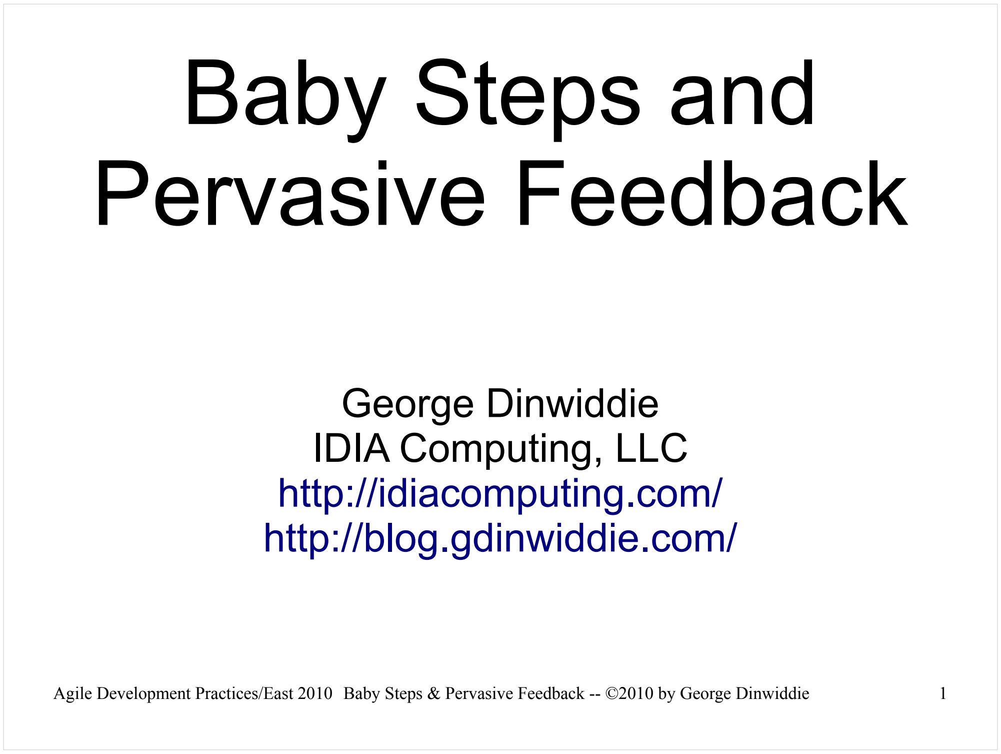
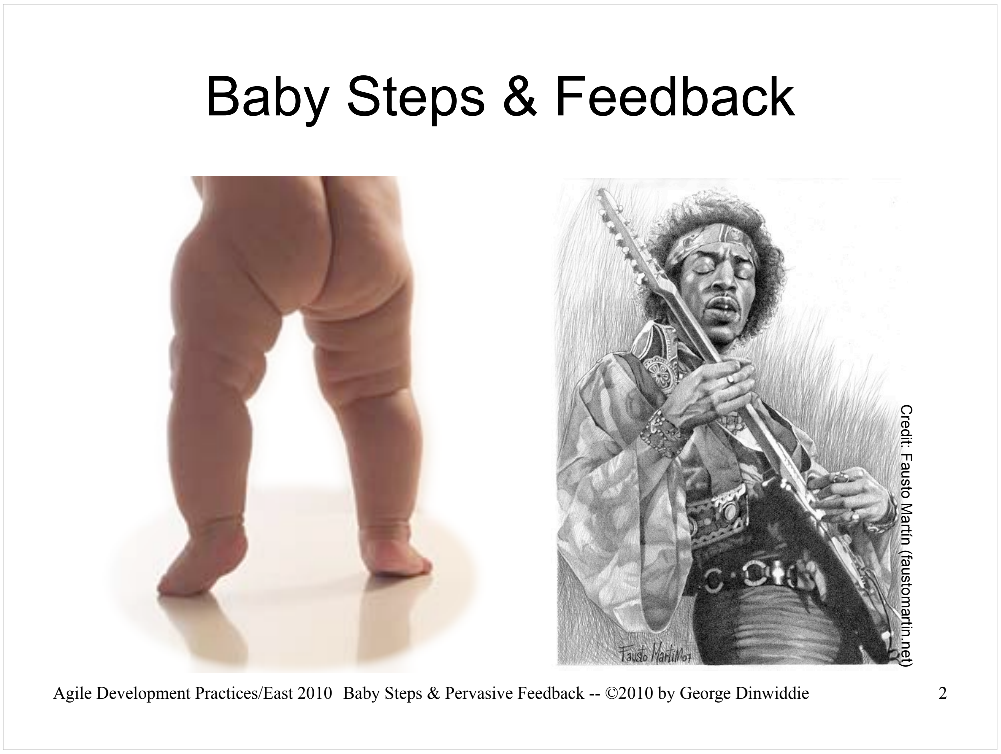


Baby Steps and Pervasive Feedback

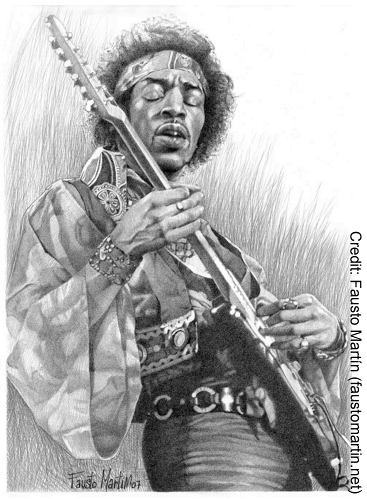
George Dinwiddie
IDIA Computing, LLC
http://idiacomputing.com/
http://blog.gdinwiddie.com/

Check http://idiacomputing.com/publications.html to see if there is an updated version of these slides.

You want to begin adopting agile practices in your team and organization. Where to start? Agile is full of strange terms, new principles, and unfamiliar practices that are not even described consistently from person-to-person and book-to-book. Customer or Product Owner? Sprint or Iteration? Then there are all those new gatherings—sprint planning, stand-ups, retrospectives, reviews, and more. As hard as you try to do things "by the book," it's hard to tell which practices really help and when to implement each. And you thought agile was supposed to be simpler!



Baby Steps & Feedback

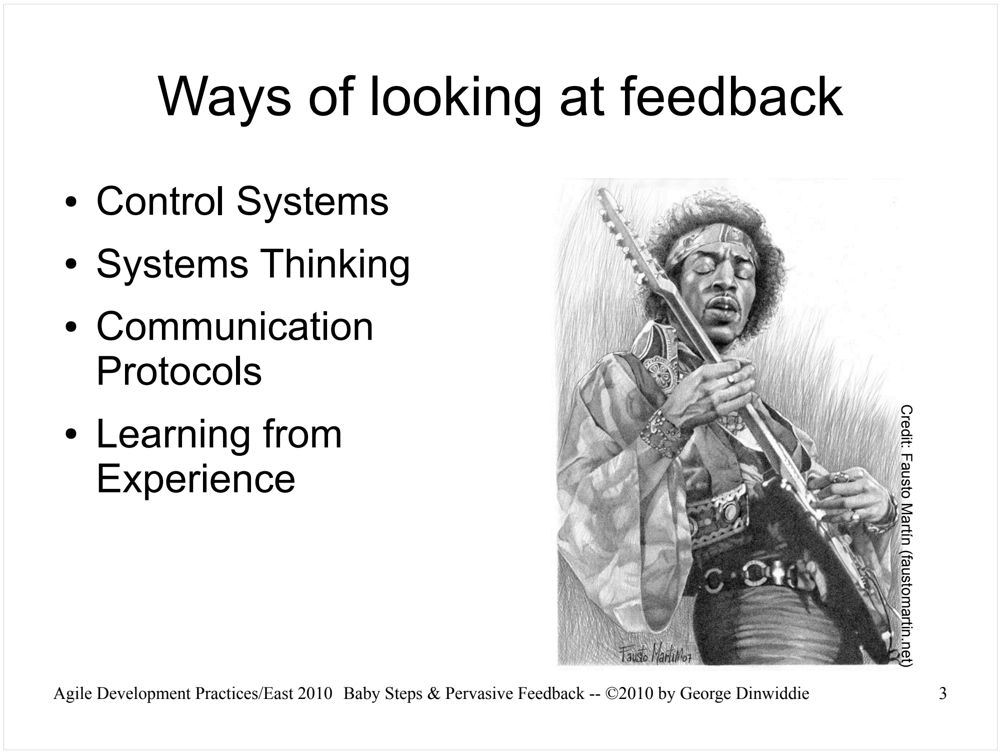


Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 2

The vast array of Agile practices can be confusing when you're starting out. We're going to take a look at them from two points of view, Baby Steps, and Pervasive Feedback, to put them into perspective.

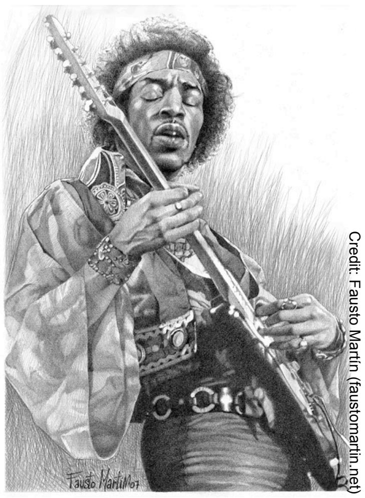
.

First we'll examine what we mean by Baby Steps and Pervasive Feedback—then we'll look at some common practices in these terms.



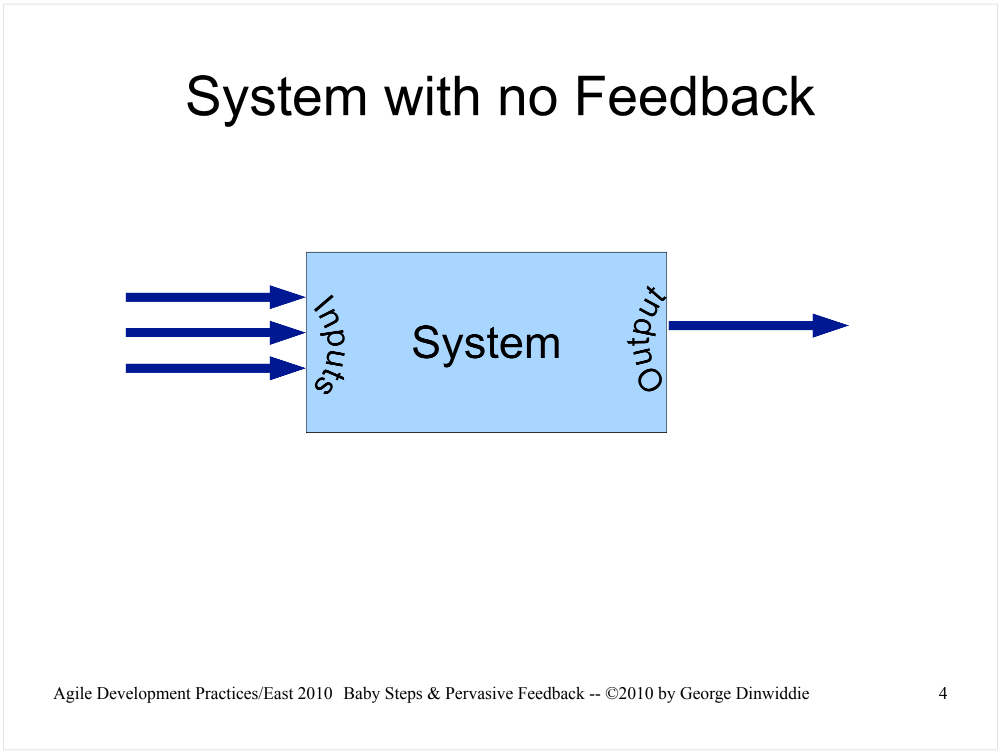
Ways of looking at feedback

- Control Systems
- Systems Thinking
- Communication Protocols
- Learning from Experience



Credit: Fausto Martin (faustomartin.net)

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 3



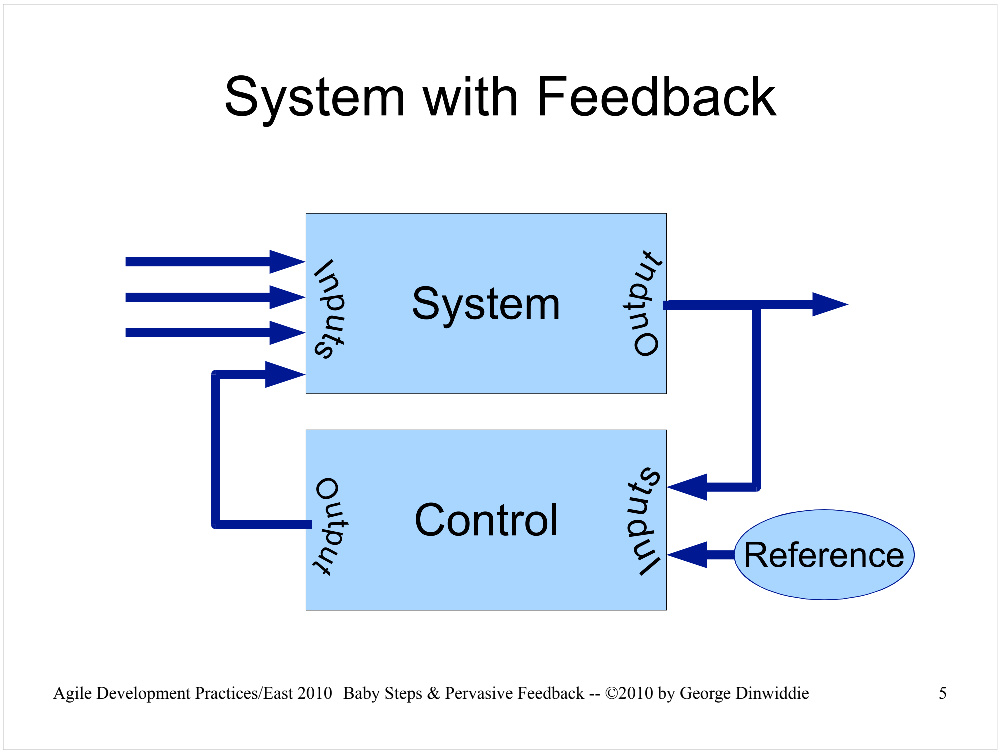
System with no Feedback

This drawing represents a simple system with no feedback mechanisms. The output is a function of the inputs, and as long as

- the inputs are what is expected,
- the system has no distortions or flaws, and
- the system is correctly specified and constructed,

then nothing can go wrong.

Our faith (or lack of it) in such an approach leads to the expression of Murphy's Law: "*Whatever can go wrong, will*" and the many corollaries and similar expressions.



System with Feedback

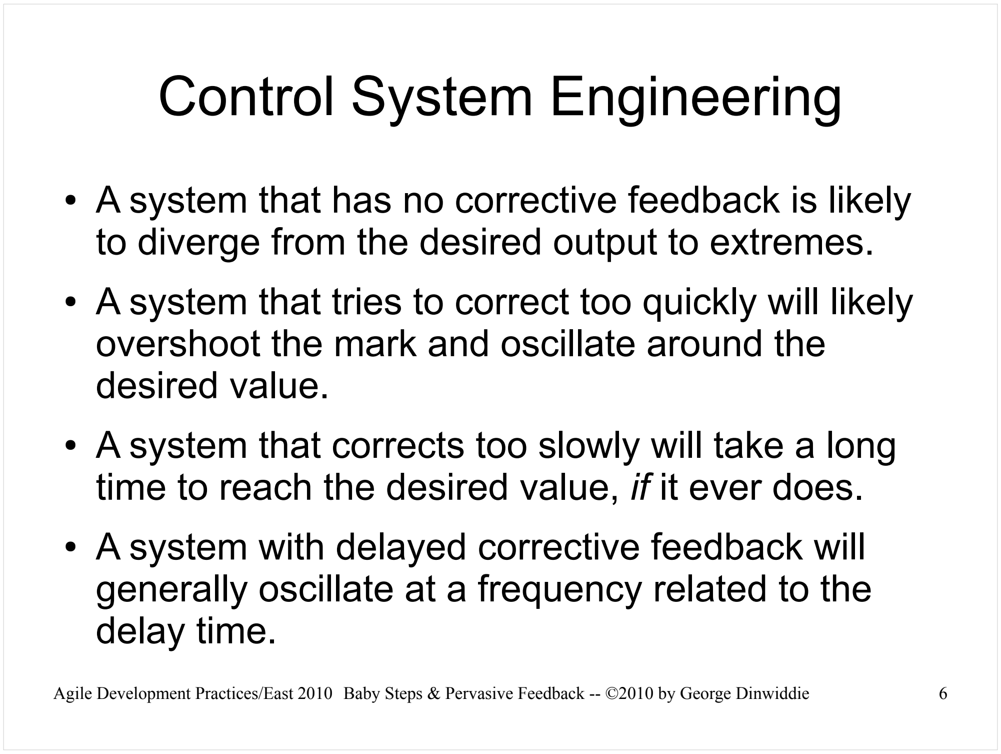
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 5

In this drawing, we've added a control system that compares our system output with an independent reference.

The output of this control system is an additional input to the system under control. The combination should stabilize the system to produce output more nearly what we want.

This is the basics of control systems engineering.

A classic example is a heating system with a thermostat.



Control System Engineering

- A system that has no corrective feedback is likely to diverge from the desired output to extremes.
- A system that tries to correct too quickly will likely overshoot the mark and oscillate around the desired value.
- A system that corrects too slowly will take a long time to reach the desired value, *if* it ever does.
- A system with delayed corrective feedback will generally oscillate at a frequency related to the delay time.

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 6

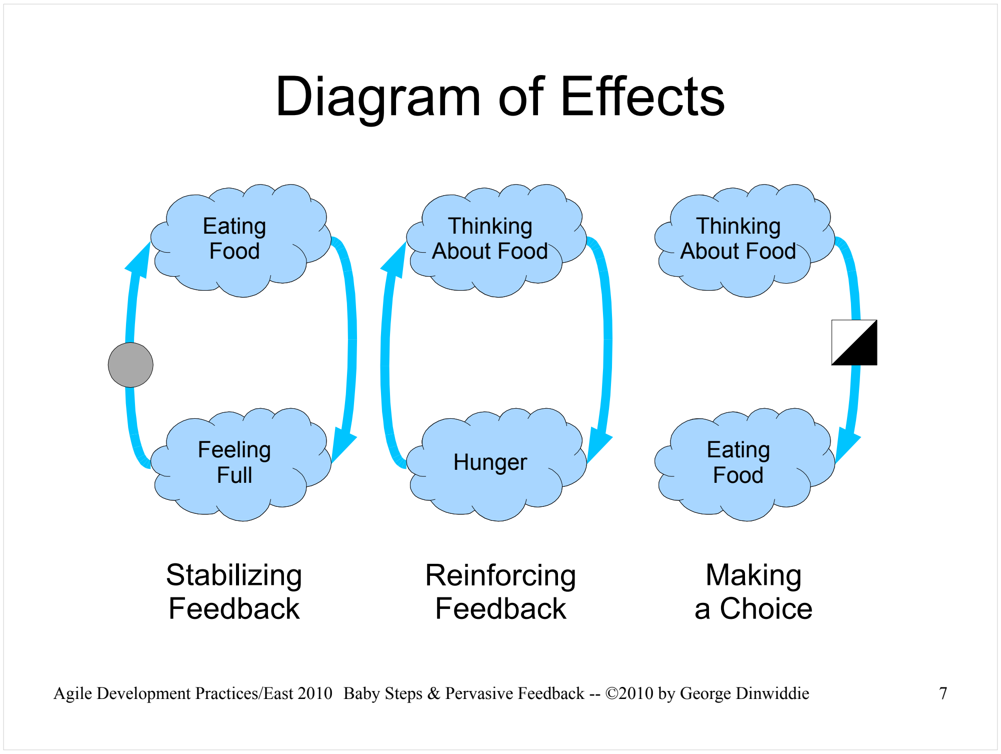
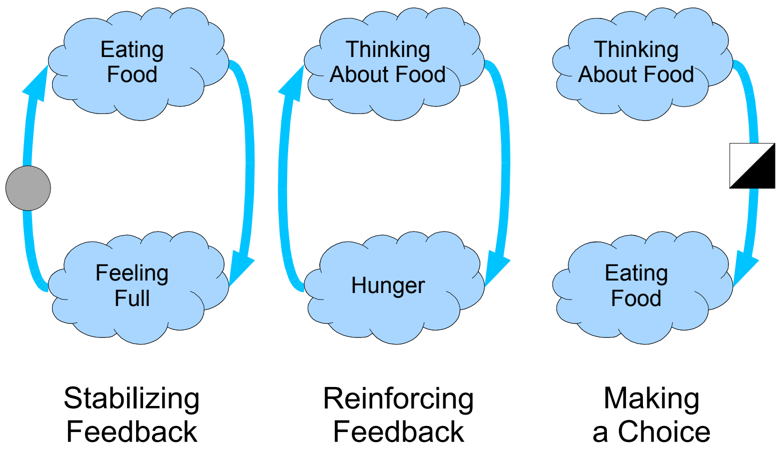


Diagram of Effects



Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 7

Another way of picturing feedback comes from Systems Thinking. This diagram is called a *Diagram of Effects*. This notation is used by Gerald M Weinberg in *Quality Software Management, Vol 1: Systems* Thinking. Peter Senge uses a different notation for the same concept in *The Fifth Discipline*.

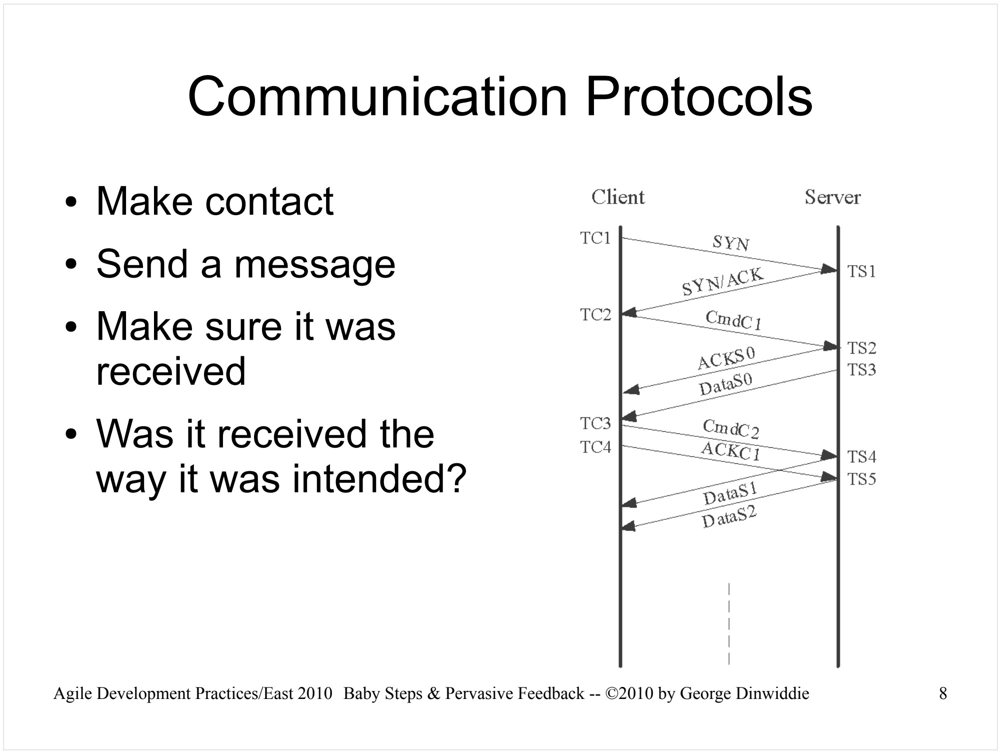
The clouds represent observable (and potentially measurable) quantities. A plain arrow indicate that an increase or decrease of one quantity influences a similar increase or decrease of the other. If there is a dot on the arrow, then the effect is the opposite—an increase of one influences a decrease of the other.

A **stabilizing** loop reaches a balance. The (A) *more food I eat*, then the (B) *fuller I feel*. The (B) *fuller I feel* then the (A) *less food I eat*.

A **reinforcing** loop continues until something else in the system intervenes. The (A) *more I think of food* then the (B) *hungrier I get*, and the (B) *hungrier I get* then the (A) *more I think of food*.

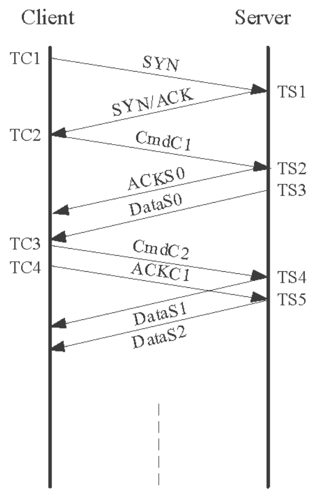
Sometimes, when there are human components in the system, there's not a direct causal relationship between one effect and another. Instead, we have a **choice** as to whether one effect increases or decreases the other.

See also:
http://www.developerdotstar.com/mag/articles/gray_diagram_of_effects.html
Quality Software Management, Vol 1: Systems Thinking by Gerald M. Weinberg
The Fifth Discipline: The Art & Practice of The Learning Organization by Peter M. Senge



Communication Protocols

- Make contact
- Send a message
- Make sure it was received
- Was it received the way it was intended?



Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 8

Conversations

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 9

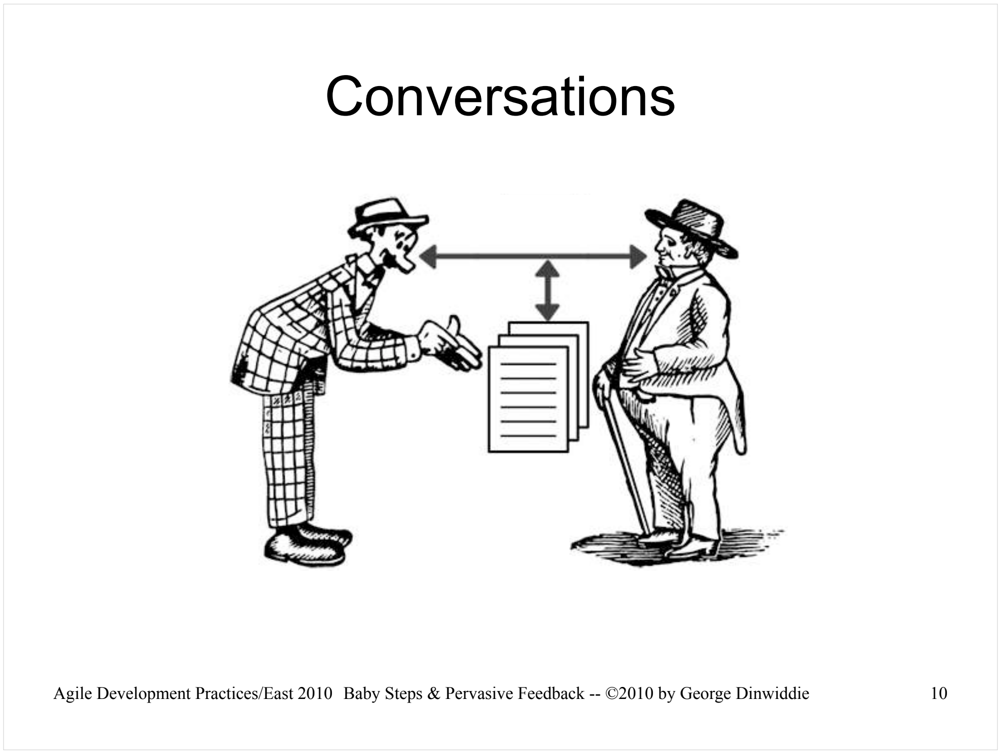
When we communicate via a document, we get no feedback as to whether the recipient understood us as we intended. We may not be sure if they even read it.

A conversation gives us opportunities to verify that the recipient understands what we meant. We can ask them to describe their understanding. They can ask clarifying questions.

And if we have that conversation face to face, then we get many non-verbal clues about their understanding of, and reaction to, what we say. We give non-verbal clues, too, both intentionally and unintentionally.

Documents are great at holding detailed information, but they're better used for aiding a conversation than for replacing it.

(Portions of these images courtesy of Briar Press www.briarpress.org)



Conversations

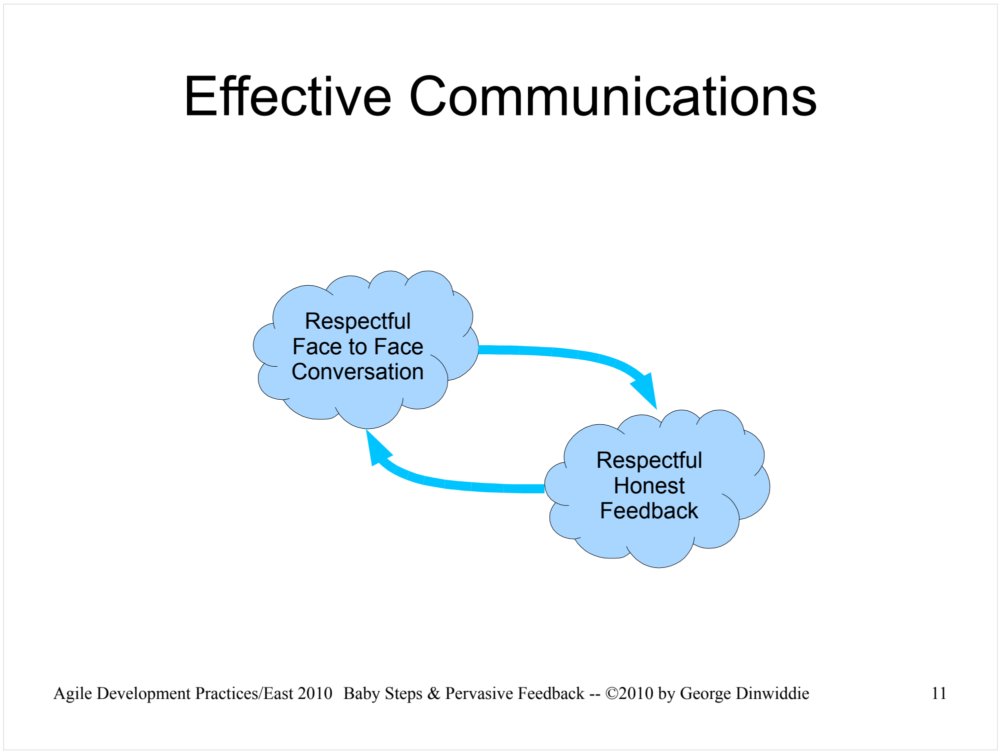
When we communicate via a document, we get no feedback as to whether the recipient understood us as we intended. We may not be sure if they even read it.

A conversation gives us opportunities to verify that the recipient understands what we meant. We can ask them to describe their understanding. They can ask clarifying questions.

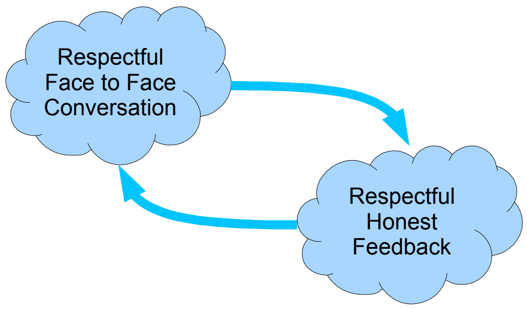
And if we have that conversation face to face, then we get many non-verbal clues about their understanding of, and reaction to, what we say. We give non-verbal clues, too, both intentionally and unintentionally.

Documents are great at holding detailed information, but they're better used for aiding a conversation than for replacing it.

(Portions of these images courtesy of Briar Press www.briarpress.org)



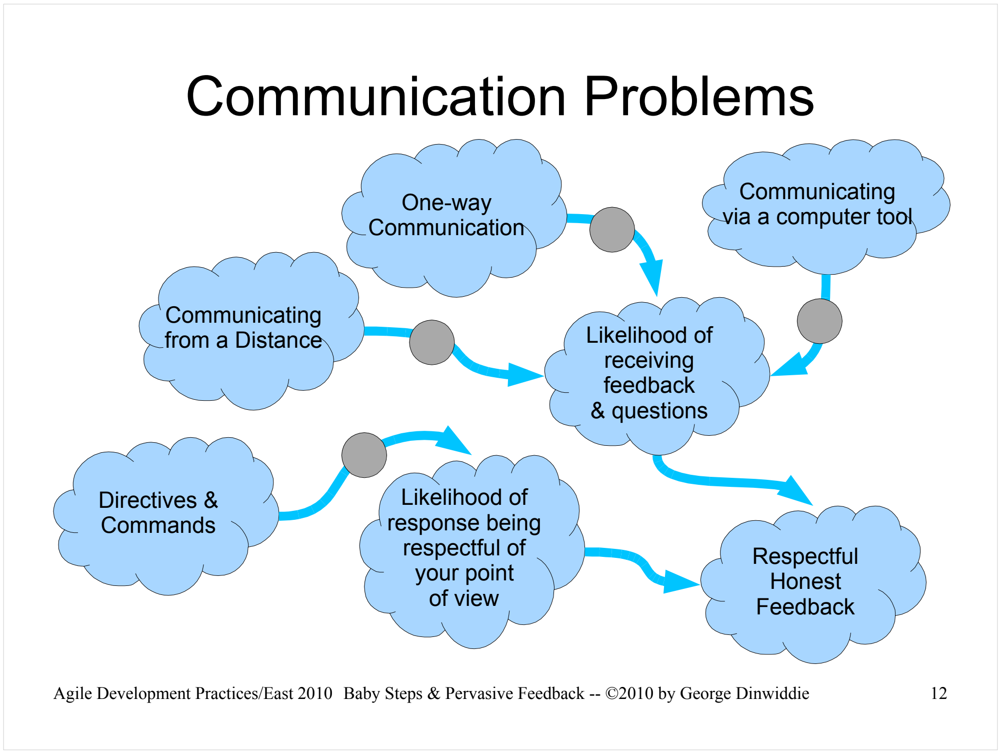
Effective Communications


Respectful Face to Face Conversation
Respectful Honest Feedback

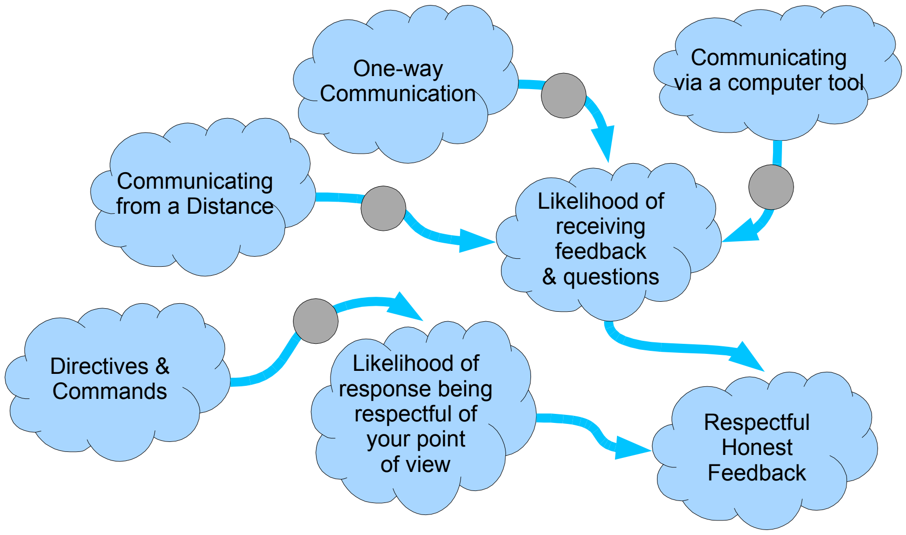
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 11

If the feedback you get in conversations isn't honest, it's not very helpful.

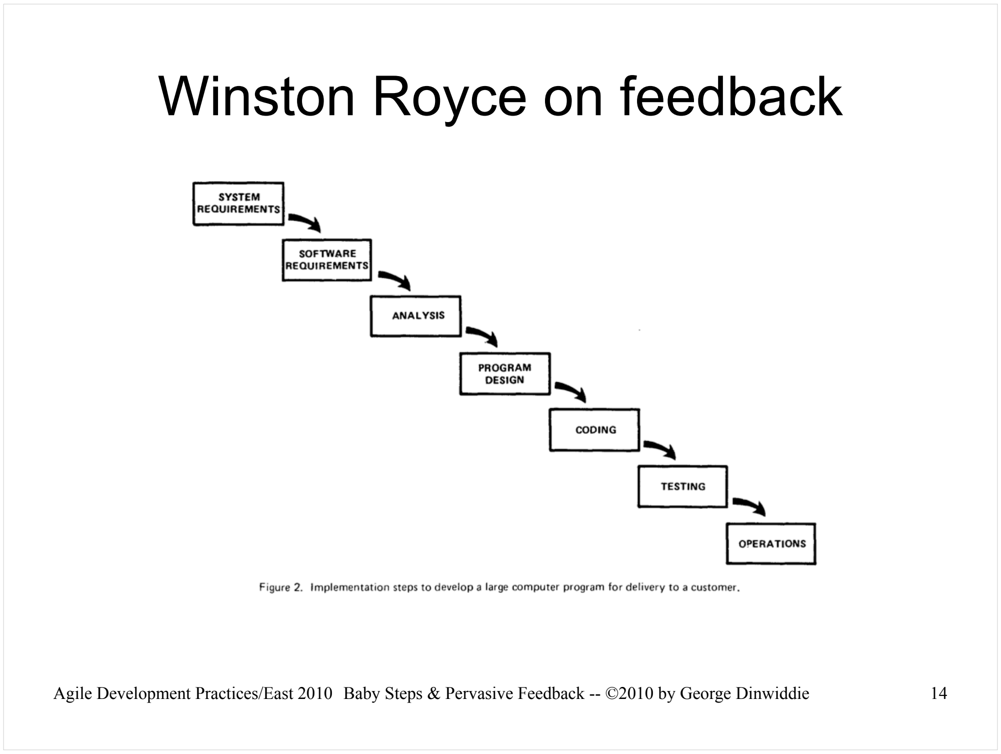
If the feedback you get isn't respectful, it's hard to listen to it.s



Communication Problems


One-way Communication
Communicating via a computer tool
Communicating from a Distance
Likelihood of receiving feedback & questions
Directives & Commands
Likelihood of response being respectful of your point of view
Respectful Honest Feedback

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 12



Winston Royce on feedback

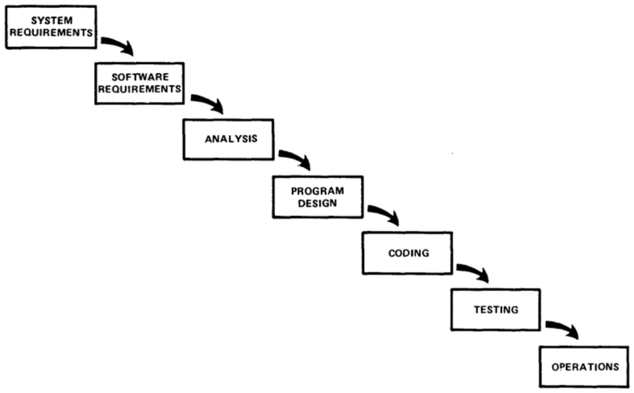


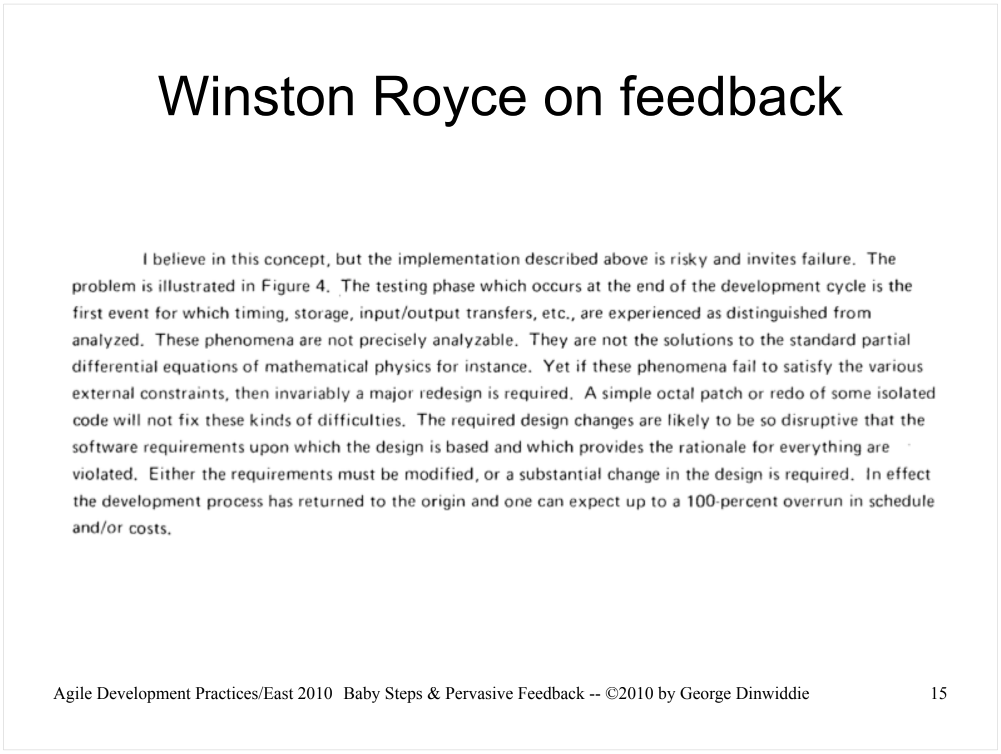
Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 14

The use of feedback in software development processes is nothing new, of course.

Winston Royce, in his classic paper, *Managing the Development of Large Software Systems*, noted that later phases often turned up information that invalidated work in earlier phases.

His approach suggested increasing the amount of up-front documentation, because the act of creating that documentation would require having a deep understanding of the requirements and design. Unfortunately, people have learned to create such documents *without* a deep understanding.



Winston Royce on feedback

I believe in this concept, but the implementation described above is risky and invites failure. The problem is illustrated in Figure 4. The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable. They are not the solutions to the standard partial differential equations of mathematical physics for instance. Yet if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required. A simple octal patch or redo of some isolated code will not fix these kinds of difficulties. The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and one can expect up to a 100-percent overrun in schedule and/or costs.

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 15

The use of feedback in software development processes is nothing new, of course.

Winston Royce, in his classic paper, *Managing the Development of Large Software Systems*, noted that later phases often turned up information that invalidated work in earlier phases.

His approach suggested increasing the amount of up-front documentation, because the act of creating that documentation would require having a deep understanding of the requirements and design. Unfortunately, people have learned to create such documents *without* a deep understanding.

Winston Royce on feedback

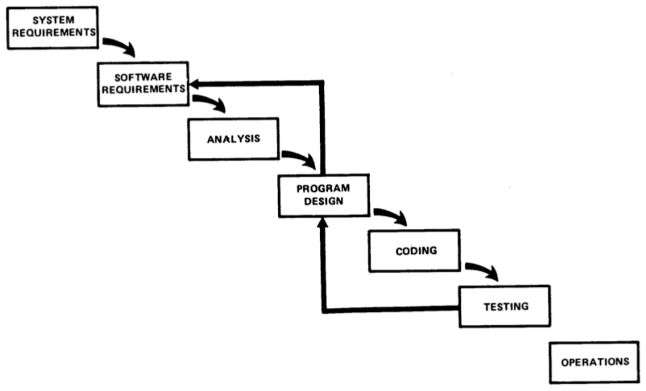


Figure 4. Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 16

The use of feedback in software development processes is nothing new, of course.

Winston Royce, in his classic paper, *Managing the Development of Large Software Systems*, noted that later phases often turned up information that invalidated work in earlier phases.

His approach suggested increasing the amount of up-front documentation, because the act of creating that documentation would require having a deep understanding of the requirements and design. Unfortunately, people have learned to create such documents *without* a deep understanding.

Winston Royce on iterations

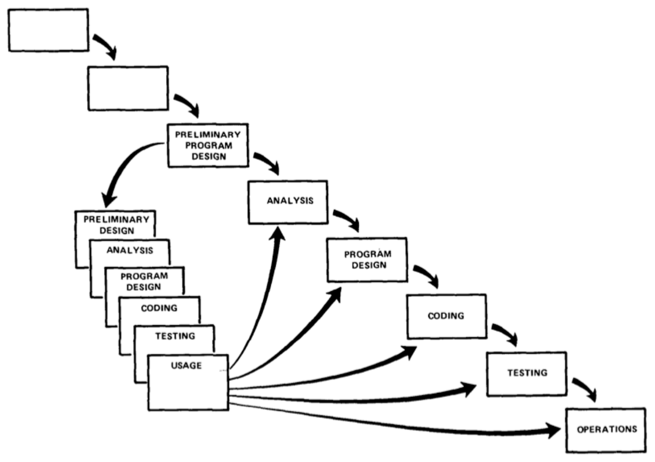


Figure 7. Step 3: Attempt to do the job twice – the first result provides an early simulation of the final product.

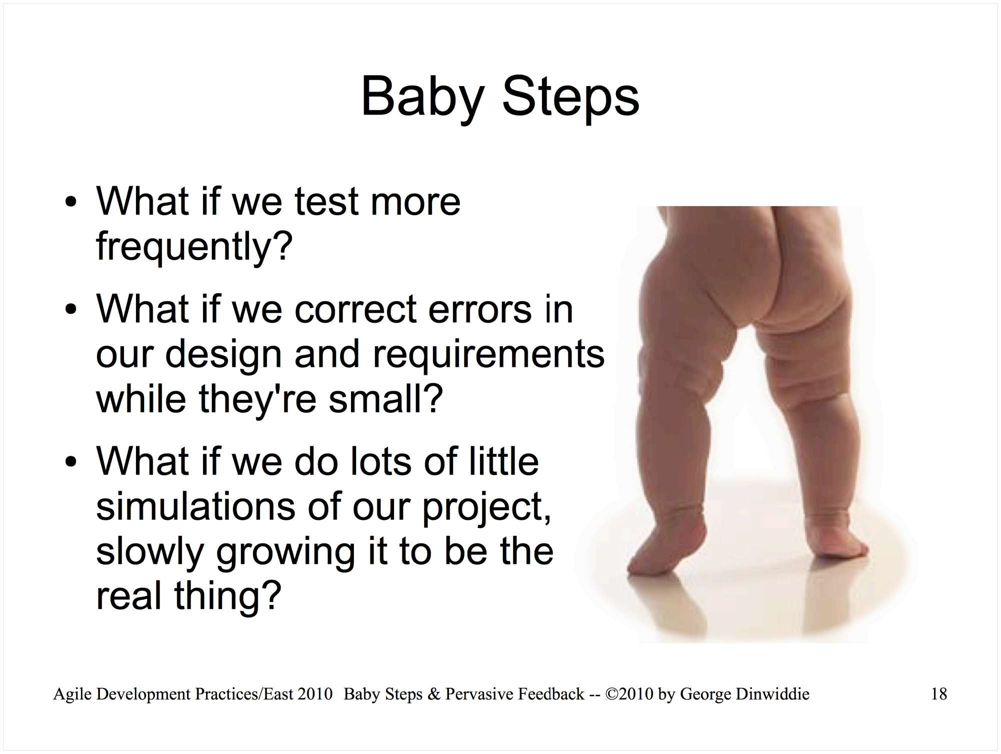
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 17

Royce suggested doing a “simulation” or prototype to learn the essential lessons to guide the design and development. (Brooks said, “Plan to throw the first one away.”)

In Agile, we instead shorten the feedback cycles using real, executable code and tests. We expand this from twice through the system to going through it all the time.

He noted that, “A very special kind of broad competence is required on the part of the personnel involved. They must have an intuitive feel for analysis, coding, and program design.”

This is true for Agile development, and many practices are designed to make this easier for the development team.



Baby Steps

- What if we test more frequently?
- What if we correct errors in our design and requirements while they're small?
- What if we do lots of little simulations of our project, slowly growing it to be the real thing?

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 18

Many of the Agile practices that enable using many iterations come down to the practice of using baby steps.

Successful Steering

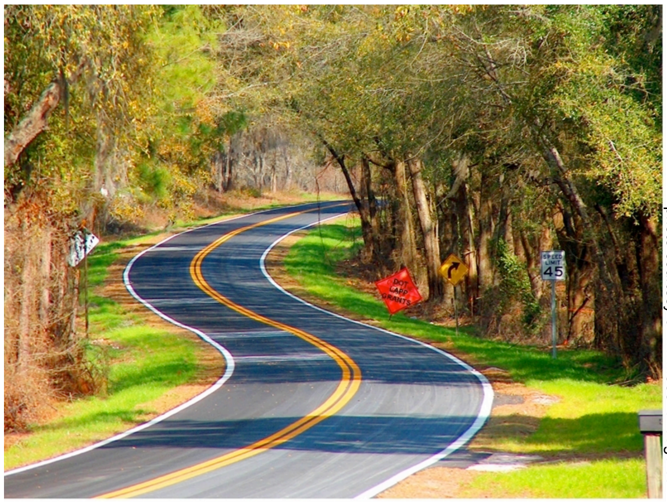


photo courtesy of www.seefloridago.com

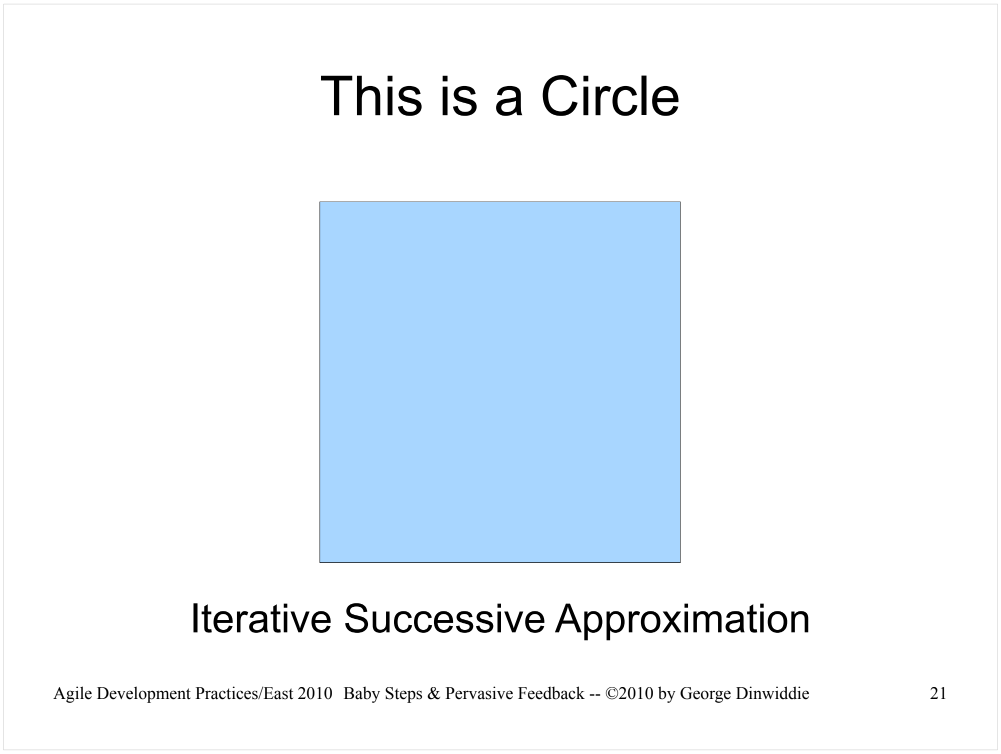
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 19

When you're driving on a road like this, how far do you go before you adjust your direction?

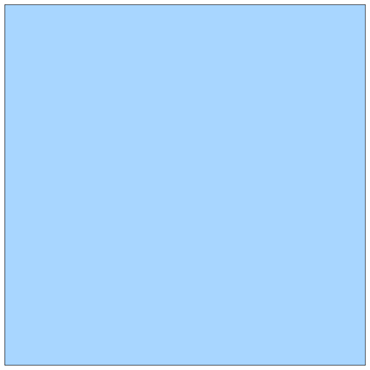
Think what happens if you only adjust the steering wheel and pedals only every 30 seconds while driving.

-

Your project is like this road, and if you don't pay attention and steer frequently, you'll end up in the woods.



This is a Circle

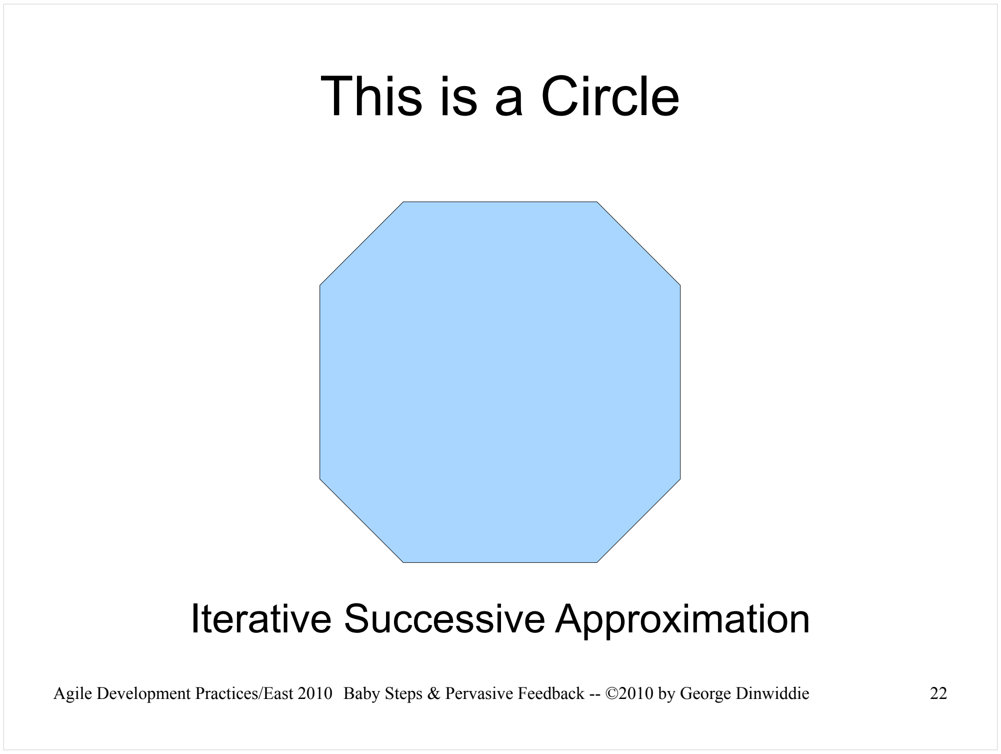


Iterative Successive Approximation

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 21

Another way of looking at feedback—in terms of something produced rather than in terms of a process—is **iterative successive approximation**.

We start with a very rough approximation of our goal.



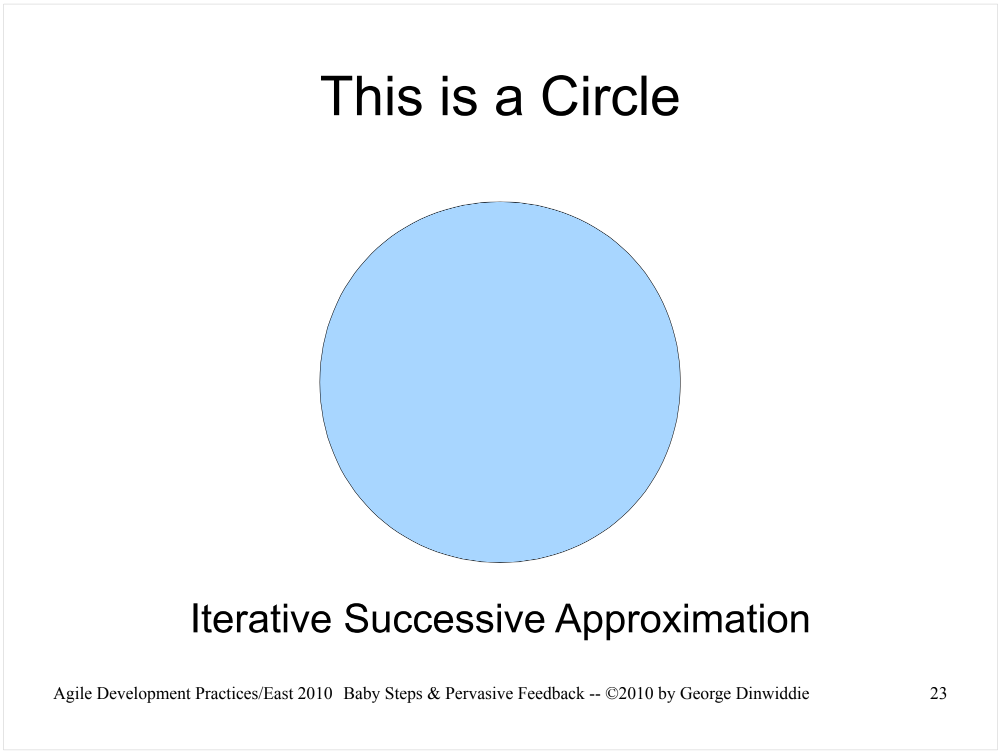
This is a Circle

Iterative Successive Approximation

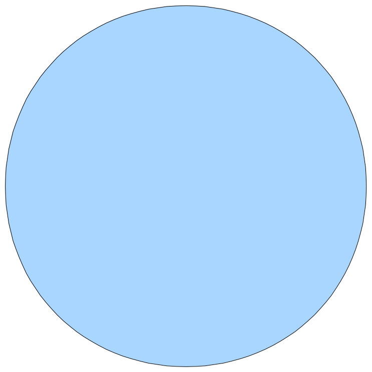
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 22

Another way of looking at feedback—in terms of something produced rather than in terms of a process—is **iterative successive approximation**.

We start with a very rough approximation of our goal. Then, noticing how our approximation differs from what we desire, we make small improvements.



This is a Circle



Iterative Successive Approximation

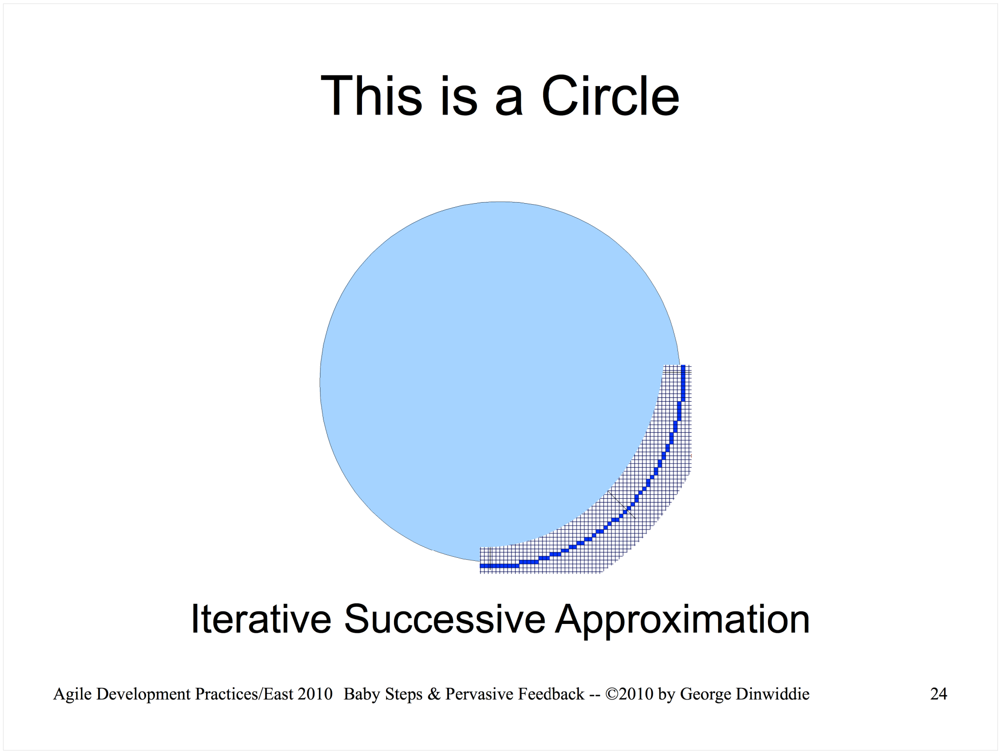
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 23

Another way of looking at feedback—in terms of something produced rather than in terms of a process—is **iterative successive approximation**.

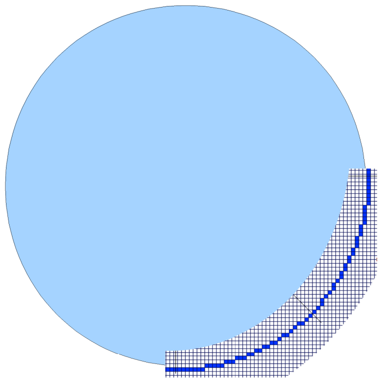
We start with a very rough approximation of our goal.

Then, noticing how our approximation differs from what we desire, we make small improvements.

Bit by bit, we get closer to our ideal, but we have the option to stop whenever we feel we're “close enough.”



This is a Circle



Iterative Successive Approximation

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 24

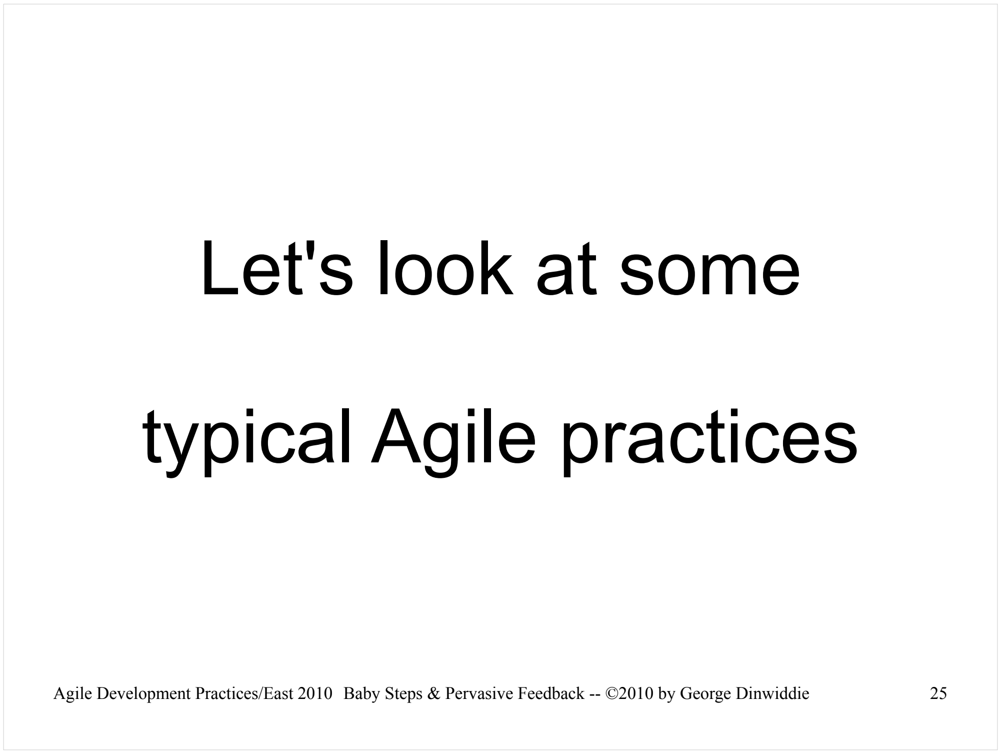
Another way of looking at feedback—in terms of something produced rather than in terms of a process—is **iterative successive approximation**.

We start with a very rough approximation of our goal.

Then, noticing how our approximation differs from what we desire, we make small improvements.

Bit by bit, we get closer to our ideal, but we have the option to stop whenever we feel we're “close enough.”

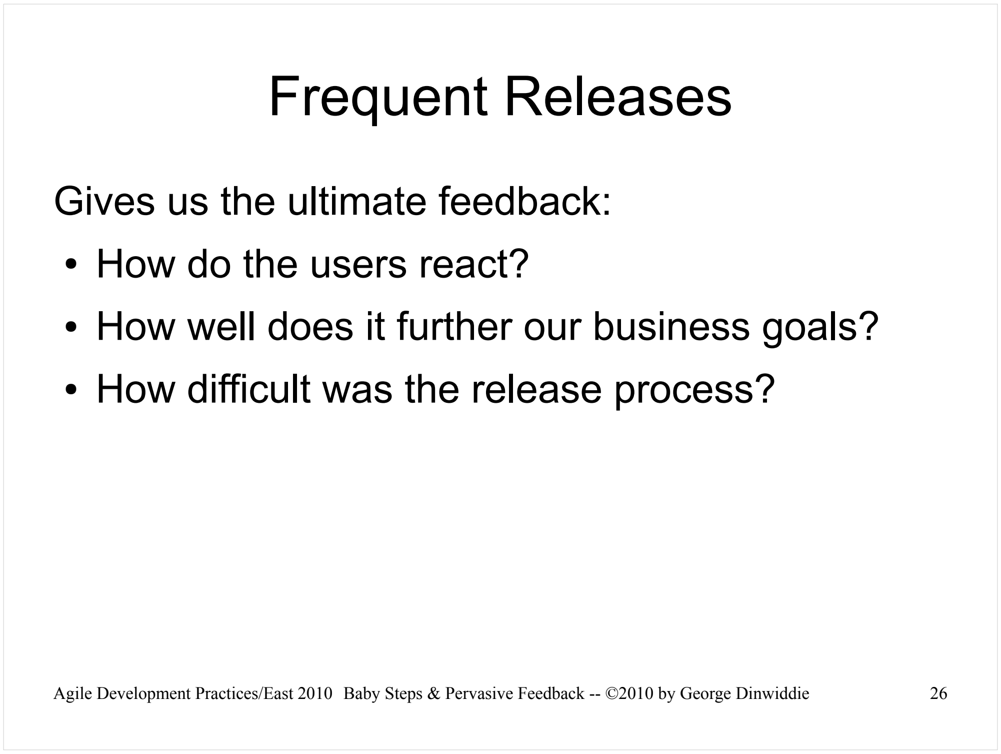
Note that the final circle is not *<u>really</u>* a smooth circle, but it's as close as we can represent with this resolution of square pixels. The smaller the pixels, the better the representation.



Let's look at some

typical Agile practices

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 25



Frequent Releases

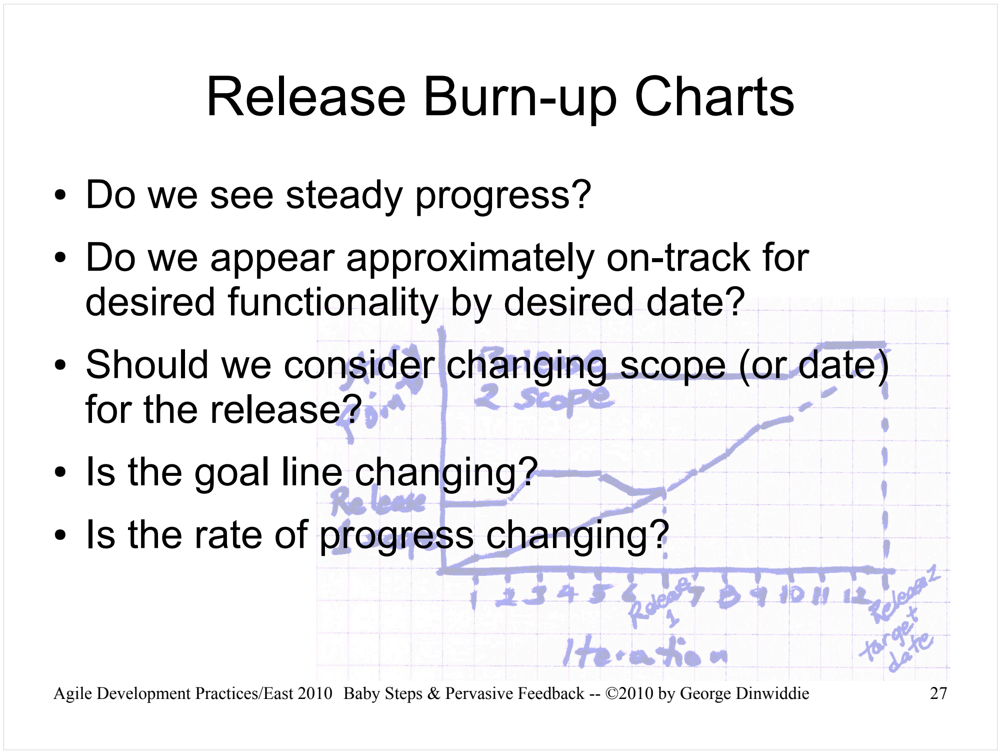
Gives us the ultimate feedback:

- How do the users react?
- How well does it further our business goals?
- How difficult was the release process?

This is the most important feedback, but is generally a bit late for learning some of our lessons.

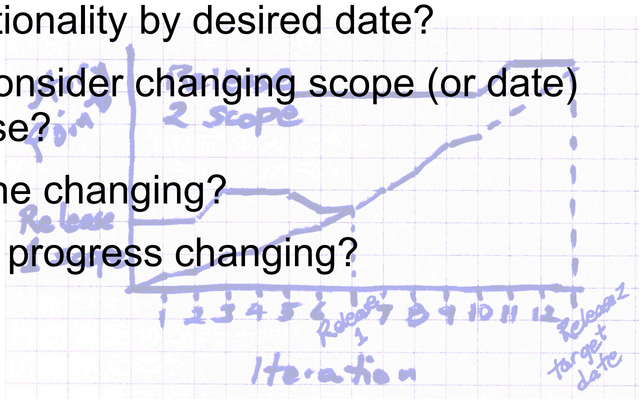
How frequently can you release?

What holds you back?



Release Burn-up Charts

- Do we see steady progress?
- Do we appear approximately on-track for desired functionality by desired date?
- Should we consider changing scope (or date) for the release?
- Is the goal line changing?
- Is the rate of progress changing?

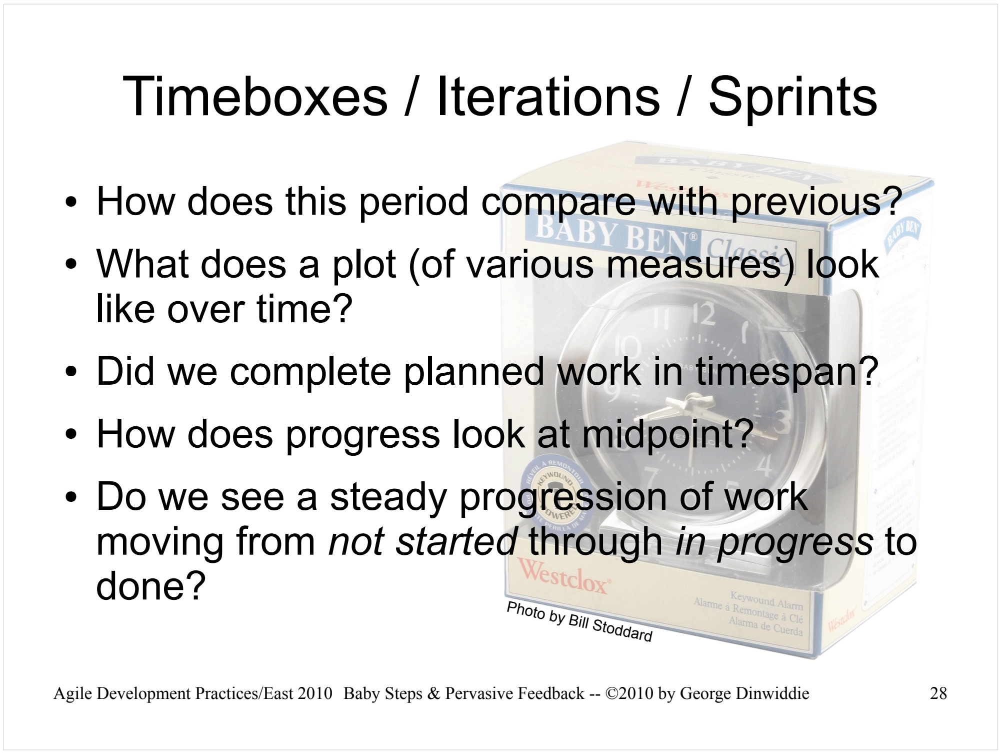


Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 27

See, also:

http://idiacomputing.com/pub/BetterSoftware-BurnCharts.pdf

http://blog.gdinwiddie.com/2010/04/22/projecting-into-the-future/



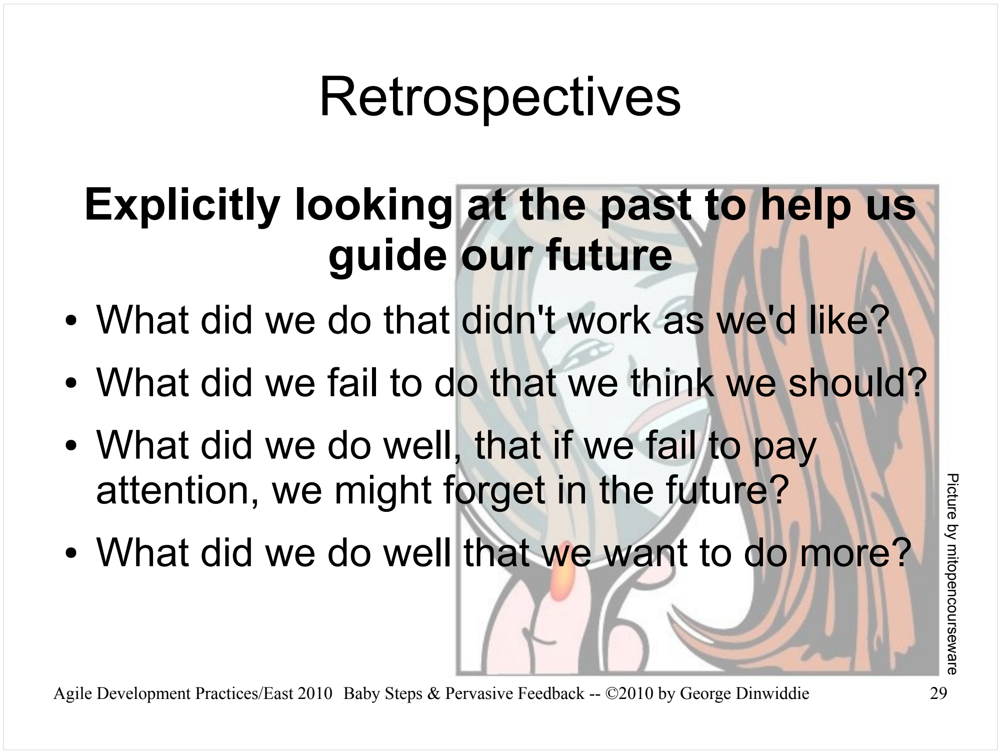
Timeboxes / Iterations / Sprints

- How does this period compare with previous?
- What does a plot (of various measures) look like over time?
- Did we complete planned work in timespan?
- How does progress look at midpoint?
- Do we see a steady progression of work moving from *not started* through *in progress* to done?

Photo by Bill Stoddard

BABY BEN Classic
Westclox
Keywound Alarm
Alarme à Remontage à Clé
Alarma de Cuerda

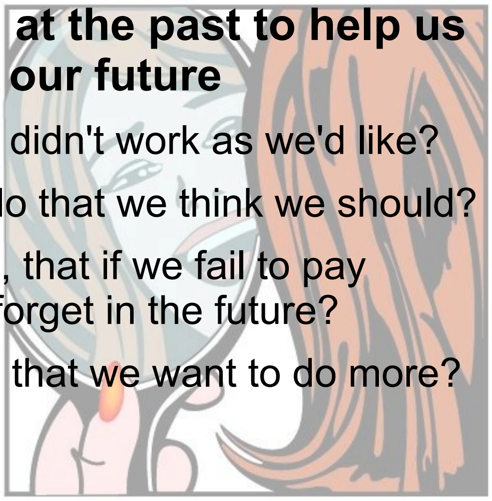
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 28



Retrospectives

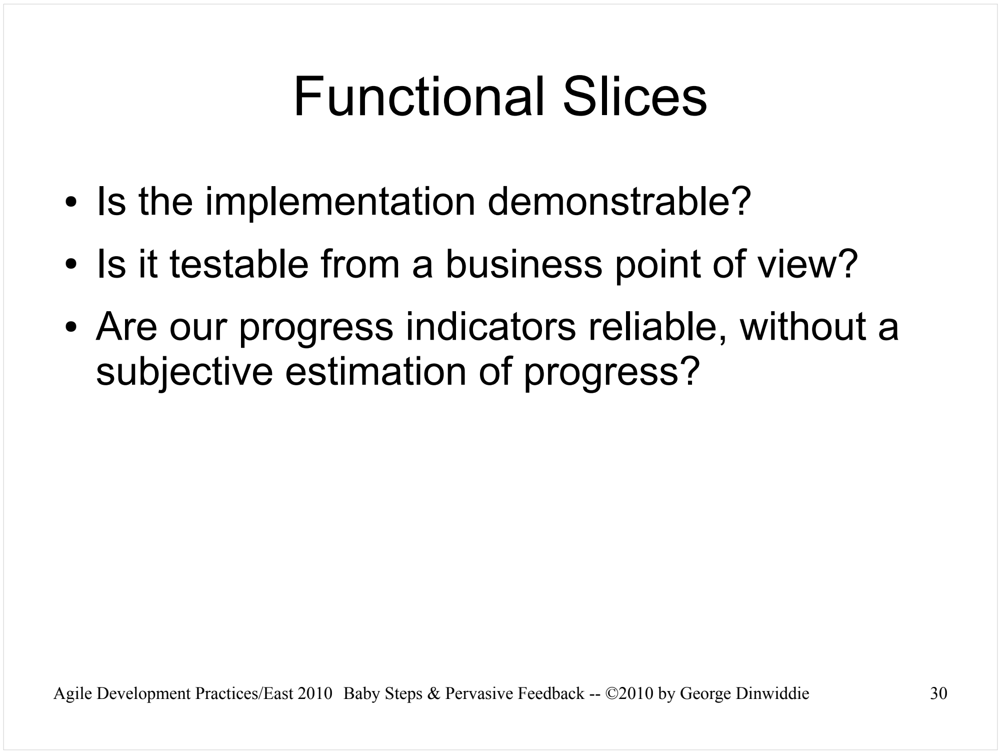
Explicitly looking at the past to help us guide our future

- What did we do that didn't work as we'd like?
- What did we fail to do that we think we should?
- What did we do well, that if we fail to pay attention, we might forget in the future?
- What did we do well that we want to do more?



Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 29

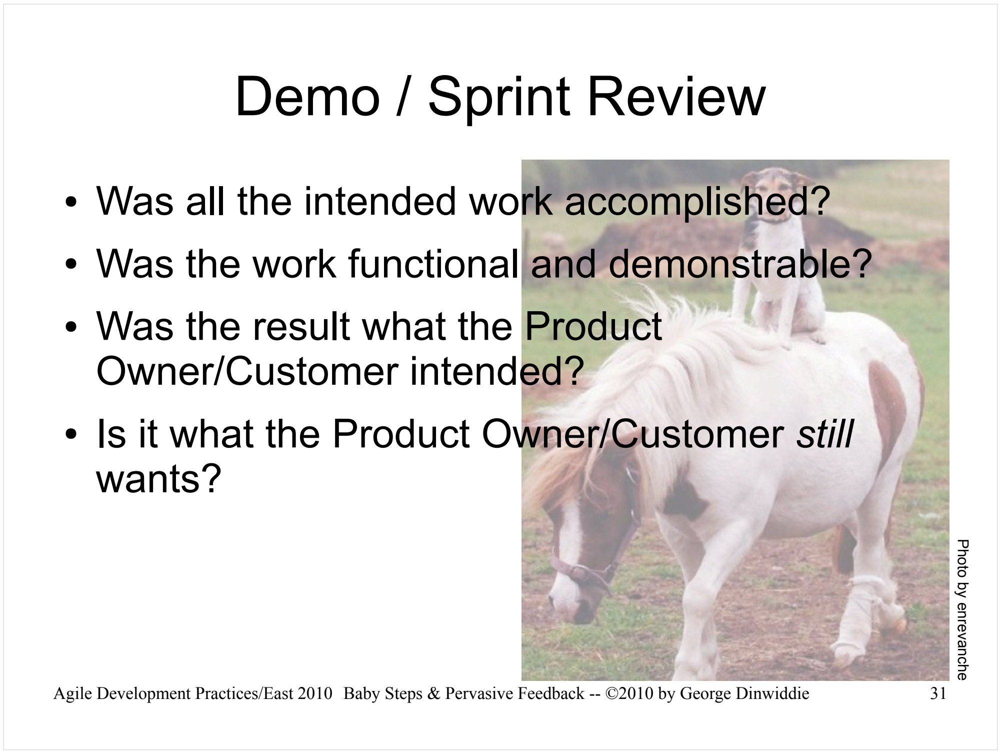
Learning from experience – made explicit.



Functional Slices

- Is the implementation demonstrable?
- Is it testable from a business point of view?
- Are our progress indicators reliable, without a subjective estimation of progress?

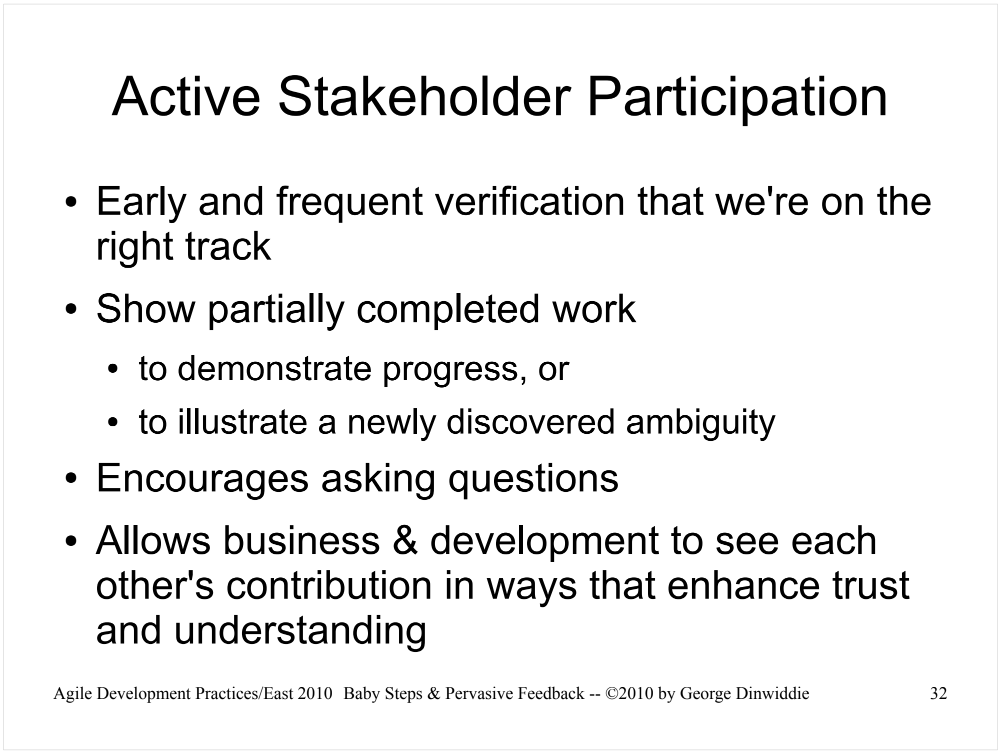
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 30



Demo / Sprint Review

- Was all the intended work accomplished?
- Was the work functional and demonstrable?
- Was the result what the Product Owner/Customer intended?
- Is it what the Product Owner/Customer *still* wants?

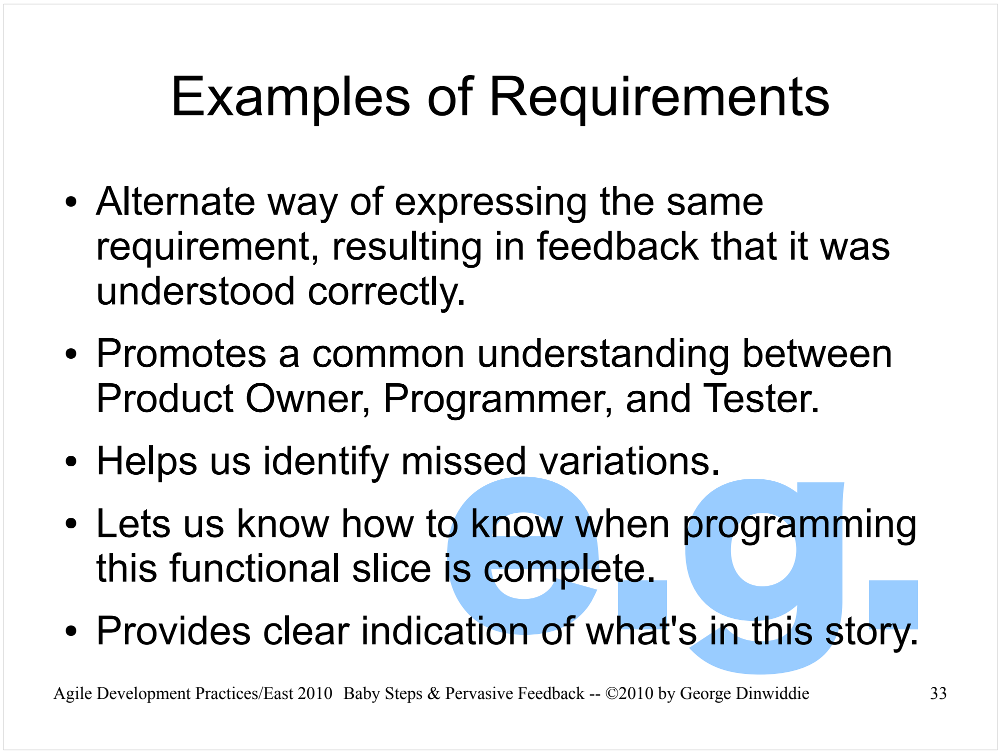
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 31



Active Stakeholder Participation

- Early and frequent verification that we're on the right track
- Show partially completed work
  - to demonstrate progress, or
  - to illustrate a newly discovered ambiguity
- Encourages asking questions
- Allows business & development to see each other's contribution in ways that enhance trust and understanding

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 32

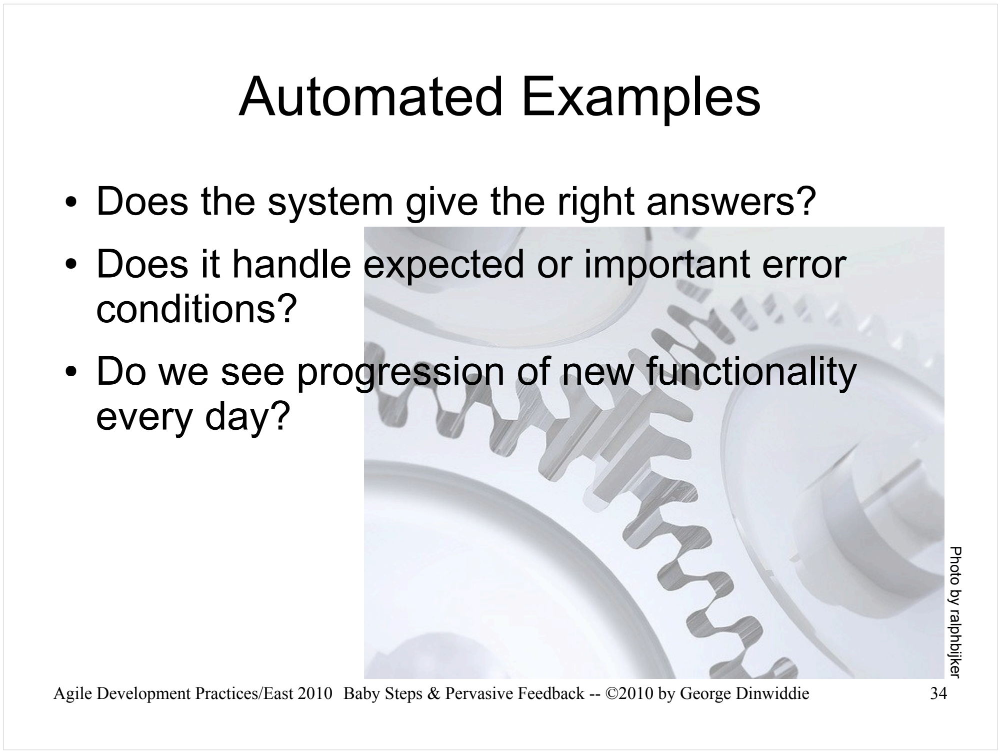


Examples of Requirements

- Alternate way of expressing the same requirement, resulting in feedback that it was understood correctly.
- Promotes a common understanding between Product Owner, Programmer, and Tester.
- Helps us identify missed variations.
- Lets us know how to know when programming this functional slice is complete.
- Provides clear indication of what's in this story.

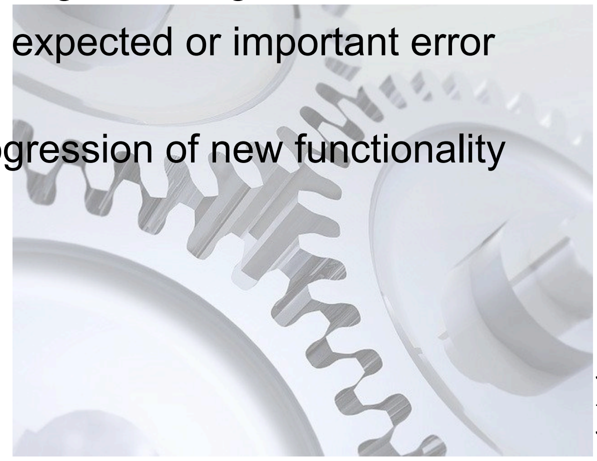
e.g.

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 33

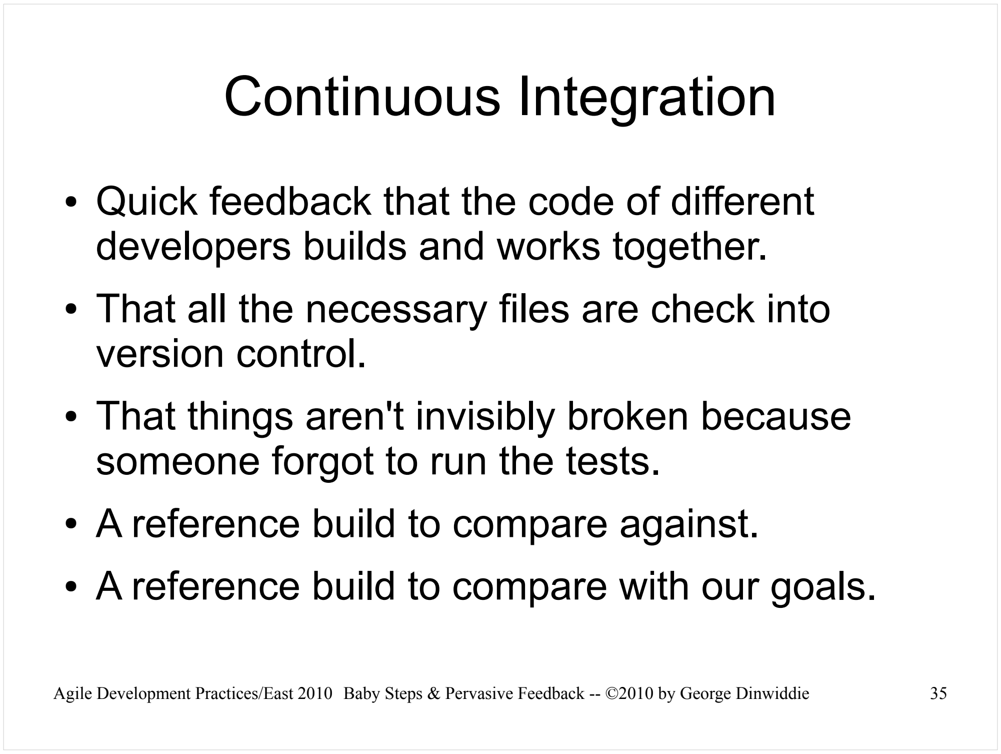


Automated Examples

- Does the system give the right answers?
- Does it handle expected or important error conditions?
- Do we see progression of new functionality every day?



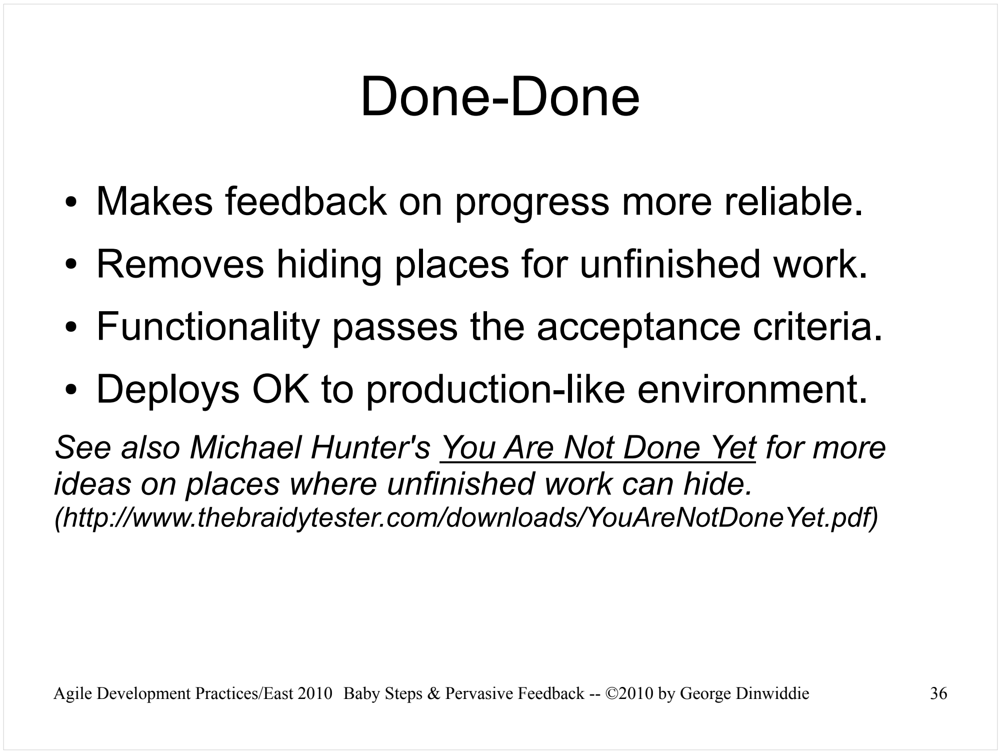
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 34



Continuous Integration

- Quick feedback that the code of different developers builds and works together.
- That all the necessary files are check into version control.
- That things aren't invisibly broken because someone forgot to run the tests.
- A reference build to compare against.
- A reference build to compare with our goals.

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 35

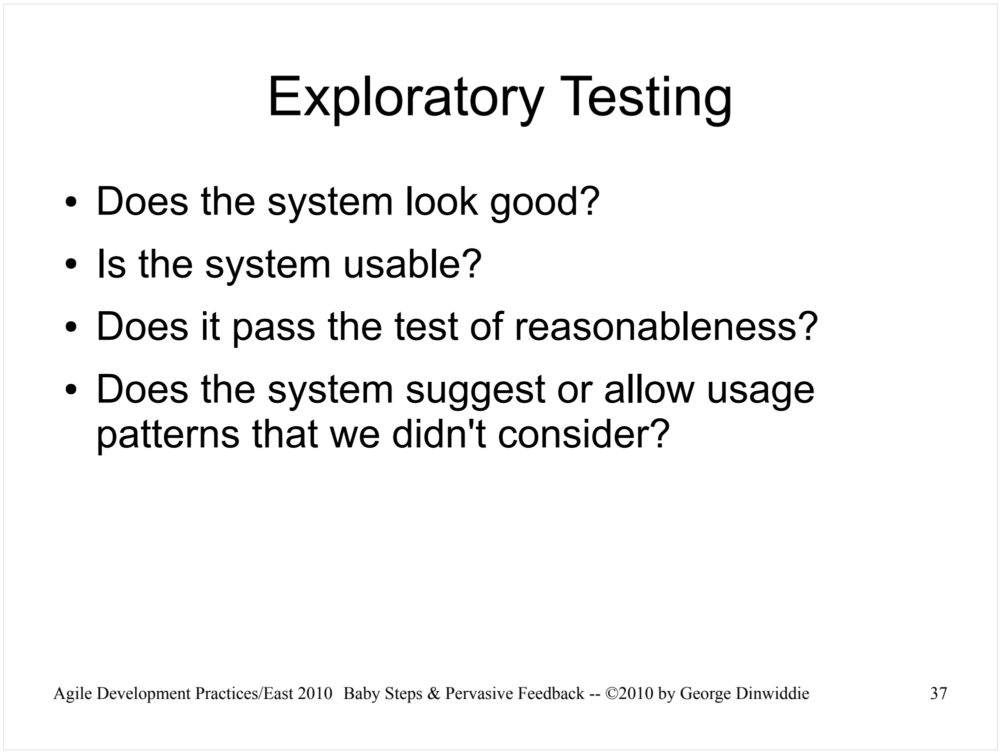


Done-Done

- Makes feedback on progress more reliable.
- Removes hiding places for unfinished work.
- Functionality passes the acceptance criteria.
- Deploys OK to production-like environment.

See also Michael Hunter's You Are Not Done Yet for more ideas on places where unfinished work can hide.
(http://www.thebraidytester.com/downloads/YouAreNotDoneYet.pdf)

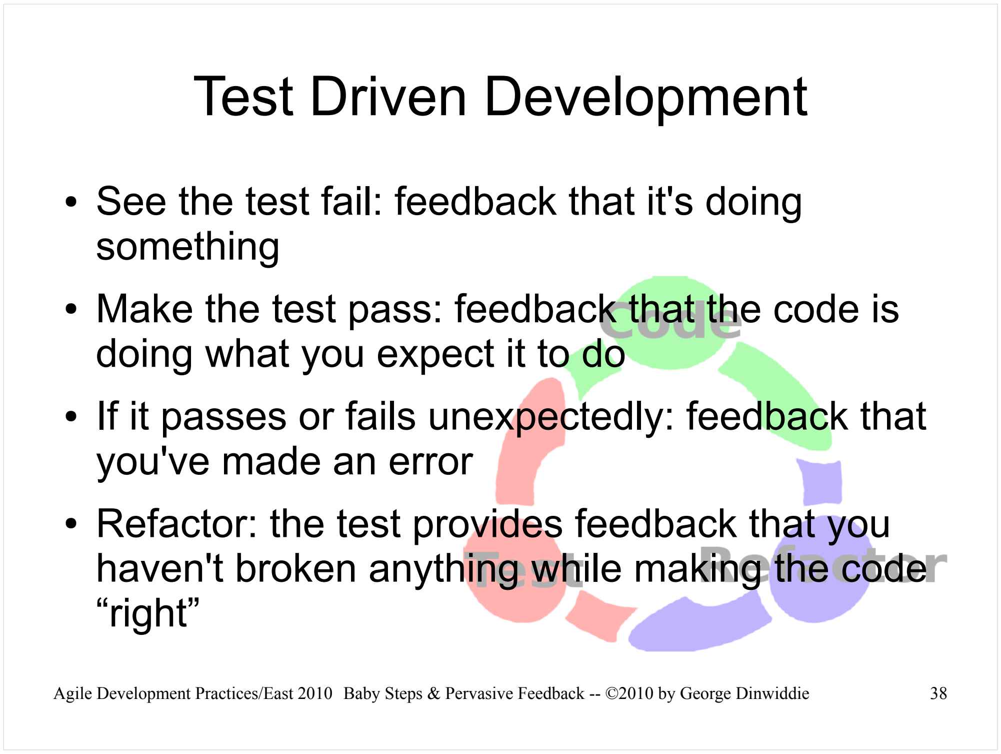
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 36



Exploratory Testing

- Does the system look good?
- Is the system usable?
- Does it pass the test of reasonableness?
- Does the system suggest or allow usage patterns that we didn't consider?

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 37



Test Driven Development

- See the test fail: feedback that it's doing something
- Make the test pass: feedback that the code is doing what you expect it to do
- If it passes or fails unexpectedly: feedback that you've made an error
- Refactor: the test provides feedback that you haven't broken anything while making the code “right”

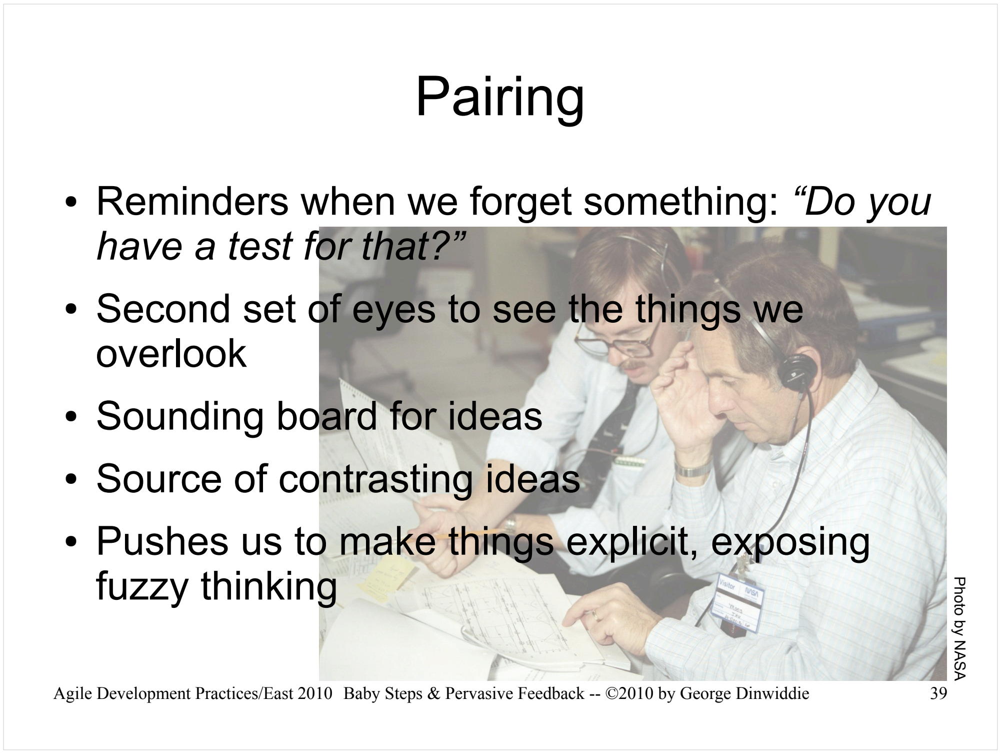
Code
Test
Refactor

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 38

Work in as small an increment as you can: one line of test, one line of code, or maybe less.

•Make it fail
•Make it pass
•Make it right

The test tests the code and the code tests the test.

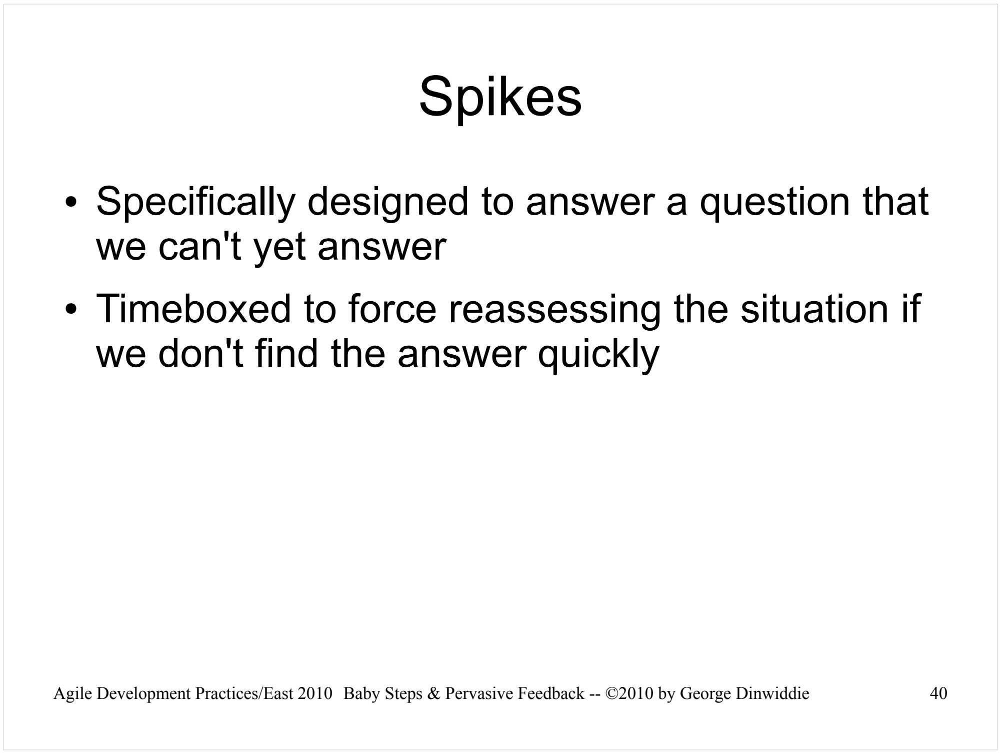


Pairing

- Reminders when we forget something: *"Do you have a test for that?"*
- Second set of eyes to see the things we overlook
- Sounding board for ideas
- Source of contrasting ideas
- Pushes us to make things explicit, exposing fuzzy thinking

Photo by NASA

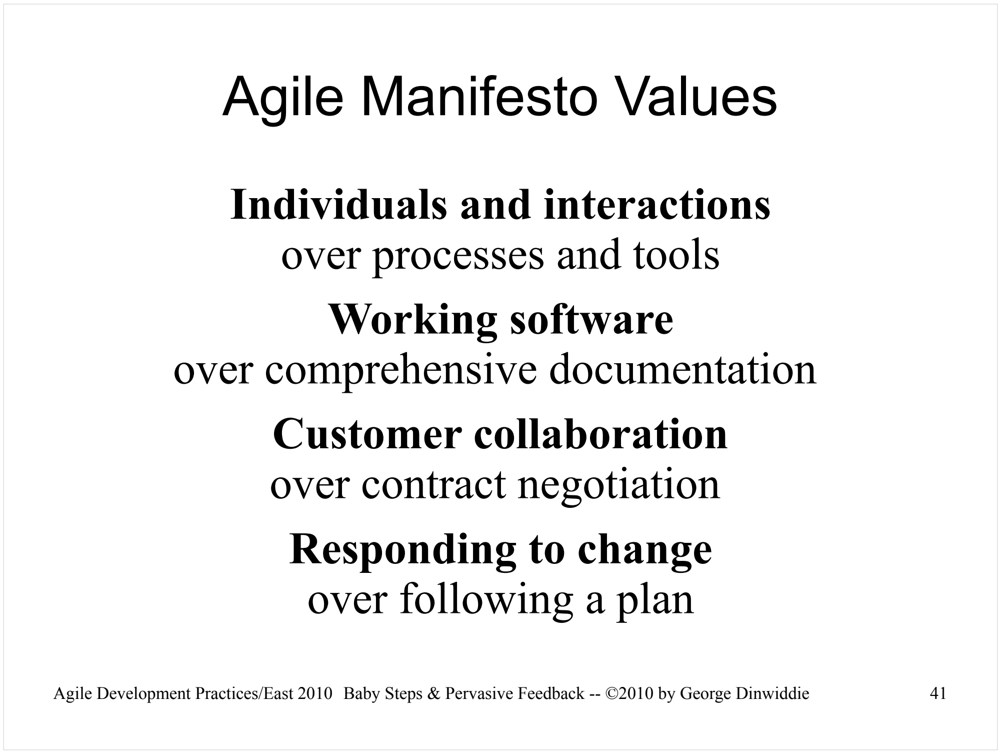
Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 39



Spikes

- Specifically designed to answer a question that we can't yet answer
- Timeboxed to force reassessing the situation if we don't find the answer quickly

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 40



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

Agile Development Practices/East 2010 Baby Steps & Pervasive Feedback -- ©2010 by George Dinwiddie 41

Thanks to Bill Caputo for the reminder that these practices, alone, won't make you Agile. But practiced in the spirit of these values, they can make a huge difference.