Introduction to Agile/Extreme Programming

Matt Ganis
Senior Technical Staff Member
(Certified Scrum Master)

IBM – Hawthorne, New York
ganis@us.ibm.com

Share 2008 – Orlando Session: 8381

Current slides at: http://webpage.pace.edu/mganis

Agenda

- Overview of Agile
- What are some of the components/practices
- How is this different from what we do today
- Some hints/tips/suggestions

Definition of Agile[1]

*Agile is an **iterative** and **incremental** (evolutionary) approach to software development which is performed in a highly collaborative manner with "**just enough**" ceremony that produces high quality software which meets the changing needs of its stakeholders.*

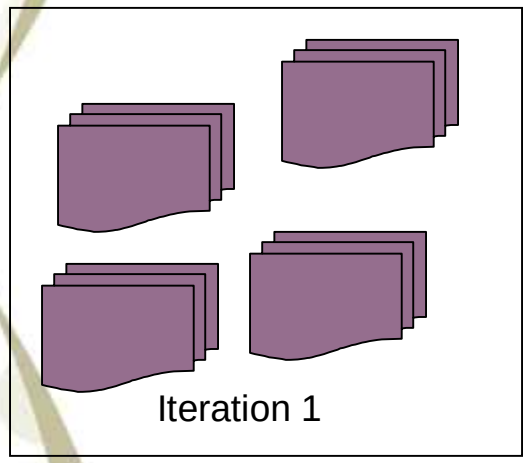
[1] http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

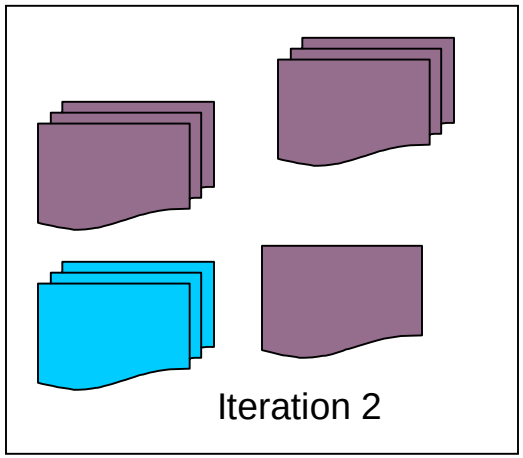
What is Agile ?

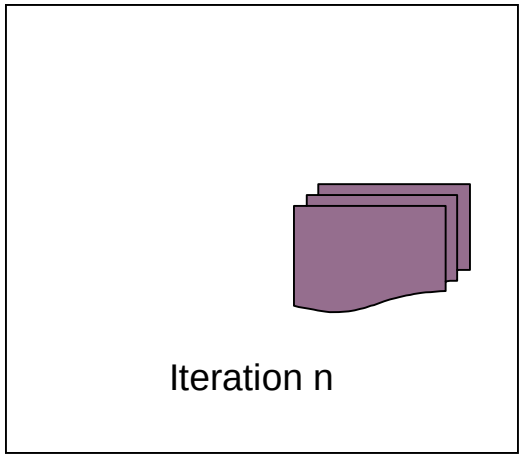
- XP, SCRUM, DSDM, Adaptive Software Development, Crystal, FDD
  - February 2001 (Snowbird, UT)
  - Agile Alliance
- Started as 17 methodology authors and practitioners
- Agile Manifesto Values:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

http://www.agilemanifesto.org/

The “promise” of Agile







Within an iteration, stories are created

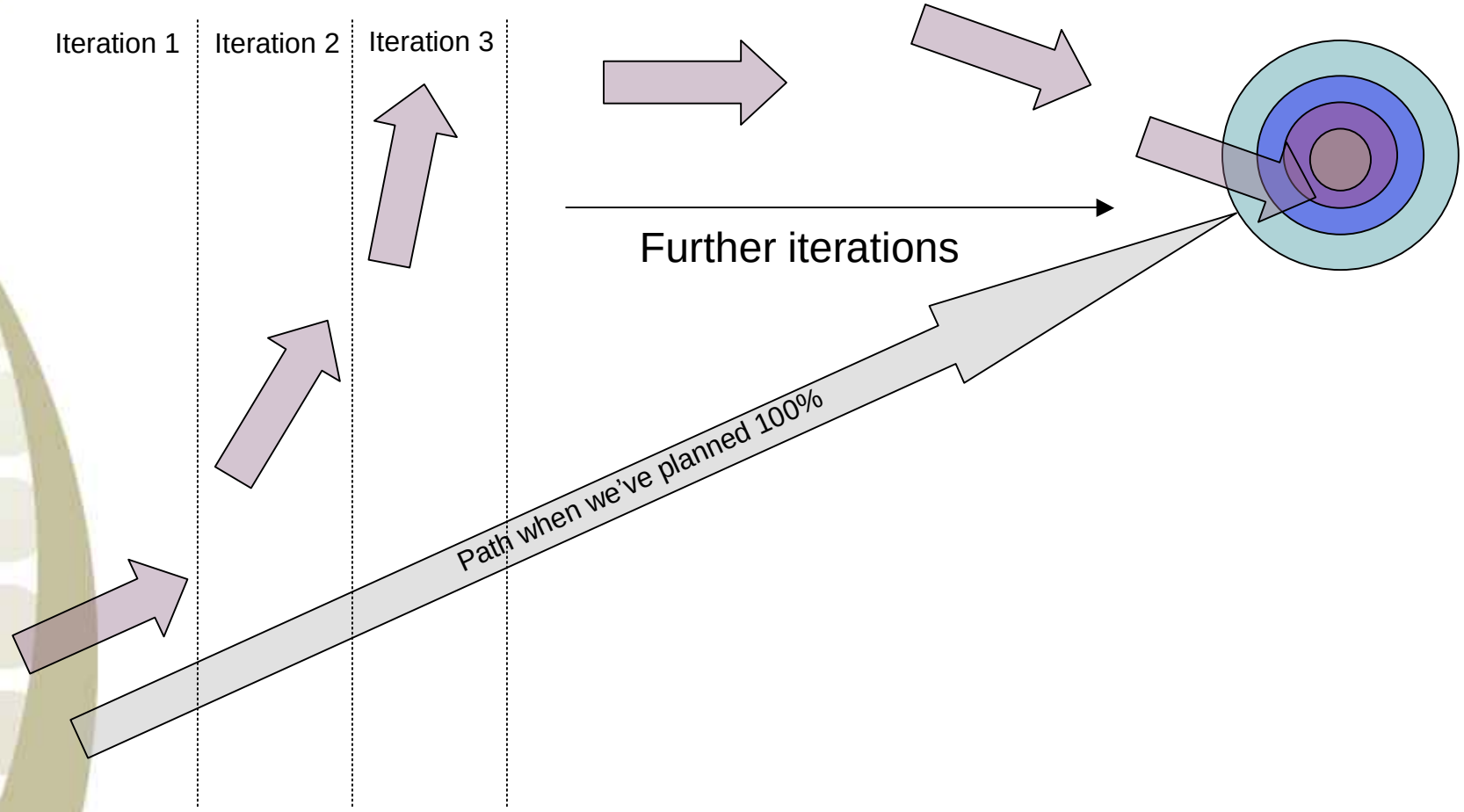
In a planning game, stories are selected by the customer based on value

Releases are composed of a number of iterations, as the iterations progress, stories are completed, and new ones are introduced

At some point, there exists a deliverable, that delivers enough value that the customer says “stop” since the remaining stories don’t contribute sufficient value

Agile allows for faster deliverables at a lower cost (assuming the customer decides, based on what they see, that a set of stories aren’t needed)

Why is Iterative development “good”?



What is Agile

- Agile Software methodologies and practices emphasize:
  - Empirical process control
  - Emergence
  - Self-organization
- Agile methodologies span the spectrum between“hacking” and “milestone plan-driven” development
  - They are VERY disciplined
  - VERY flexible
    (like a gun, they can be quite effective and dangerous at the same time)

What is Agile? (Incremental, Iterative, Adaptive)

- Incremental
  - *Build a system gradually*
  - *See the system grow*
  - *Demonstrating progress*
- Iterative
  - *Multiple releases or check points during a project, each closer to the target*
  - *Iterations include requirements development and testing*
  - *Typical iterations are two weeks long*
  - *Plan as you go*
- Adaptive
  - *Goals change based on lessons from prior iterations, feedback and business opportunities*

Agile Methods

- Agile methods tend to fall into two categories:
  - Engineering (the “how to do it”)
  - Project Management (the “how to manage it”)
- ***Both*** operate in an iterative manner

Why Agile?

Organizations are interested in Agile development

- Forrester Research indicates that 2/3 of their clients are interested in Agile capabilities to deliver more value to the business faster
- Clients are asking for it
- It has become even more popular in recent years due to easy-to-follow publications and internet buzz

The demand for Agile development (cont)

- **Many clients are struggling with application delivery issues**
  - Poor I/T relationship with the business
  - Track record of poor project delivery
  - Inability to deliver on-time, on-budget
  - Inability to delivery solutions that meet the needs of the business
  - Poor results with offshore delivery *or* seek to avoid offshore delivery
  - Large project backlog
- **Internal and external IBM delivery projects are interested in Agile techniques to help address delivery excellence challenges**
  - Requirements often are not well-defined
  - Speed time to deliver critical business functionality
  - Reduce technical risk for first of a kind solutions

Agile Survey[2]

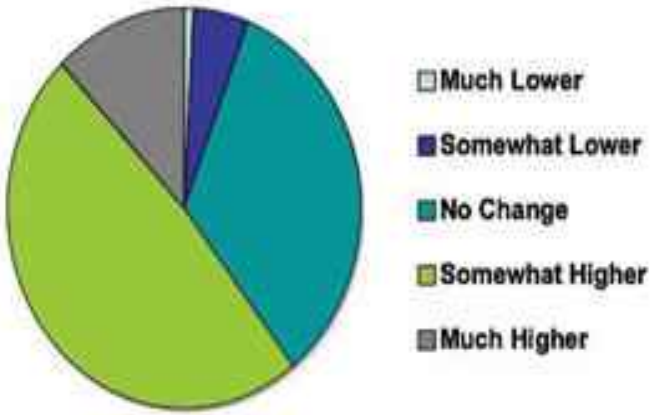
Agile Adoption Rates

A March 2006 survey of 4232 IT professionals shows:

- 65 percent work in organizations that have adopted one or more agile development techniques
- 41 percent work in organizations that have adopted one or more agile methodologies
- 60 percent report increased productivity
- 66 percent report increased quality
- 58 percent report improved stakeholder satisfaction

[2] Results of an Industry survey by Scott Ambler: http://www.ddj.com/dept/architect/191800169?pgno=1

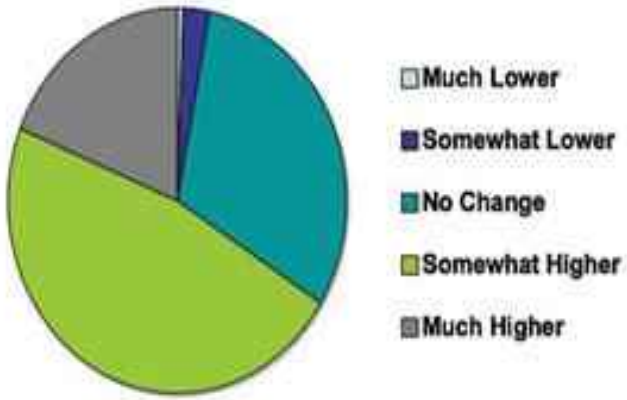
How has Agile affected your productivity?
Survey says:



How has Agile affected your productivity?

How has Agile affected your quality of deployed systems?

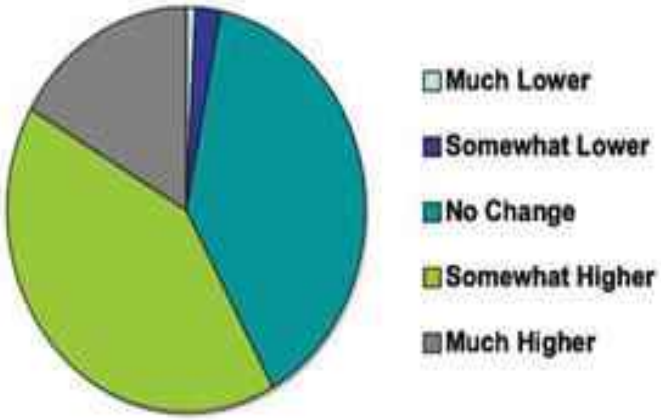
Survey says:



How has Agile affected your quality of deployed systems?

How has Agile affected the stakeholders satisfaction?

Survey says:



How has Agile affected the stakeholders satisfaction?

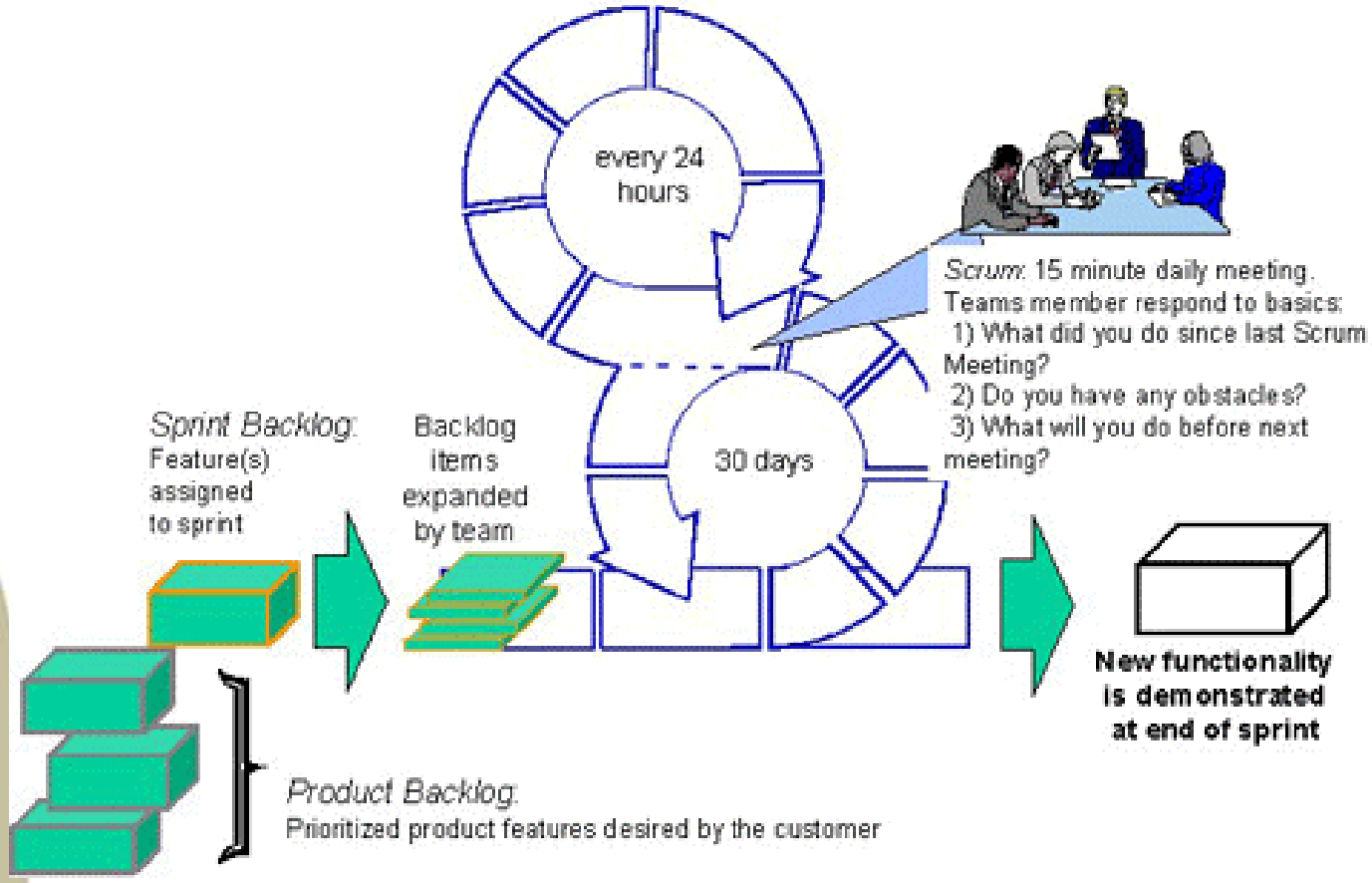
Scrum (Project Management)

Scrum overview

Scrum is an iterative, incremental process for developing any product or managing any work. It produces a potentially shippable set of functionality at the end of every iteration. It's attributes are:

- An agile process to manage and control development work.
  - ✓A team-based approach to iteratively, incrementally develop systems and products when requirements are rapidly changing
  - ✓ a way to detect and cause the removal of anything that gets in the way of developing and delivering products.
  - ✓ Scrum is scalable from single projects to entire organizations. Scrum has controlled and organized development and implementation for multiple interrelated products and projects with over a thousand developers and implementers.

Scrum (project management)



What is Extreme Programming (engineering)

What is XP

- Extreme Programming (or XP) is a set of values, principles and practices for *rapidly developing* high-quality software that provides the *highest value* for the *customer* in the fastest way possible

- It's an "engineering" method in that it prescribes "HOW" to create the code unlike SCRUM that talks about "how to manage an agile project"

Some “Infrastructure” definitions

Stories

- User requirements come to the development team in what we call “Stories”

- These are short descriptions of what the customers would like to see done – not specified in any technical language – but represented as a “thought” or as an “idea”

What makes a “good” story

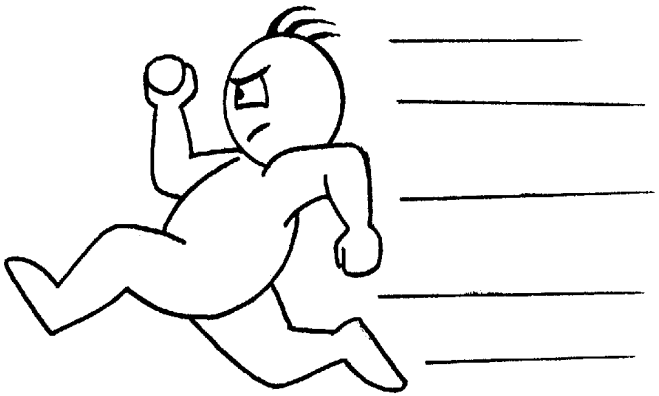
(from Bill Wake: http://www.xp123.com/xplor/xp0308/index.shtml)

SHARE
Technology · Connections · Results

I	***Independent.*** If they are independent we can schedule and build them in any order. This allows us to select stories with the highest value without worrying about all the expensive, low value dependent stories.
N	***Negotiable*** - Remember that agile methods are typically variable in scope. That is the time line is fixed (iteration length) and the quality and scope are varied. A story is a placeholder for a discussion. We need to be able to split, combine, tweak, clarify, etc stories at any time.
V	***Value.*** Stories need to have real business value to the stakeholders. Typically this is the customer although there are other stakeholders (including the development organization). They should be expressed in ways that the primary stakeholder can understand the value provided by the story.
E	***Estimate.*** If a story can't be estimated then the customer can't derive value or assign a priority to it. We don't need a precise estimate or a guarantee that the estimate will never change.
S	***Small.*** Having small stories is a result of estimable and negotiable. The larger the story the harder it is to estimate, the less flexibility in negotiation.
T	***Testable.*** Stories need to be testable, otherwise how do you know the story is complete?

“How fast can you go ?” - Your Velocity

- A project velocity is used to determine if the iteration is over booked or not.
  - Total up the time estimates in ideal programming days of the tasks, this must not exceed the project velocity from the previous iteration.

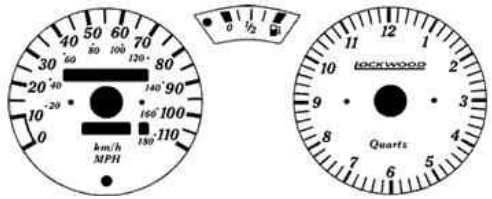


If the iteration has too much then the customer must

user stories to be put off until a later iteration

XP Practices

XP is extreme in the sense that it takes 12 well-known software development "best practices" to an extreme



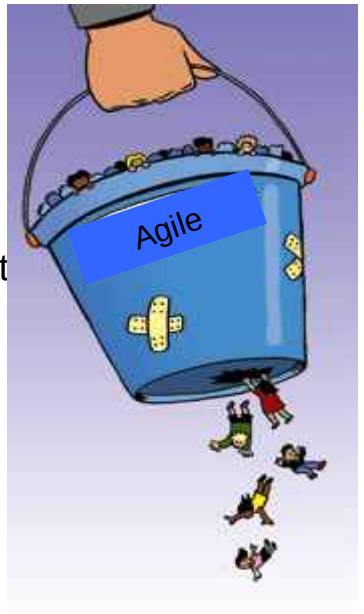
- Planning Game
- Small Releases
- Simple Design
- Continuous Testing
- Refactoring
- 40-hour work week
- Pair Programming
- Collective code ownership
- Continuous Integration
- On-Site customer
- Coding standards

Key Ideas

- Practices are synergistic and support each other

  The more holes you “poke in the bucket” the less Agile you become

  Two things remain “true”:

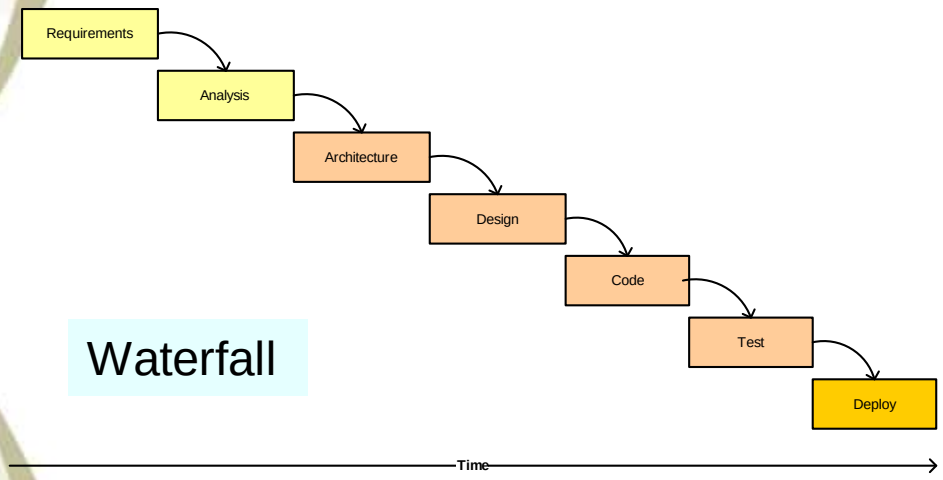


- Distance is expensive
  - Drives the need for a high degree of communicat
- Schedules never slip
  - Stories fall out, only to be done later

What is missing from traditional waterfall methods ?

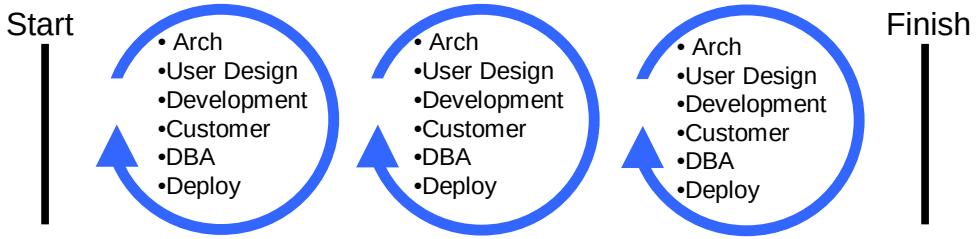
- Upfront requirements gathering and sign-off (no need to commit early
- Upfront design documents (easy to retarget)
- Early costs amortized over life of project
- Intimidation: schedule, cost, or value

What's different ?



Waterfall

VS.



Iterative

What's different ?
today we use Plan-driven methods

- Waterfall assumes requirements are understood up front and are relatively stable
- Assumes software can be "manufactured"
- Emphasizes Big-Design Up Front (BDUF)
- Step-by-step execution
  - Decouple architecture and design from coding and testing
  - Different teams for different aspects

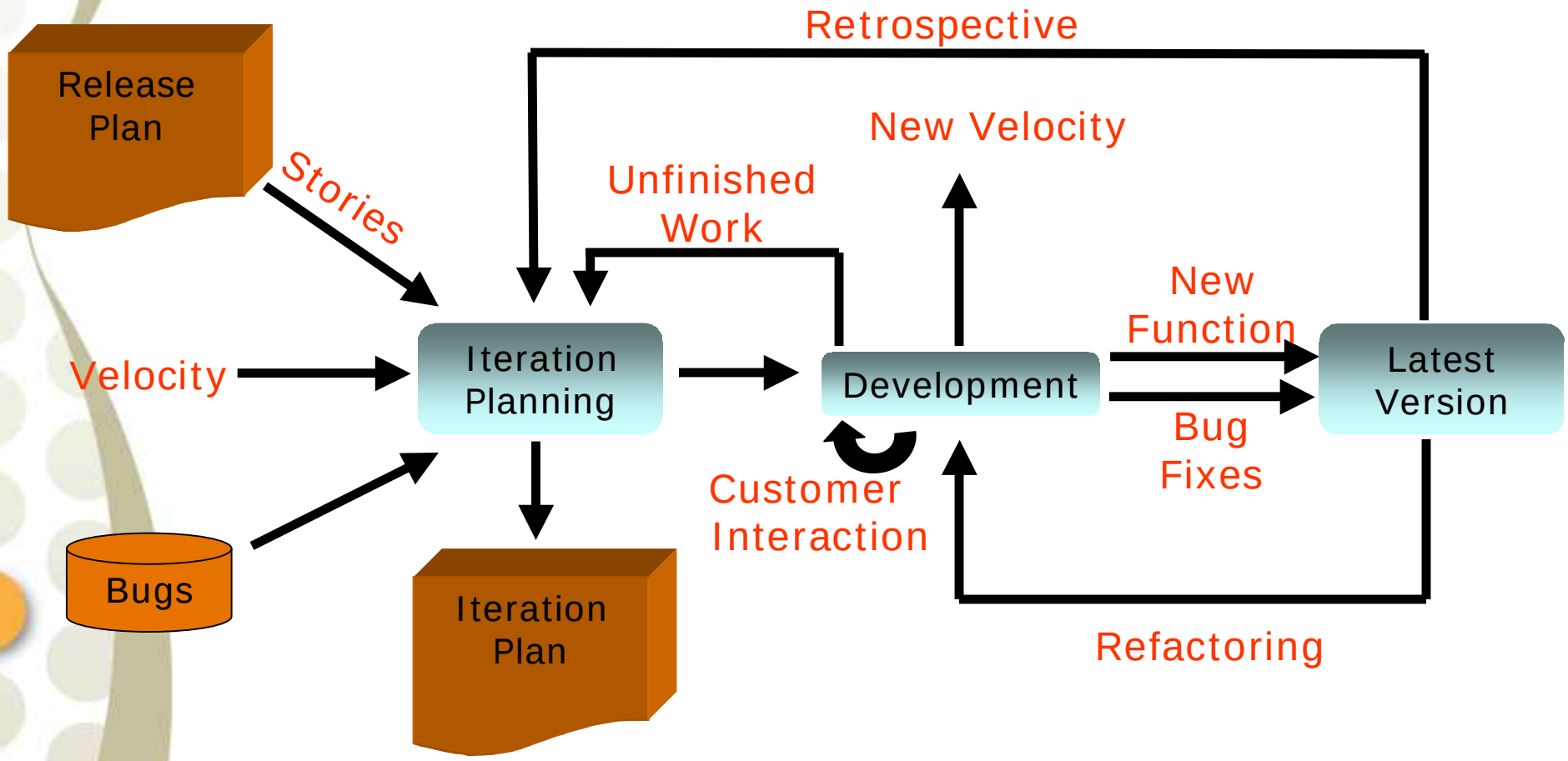
“A day in the life of an XP team…”

- XP teams work in a series of fixed iteration cycles.
  - Iterations typically last 1, 2 or 3 weeks each depending on the team. (A given team will almost always use same the iteration size for every iteration.)
- At the beginning of each iteration, the team gets together with the customer for a planning meeting. In that meeting, they go over the features the customer wants done in that iteration, breaking each feature down into individual engineering tasks.
  - Individual developers then sign up for specific tasks, and estimate those tasks. No developer is allowed to sign up for more work in the coming iteration than he completed in the previous iteration.

“A day in the life of an XP team…”
(cont)

- During the rest of the iteration, the team will implement the features they signed up for, pair programming on all production code.
  - All code is written test-first -- that is, the developers don't write any code until they have a failing test case. The developers write unit tests to test individual classes and subsystems. The customer provides functional or acceptance tests to validate the features that the programmers are developing.
- At the end of the iteration (usually on a Friday), the programmers deliver a working system to the customer. The system may not be complete, but all functionality that is implemented works completely, without bugs. The customer accepts delivery, and the team goes home early. The next Monday everyone meets again to plan the next iteration, and the cycle repeats itself.

XP flow


Retrospective
Release Plan
Stories
New Velocity
Unfinished Work
New Function
Velocity
Iteration Planning
Development
Latest Version
Bug Fixes
Customer Interaction
Bugs
Iteration Plan
Refactoring

Using these techniques, the experience has shown that a better quality of code is produced

why ?

Continuous Testing

Teams using agile methods work in short iterations to “grow” a system. These methods require the whole team to focus on quality throughout each of these iterations, ensuring that the system is built on a sound foundation.

The system must be kept in a high-quality, working condition at all times. With software builds and integration taking place on an hourly basis in some cases, there simply is no time to perform extensive manual tests. To accomplish this, the team must commit to automating as much of the testing process as possible.

(Collective code ownership, Coding standards , Integration)

What is pair programming[3]?

Two programmers working side-by-side, collaborating on the same design, algorithm, code or test. One programmer, the ***driver***, has control of the keyboard & mouse and actively implements the program. The other programmer, the ***observer***, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects and also thinks strategically about the direction of the work.

[3] Laurie Williams

(Collective code ownership, Small Releases, Simple Design, Small Release)

On-site customer

On-Site Customer describes the need to have on-site access to people, typically users or their representatives, who have the authority and ability to provide information pertaining to the system being built and to make pertinent and timely decisions regarding the requirements, and prioritization

(Continuous Integration, Small Releases, Pair Programming)

In Summary

XP the practice of all of these ideas continuously:

- "If code reviews are good, we'll review code all the time (pair programming)
- If testing is good, everybody will test all the time (unit testing), even the customers
- If design is good, we'll make it part of everybody's daily business (refactoring)
- If simplicity is good, we'll always leave the system with the simplest design that supports its current functionality (the simplest thing that could possibly work)
- If architecture is important, everybody will work defining and refining the architecture all the time
- If integration testing is important, then we'll integrate and test several times a day (continuous integration)
- If short iterations are good, we'll make the iterations really, really short ... (the Planning Game)."

Thank you !

If you have any questions, please feel free to contact me at:

ganis@us.ibm.com

(914) 784-5759

Or find these slides online at:

http://webpage.pace.edu/mganis