP)
  - Scrum
  - Adaptive Software Development (ASD)
  - Crystal Clear and Other Crystal Methodologies
  - Feature Driven Development
  - Lean software development

Agile methodologies

- Other methodologies include
  - Agile documentation
  - Agile ICONIX
  - Microsoft Solutions Framework (MSF)
  - Agile Data
  - Agile Modeling

- Examples of similar concepts beyond the realm of software include
  - Lean manufacturing

Extreme Programming

What is Extreme Programming (XP)?

- XP is successful because it stresses customer satisfaction.
- This methodology also emphasizes team work.
  - XP implements a simple, yet effective way to enable groupware style development.
- XP improves a software project in four essential ways: communication, simplicity, feedback, and courage.

New Way of Programming

- A typical project will spend about twenty times as much on people than on hardware.
  - E.g. A project spending 2 million dollars on programmers per year will spend about 100 thousand dollars on computer equipment each year.
  - Compare saving 20% of the hardware costs by some very clever programming tricks vs. saving no less than 10% of people costs by writing programs such that they are easy to understand and extend.

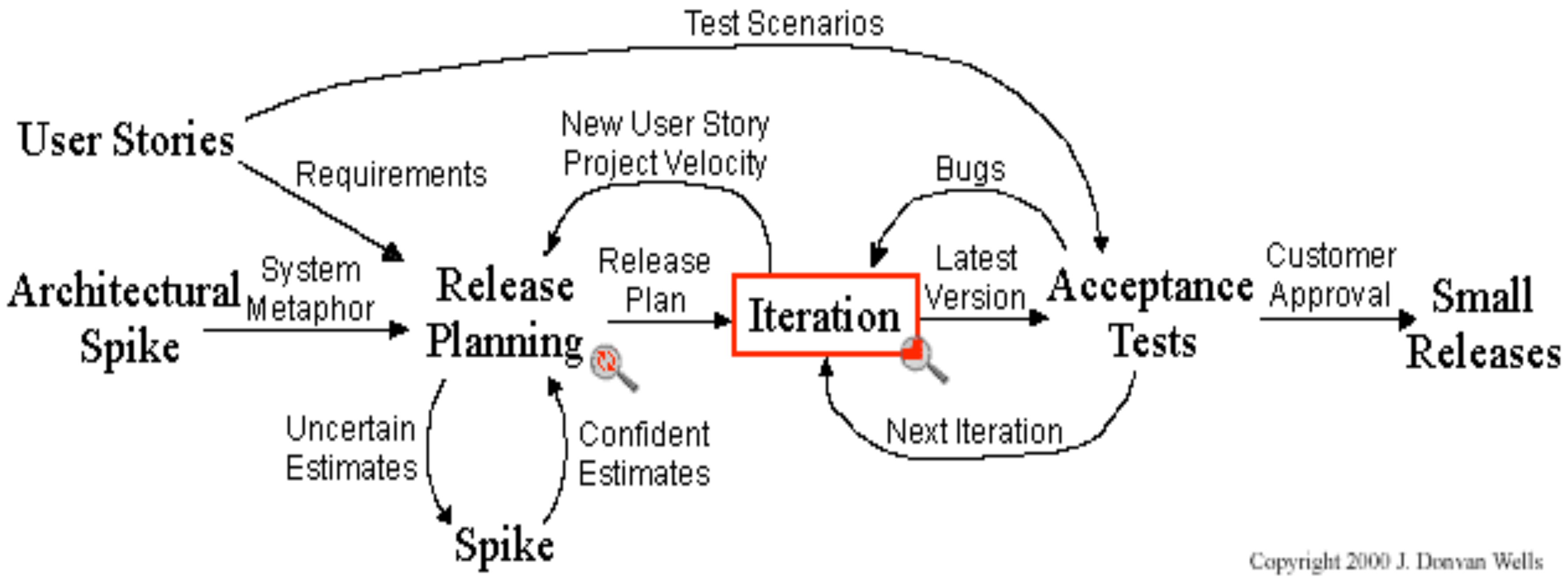
New Way of Programming

- Software which is engineered to be simple and elegant.
  - A typical project will spend about twenty times as much on people than on hardware.
- XP emphasizes not just testing, but testing well.
  - Tests are automated and provide a safety net for programmers and customers alike.
  - Tests are created before the code is written, while the code is written, and after the code is written.
  - As bugs are found new tests are added. Bugs don't get through twice, and this is something the customers will notice.
- Getting customer feed back early while there is still time to change functionality or improve user acceptance.

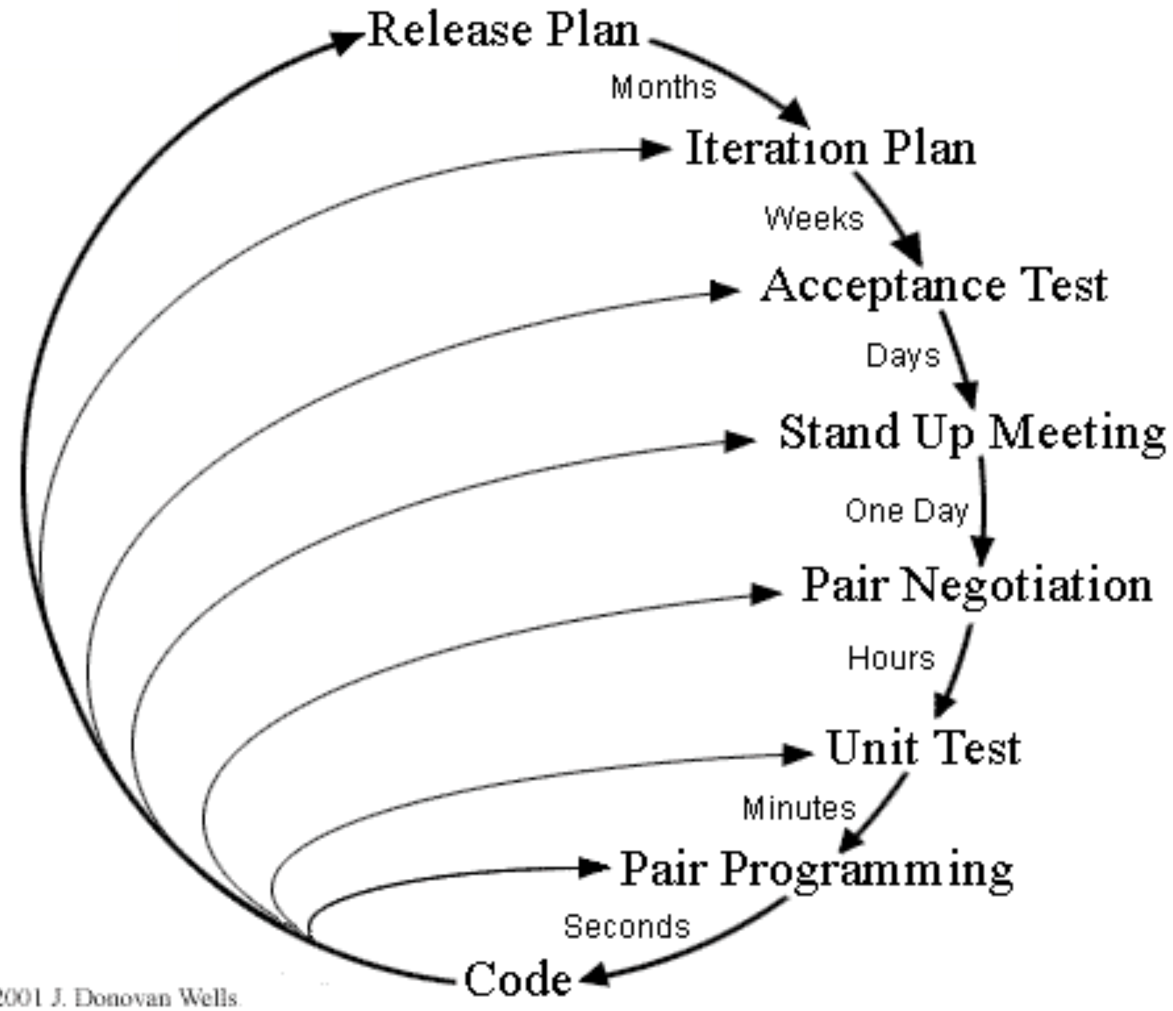
When should Extreme Programming be Used?

- XP was created in response to problem domains whose requirements change.
- It was also set up to address the problems of project risk.
- XP is set up for small groups of programmers. Between 2 and 12, though larger projects of 30 have reported success.
- Another requirement is testability. You must be able to create automated unit and functional tests.
- Greater programmer productivity.

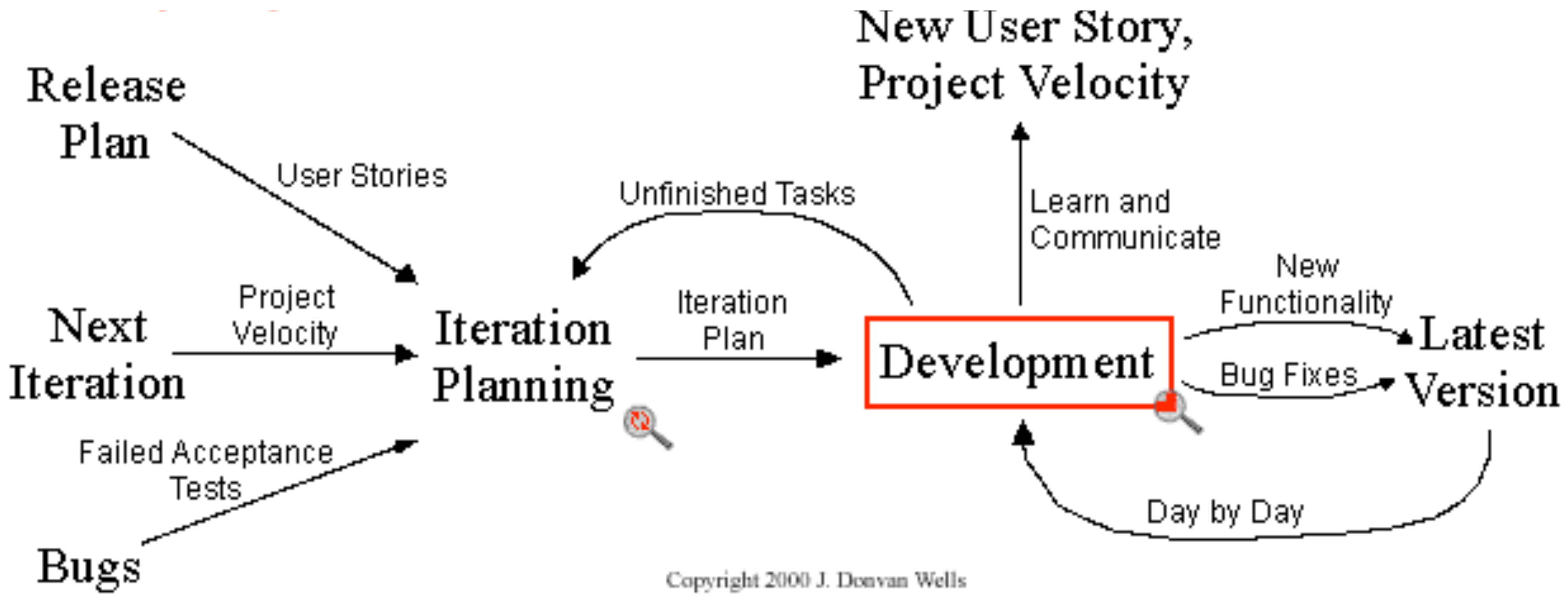
Extreme Programming Project



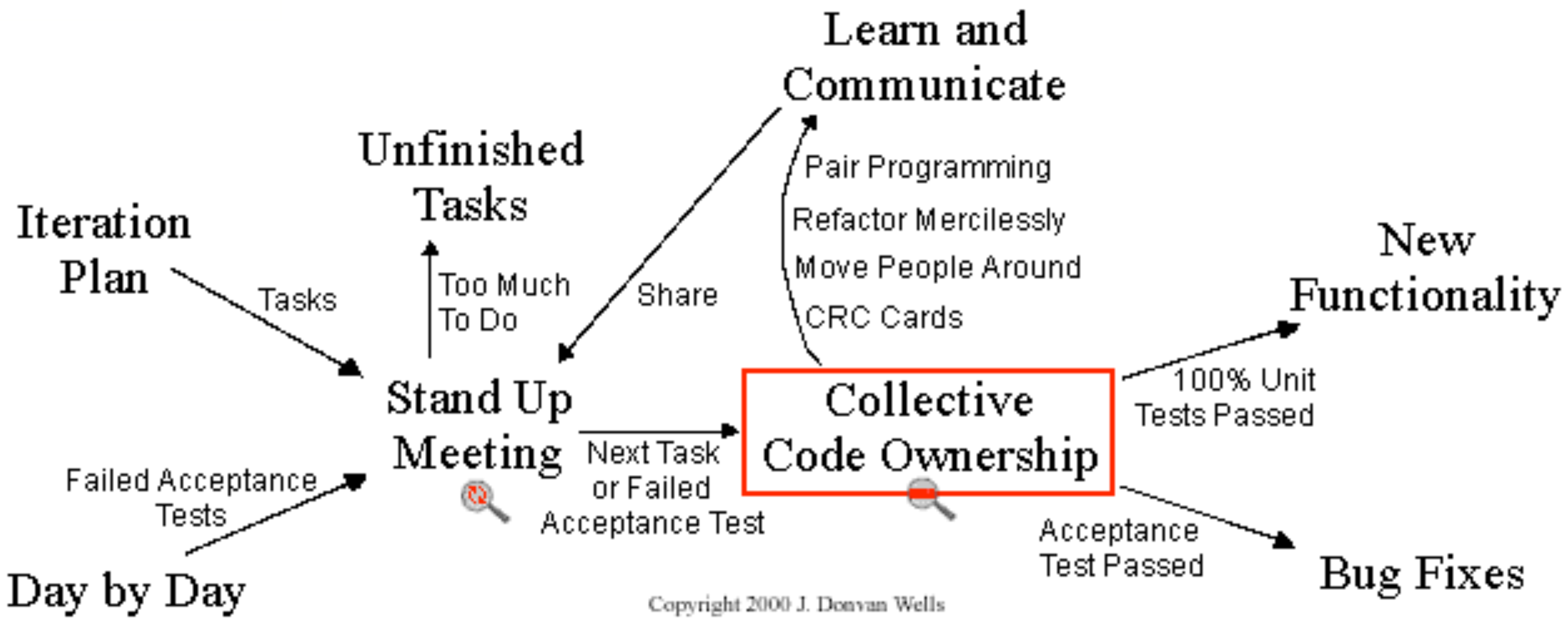
Planning/Feedback Loops



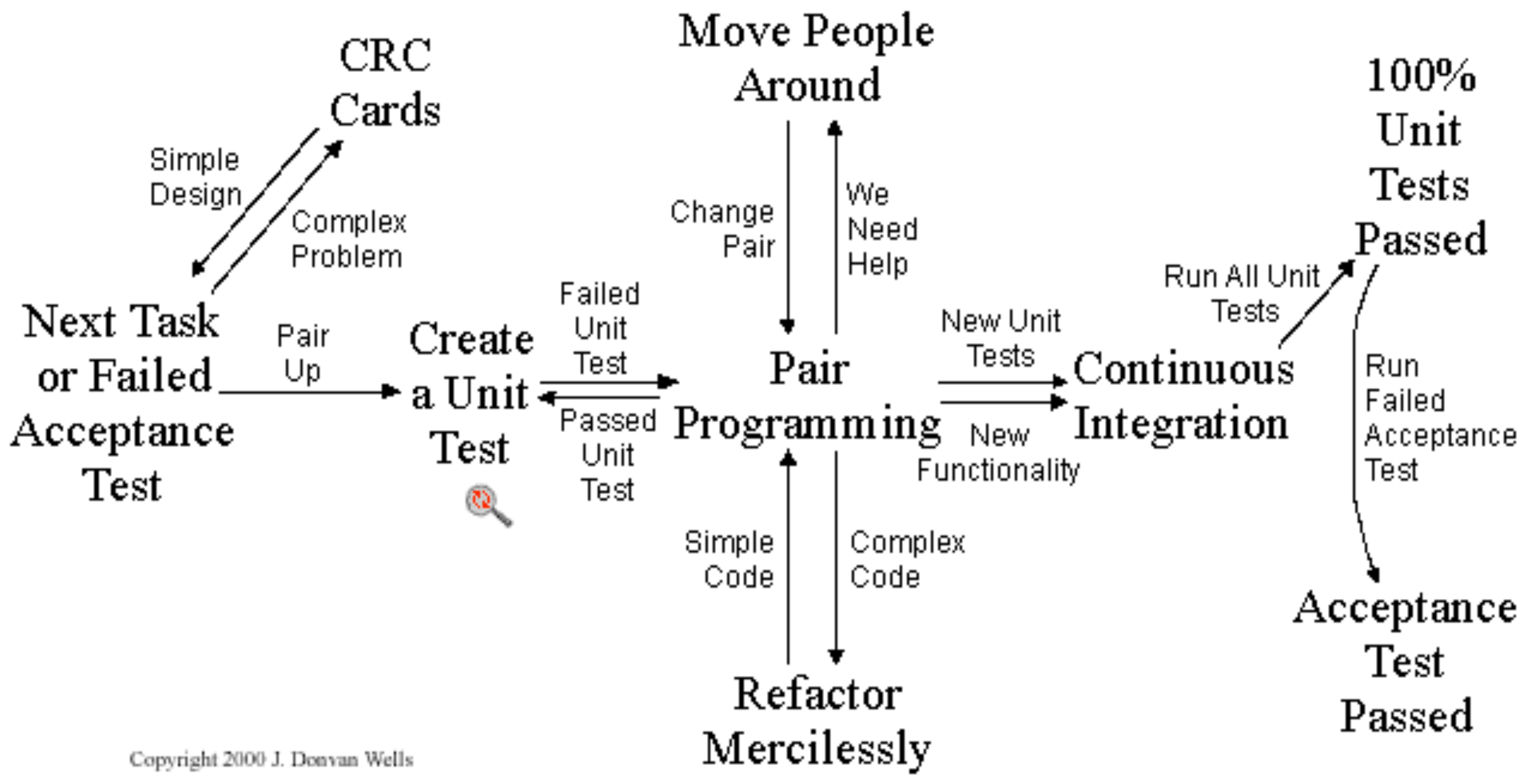
Iteration


Release Plan
User Stories
Next Iteration
Project Velocity
Failed Acceptance Tests
Bugs
Iteration Planning
Unfinished Tasks
Iteration Plan
Development
New User Story, Project Velocity
Learn and Communicate
New Functionality
Bug Fixes
Latest Version
Day by Day
Copyright 2000 J. Donvan Wells

Development



Collective Code Ownership



Further Reading:

- More on XP:
  - http://www.xprogramming.com/xpmag/whatisxp.htm
- For a contrasting view:
  - http://www.softwarereality.com/lifecycle/xp/case_against_xp.jsp
- A more balanced view:
  - http://www.martinfowler.com/articles/designDead.html