Business Value of Agile Methods

Using Return on Investment

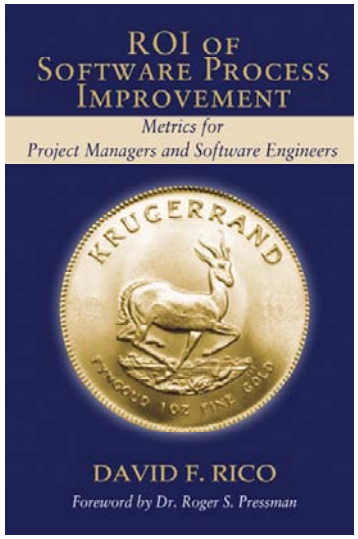
Dr. David F. Rico, PMP, CSM

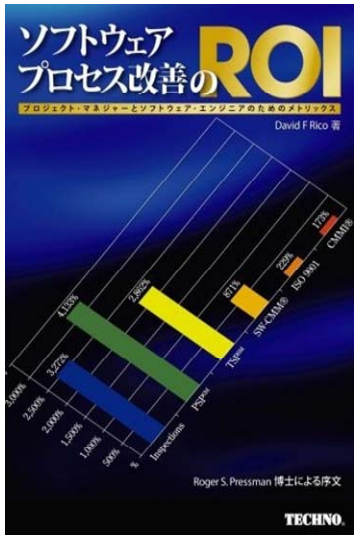
Agenda

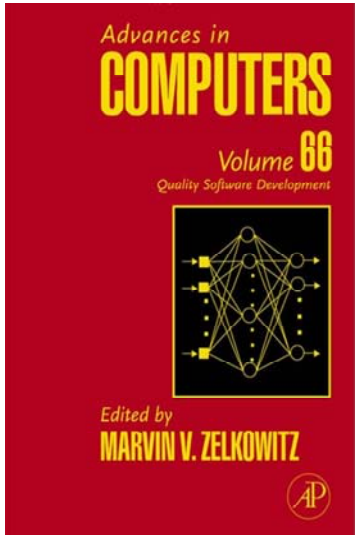
- Introduction
- Sources of Business Value
- Planning for Business Value
- Surveys of Business Value
- Measures of Business Value
- Models of Business Value
- Estimation of Business Value
- Comparison of Business Value
- Summary of Business Value

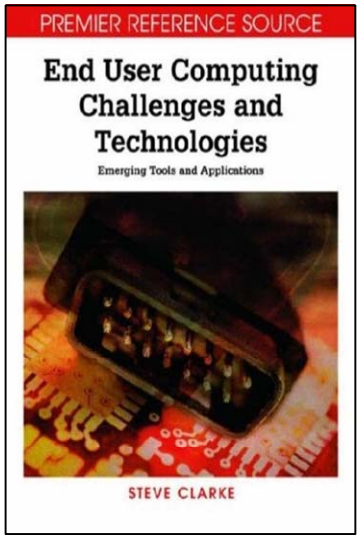
Author

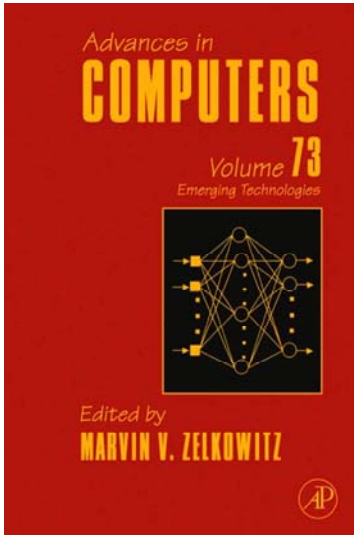
- DoD contractor with 25+ years of IT experience
- B.S. Comp. Sci., M.S. Soft. Eng., D.M. Info. Tech.
- Large NASA & DoD programs (U.S., Japan, Europe)











* Published five textbooks and over 15 articles on various topics in return on investment, information technology, agile methods, etc.

Purpose

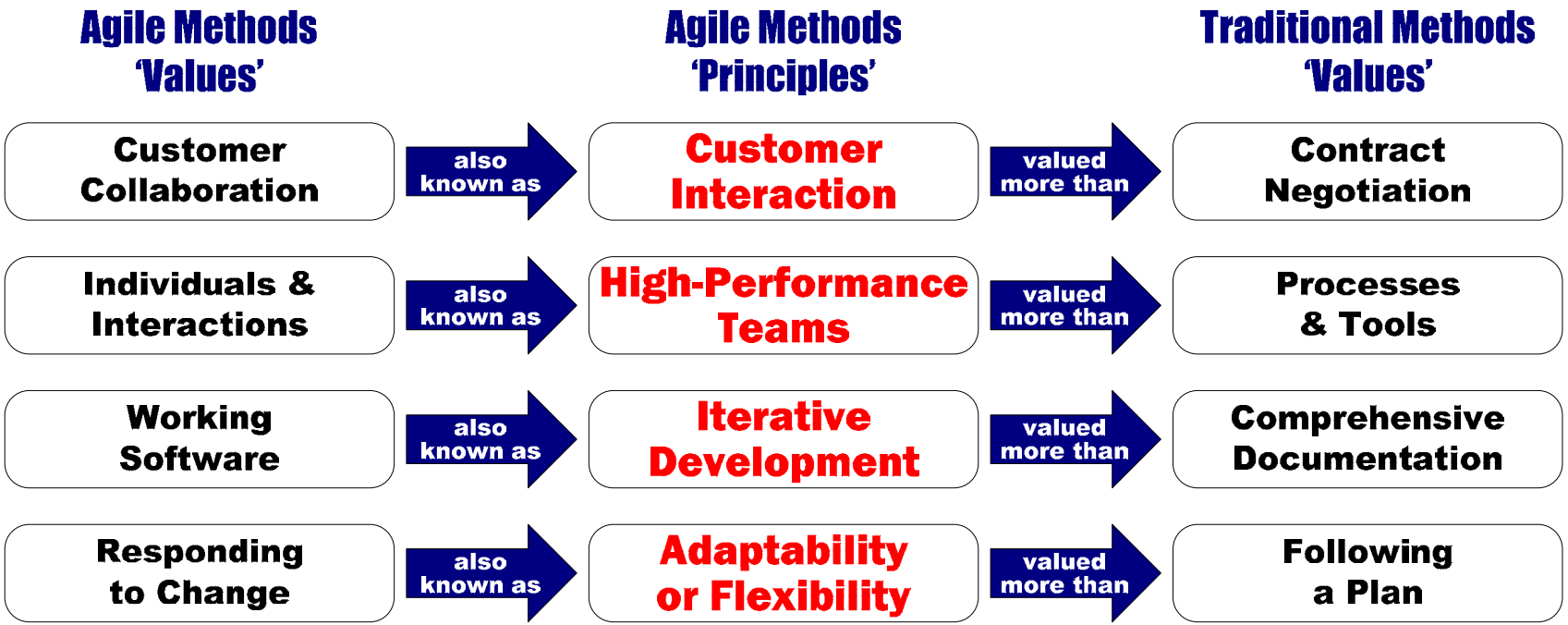
- Provide an overview of the business value of Agile Methods using return on investment:
  - Business value *is an approach for estimating the tangible and intangible worth of organizational assets*
  - Business value *is an appraisal of intellectual assets such as knowledge, experience, and skills*
  - Business value *is a technique for determining the complete worth of an investment to an enterprise*
  - Business value *is a method of determining the health and well-being of a firm in the long-run*
  - Business value *includes employee, customer, supplier, alliance, management, and societal value*

What is Agility?

- A-gil-i-ty (ə-'ji-lə-tē) Quickness, lightness, and ease of movement; nimbleness
  - Agility *is the ability to create and respond to change in order to profit in a turbulent business environment*
  - Agility *is reprioritizing for maneuverability because of shifting requirements, technology, and knowledge*
  - Agility *is a very fast response to changes in customer requirements through intensive customer interaction*
  - Agility *is the use of adaptability and evolutionary delivery to promote rapid customer responsiveness*
  - Agility *is a better way of developing products using collaboration, teamwork, iterations, and flexibility*

What are Agile Methods?

- ‘Adaptable’ software development methodologies
- ‘Human-centric’ method for creating business value
- ‘Alternative’ to large document-based methodologies



Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved September 3, 2008, from http://www.agilemanifesto.org

Essence of Agile Methods

- High degree of customer & developer interaction
- Highly-skilled teams producing frequent iterations
- Right-sized, just-enough, and just-in-time process

Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley.

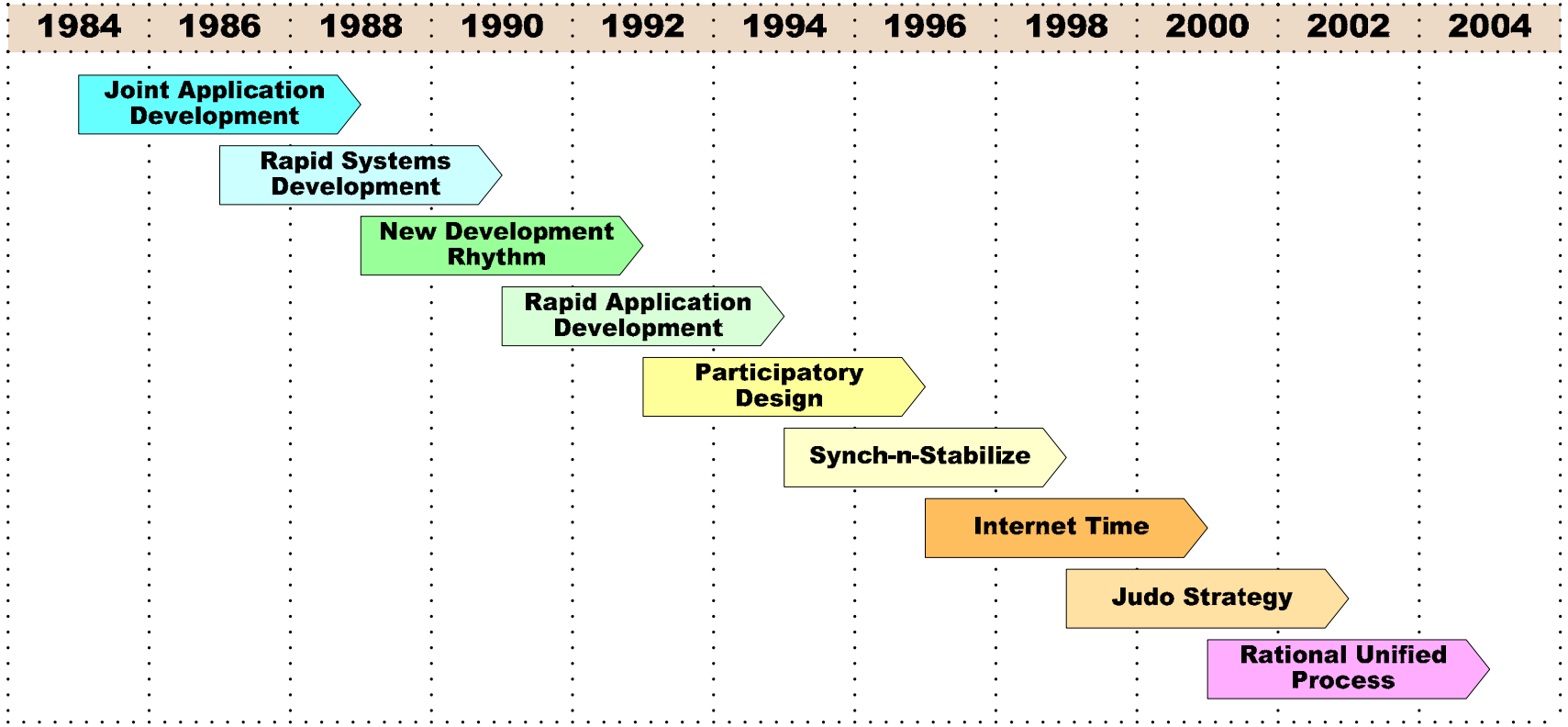
Why use Agile Methods?

- Adaptability to changing market/customer needs
- Better cost efficiencies and fastest time-to-market
- Improved quality, satisfaction, and project success

Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved September 3, 2008, from http://www.agilemanifesto.org

Antecedents of Agile Methods

- Rooted in management evolution from early 1900s
- Evolved from software methods from 1950s/1960s
- Spinoffs of NPD/RAD approaches from the 1980s



Rico, D. F., Sayani, H. H., & Field, R. F. (2008). History of computers, electronic commerce, and agile methods. In M. V. Zelkowitz (Ed.), *Advances in computers: Emerging technologies, Vol. 73*. San Diego, CA: Elsevier.

Agenda

- Introduction
- ☞ Sources of Business Value
- Planning for Business Value
- Surveys of Business Value
- Measures of Business Value
- Models of Business Value
- Estimation of Business Value
- Comparison of Business Value
- Summary of Business Value

Types of Agile Methods

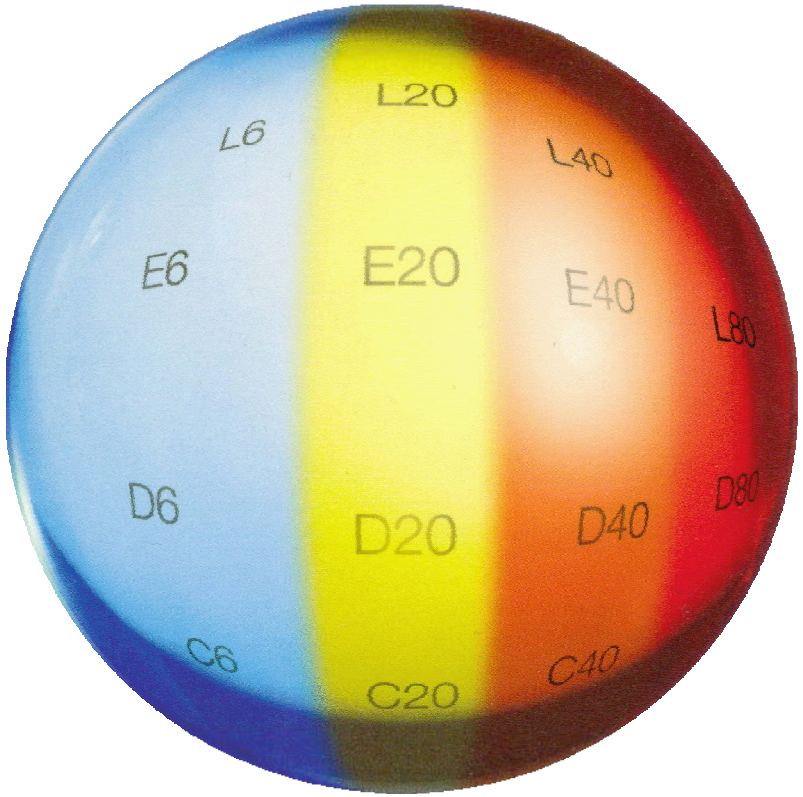
- Crystal Methods and Scrum 1st Agile Methods
- Extreme Programming swept the globe by 2002
- Scrum/Extreme Programming hybrids are popular

Year	Method	Author	Firm	Major Features
1991	Crystal Methods	Cockburn	IBM	Use Cases, Domain Models, Frequent Delivery, Reflection Workshops, Risk Management
1993	Scrum	Sutherland	Easel	Backlogs (Feature Lists), Daily Scrums, Sprints (Iterations), Retrospectives (Post Mortems)
1993	Dynamic Systems Development	Millington	DSDM	User Involvement, Time Boxes and Prototypes (Iterations), Testing and Quality Assurance
1997	Feature-Driven Development	De Luca	Nebulon	Feature Lists (Customer Needs), Domain Model (Object Orientation), Inspection (Peer Review)
1998	Extreme Programming	Beck	Chrysler	Release Planning, Onsite Customers, Iterations, Pair Programming, Test-Driven Development

Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley.

Crystal Methods

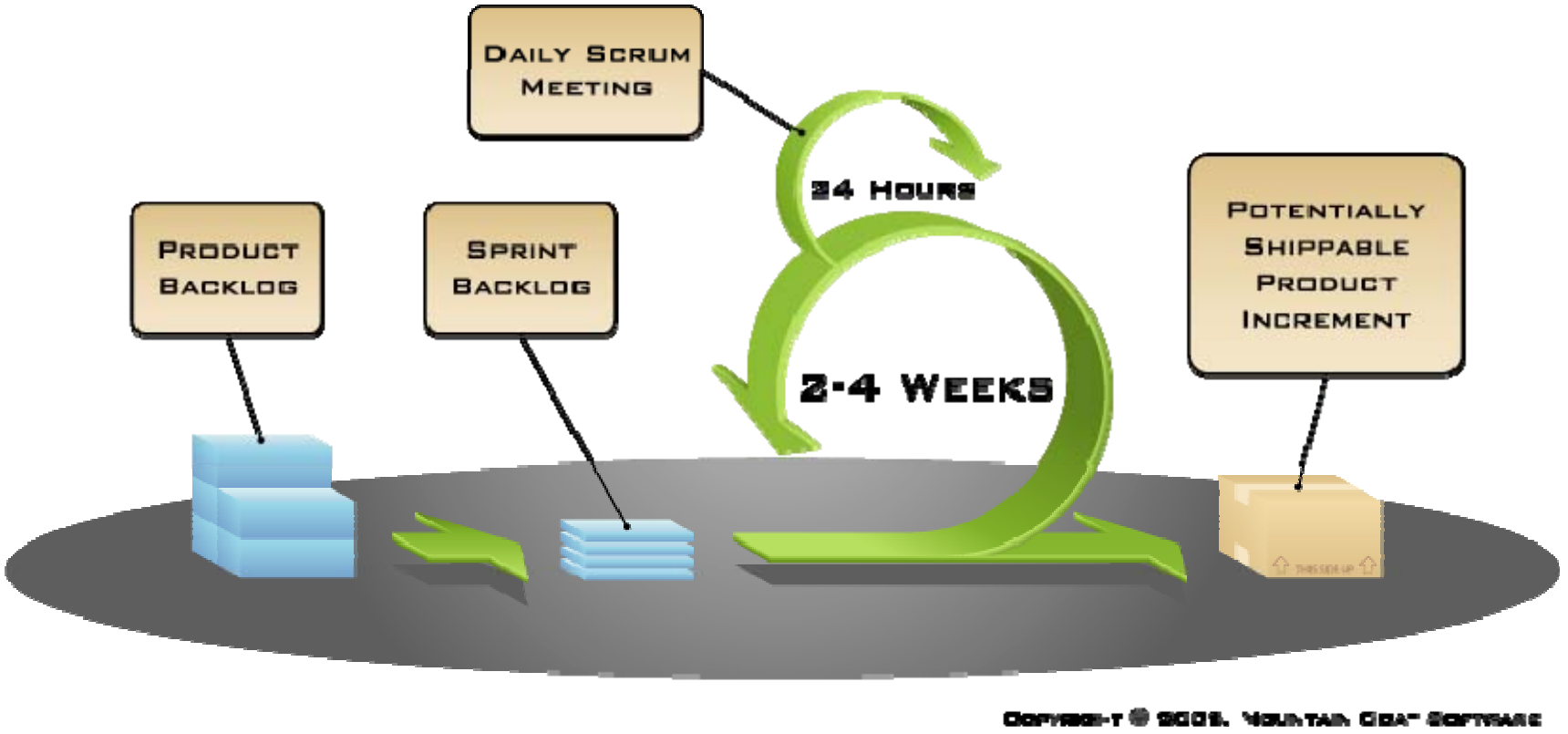
- Created by Alistair Cockburn in 1991
- Has 14 practices, 10 roles, and 25 products
- Scalable family of techniques for critical systems



Cockburn, A. (2002). *Agile software development*. Boston, MA: Addison-Wesley.

Scrum

- Created by Jeff Sutherland at Easel in 1993
- Has 5 practices, 3 roles, 5 products, rules, etc.
- Uses EVM to burn down backlog in 30-day iterations



Schwaber, K., & Beedle, M. (2001). *Agile software development with scrum*. Upper Saddle River, NJ: Prentice-Hall.

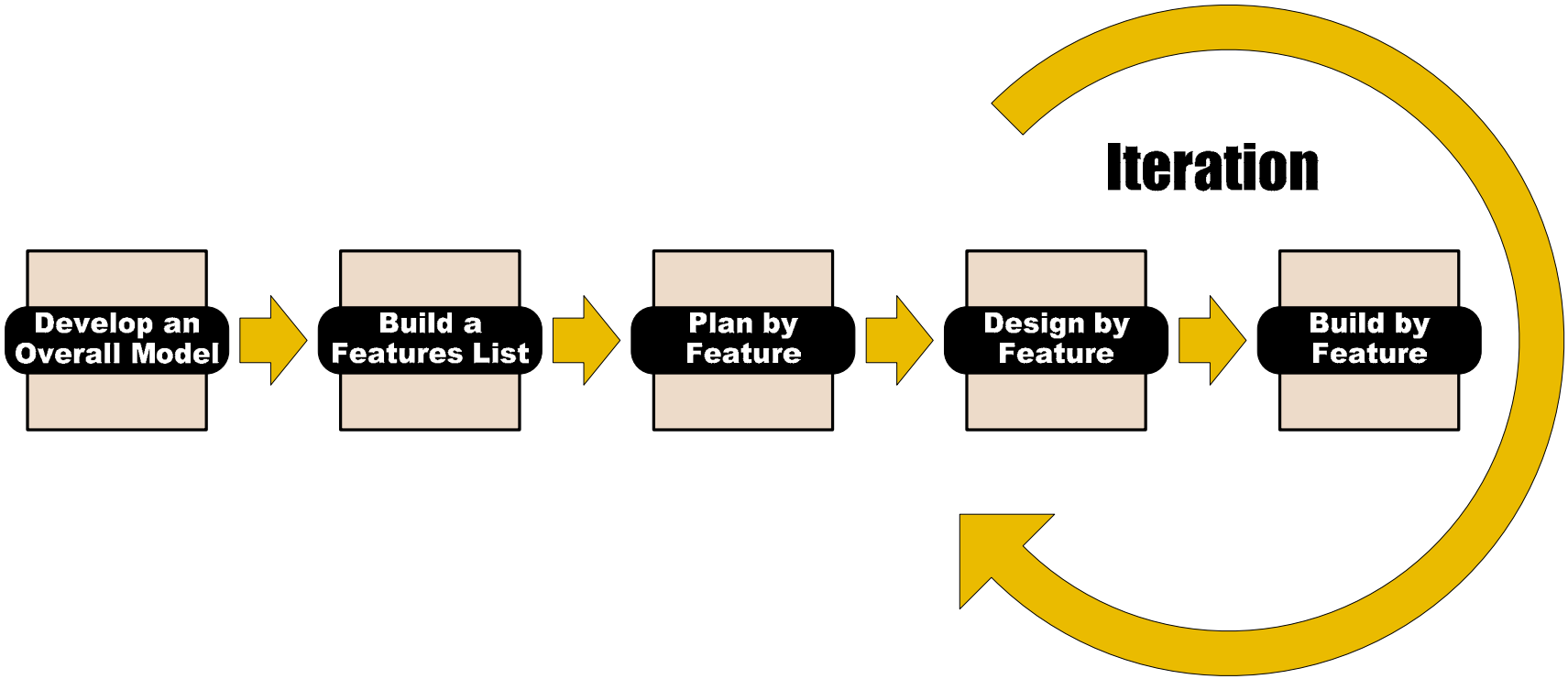
Dynamic Systems Develop.

- Created by group of British firms in 1993
- 15 practices, 12 roles, and 23 work products
- Non-proprietary RAD approach from early 1990s

Stapleton, J. (1997). *DSDM: A framework for business centered development*. Harlow, England: Addison-Wesley.

Feature Driven Development

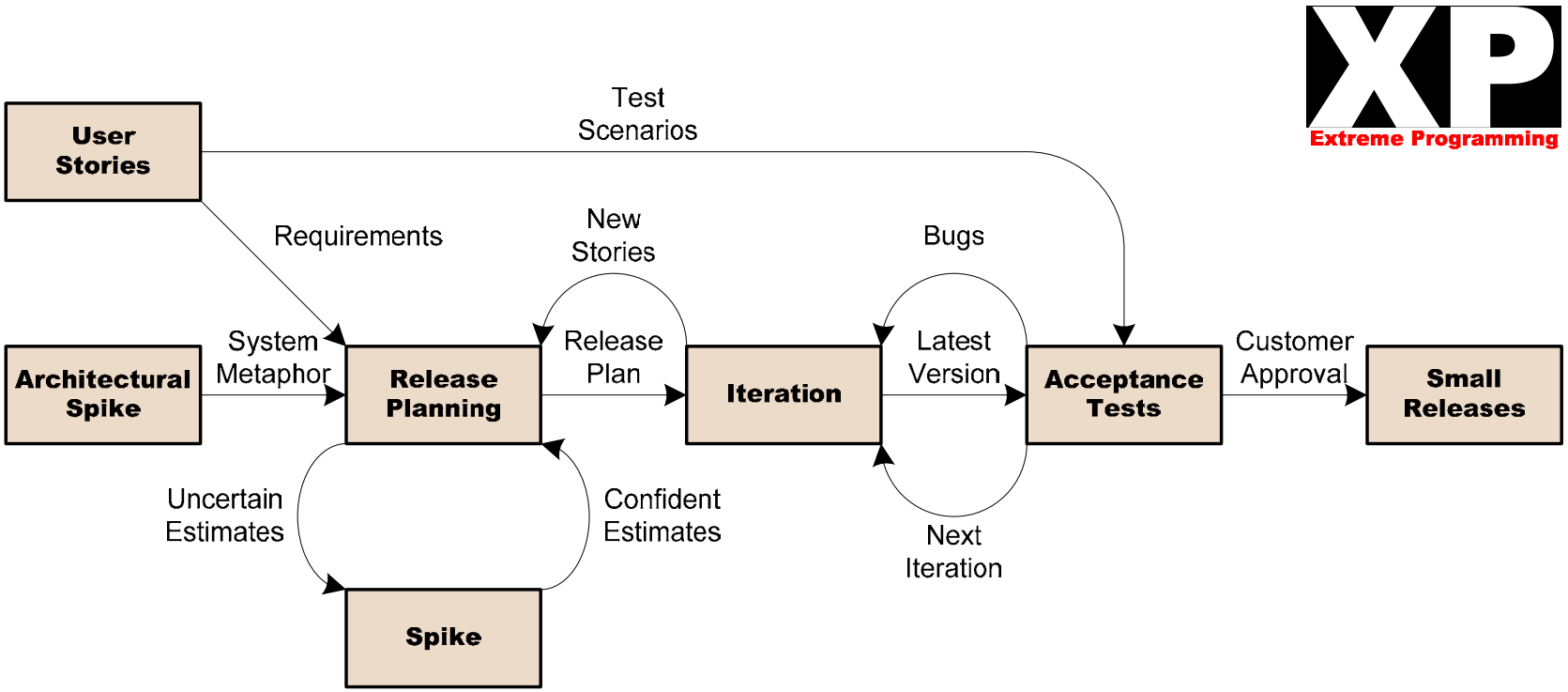
- Created by Jeff De Luca at Nebulon in 1997
- Has 8 practices, 14 roles, and 16 work products
- Uses object-oriented design and code inspections



Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature driven development*. Upper Saddle River, NJ: Prentice-Hall.

Extreme Programming

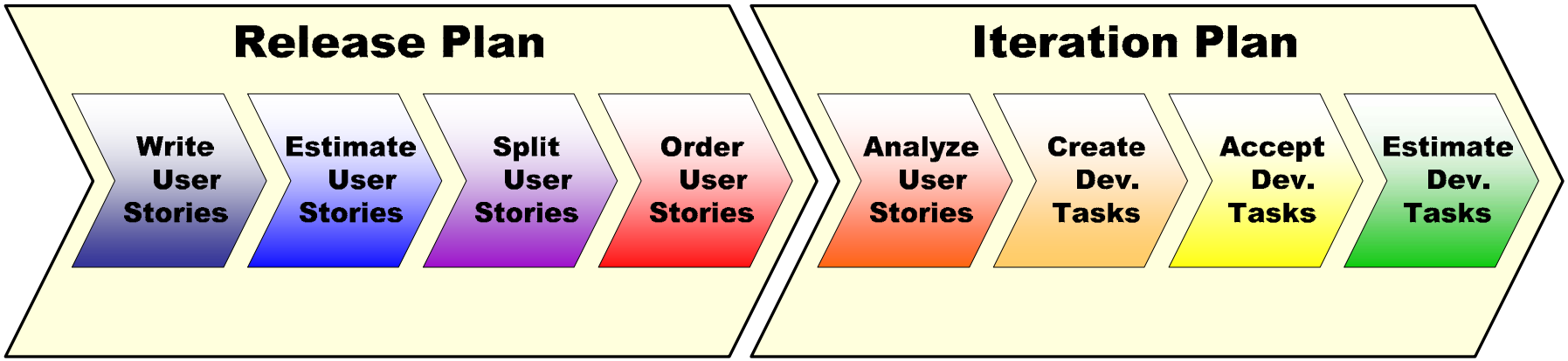
- Created by Kent Beck at Chrysler in 1998
- Has 28 practices, 7 roles, and 7 work products
- Popularized pair programming and test-driven dev.



Beck, K. (2000). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.

Extreme Programming (cont'd)

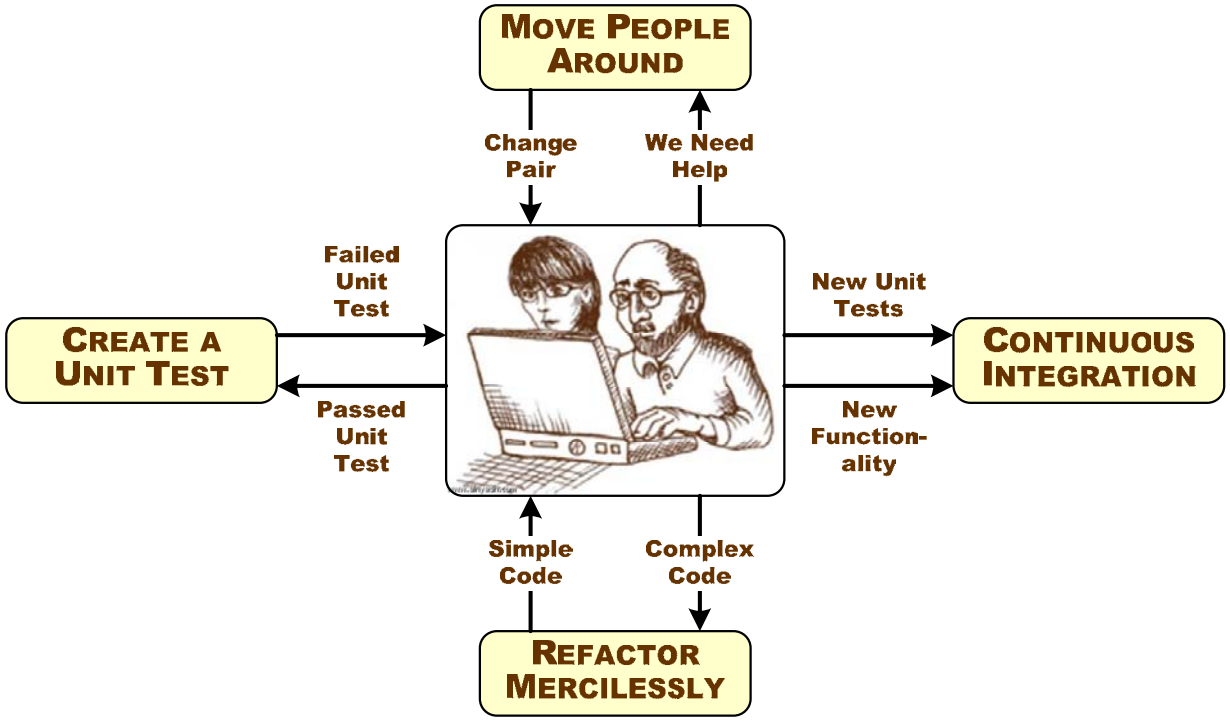
- RELEASE PLANNING — Best Practice
  - Created by Kent Beck at Chrysler in 1998
  - Lightweight project management framework
  - Used for managing both XP and Scrum projects



Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Extreme Programming (cont'd)

- PAIR PROGRAMMING — Best Practice
  - Term coined by Jim Coplien in 1995
  - Consists of two side-by-side programmers
  - Highly-effective group problem-solving technique



Williams, L., & Kessler, R. (2002). *Pair programming illuminated*. Boston, MA: Pearson Education.

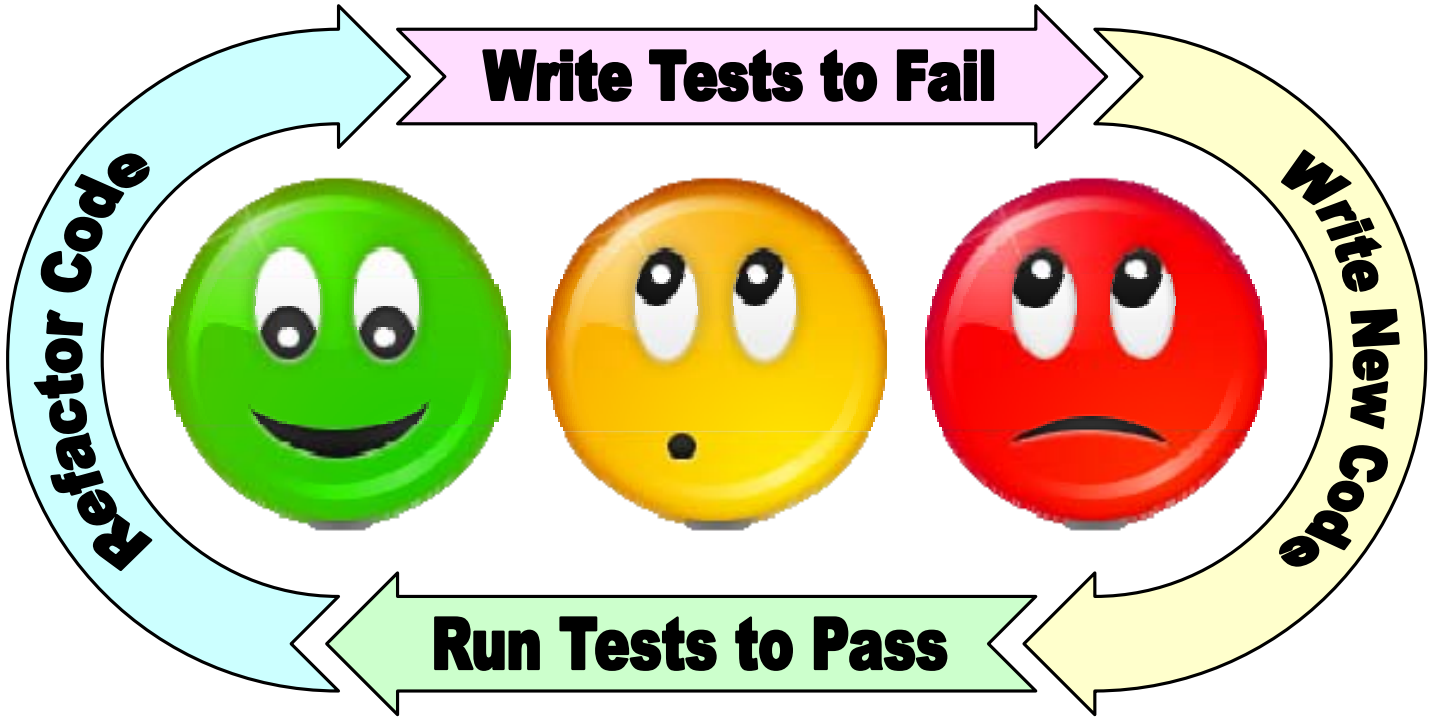
Extreme Programming (cont'd)

- REFACTORING — Best Practice
  - Term coined by William Opdyke in 1990
  - Process of frequently rewriting source code
  - Improves readability, maintainability, and quality

Fowler, M. (1999). *Refactoring: Improving the design of existing code*. Boston, MA. Addison-Wesley.

Extreme Programming (cont'd)

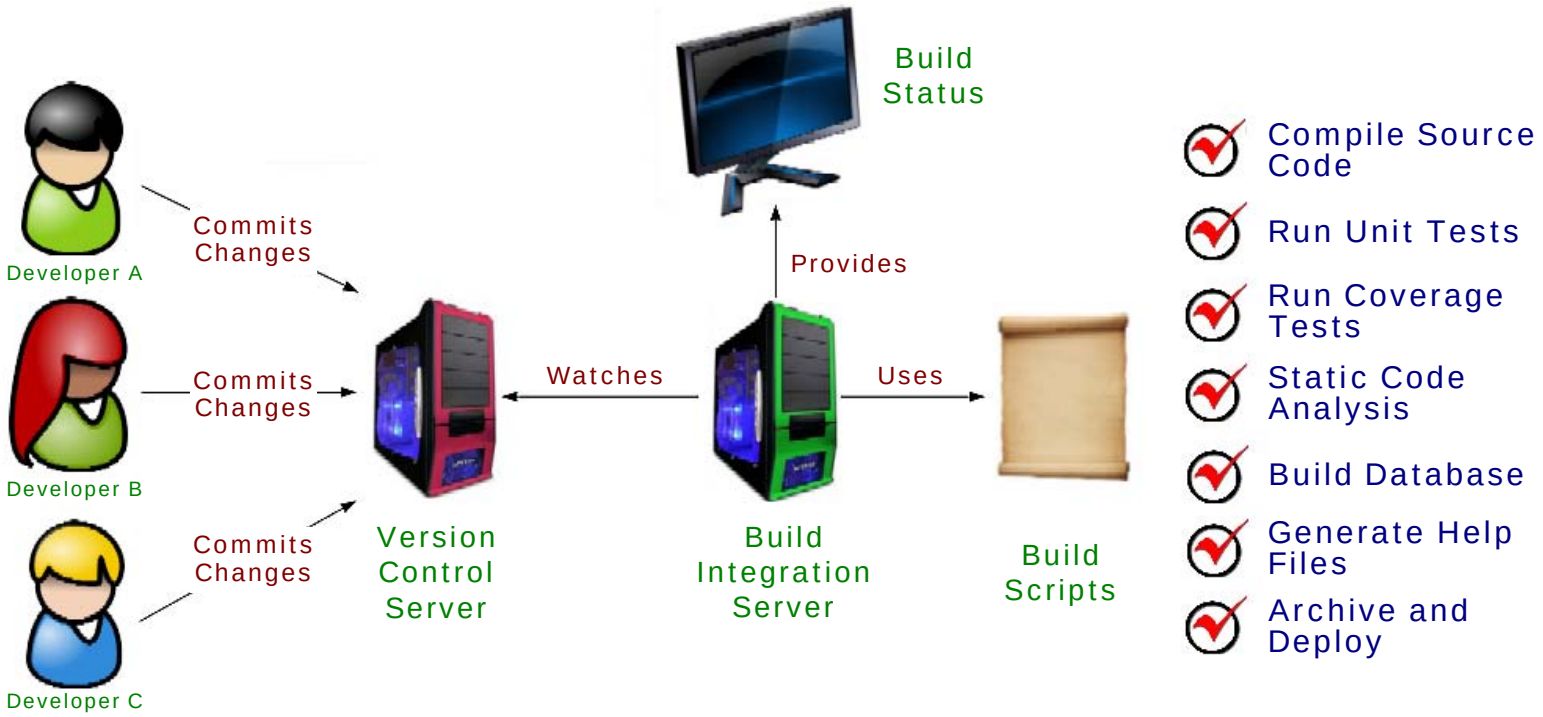
- TEST-DRIVEN DEV. — Best Practice
  - Term coined by Kent Beck in 2003
  - Consists of writing all tests before coding
  - Ensures all source code is verified and validated



Beck, K. (2003). *Test-driven development: By example*. Boston, MA: Addison-Wesley.

Extreme Programming (cont'd)

- CONT. INTEGRATION — Best Practice
  - Term coined by Martin Fowler in 1998
  - Process of automated build/regression testing
  - Evaluates impact of all changes against entire system



Duvall, P., Matyas, S., & Glover, A. (2006). *Continuous integration*: *Improving software quality and reducing risk*. Boston, MA: Addison-Wesley

Agenda

- Introduction
- Sources of Business Value
- ☞ Planning for Business Value
- Surveys of Business Value
- Measures of Business Value
- Models of Business Value
- Estimation of Business Value
- Comparison of Business Value
- Summary of Business Value

Release Planning Deliverables

- Used in both Extreme Programming and Scrum
- Lightweight framework of Agile planning products
- Ranges from release plans down through unit tests

No.	Deliverable	Description
1.	Release Plan	Fluid informal roadmap (or program plan) for planning software releases (usually containing one or more iterations).
2.	Iteration Plan	Informal project plan that divides an iteration into user stories and development tasks (usually spanning two to three weeks)
3.	User Story	A software requirement that has value to end-users or customers (usually a simple sentence written on an index card)
4.	Metaphor	A simple narrative about how the whole system works (usually written as a sentence or paragraph with major objects)
5.	Development Task	A development activity necessary to satisfy a user story (usually a numbered list of software development activities)
6.	Acceptance Test	An end-user test to determine if an iteration satisfies its acceptance criteria (usually written and executed by customers)
7.	Unit Test	A development test to determine if software components are working properly (usually written and executed by developers)

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Release Plan

- Fluid, informal roadmap for planning releases
- Includes dates for releases, iterations, and stories
- Must prioritize, split, estimate, and order user stories

Release Plan

<Release Plan>: <Release 1, 2, n>

<Release>: <Iteration 1, 2, n>

<Iteration>: <Story 1, 2, n>

Release Plan

Release	Iteration	Story
1	1	01 thru 06
1	2	07 thru 12
2	3	13 thru 18
2	4	19 thru 24
n	n	25 thru nn

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Iteration Plan

- Plan that divides iterations into development tasks
- Each iteration is one to three weeks in duration
- Iteration plans updated using daily standups

Iteration Plan

<Iteration Plan>: <Story 1, 2, n>

<Story>: <Task 1, 2, n>

<Task>: <Developer 1, 2, n>

<Status>: <Days Complete>

Iteration Plan

Story	Task	Developer	Status
1	1	Bob	1/3
1	2	Sue	2/3
2	3	Mary	3/3
2	4	John	3/3
n	n	n	n/n

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

User Story

- A function or feature of value to a customer
- An estimable and testable software requirement
- Six user stories should be implemented per iteration

<Title of User Story>

As a <Type of User> I can <Goal of User> so that <Objective of User>

Make a Reservation

As a customer, I can make a reservation so that I can perform personal travel

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

System Metaphor

- Simple story about how the whole system works
- Overarching 10,000 foot view of system architecture
- Pushes the system into a sense of coherent cohesion

System Metaphor

<Metaphor>: <Object 1>, <Object 2>, <Object n>

System Metaphor

Shopping Cart: Item, Description, Barcode, Price, Quantity, Subtotal, Tax, Discount, Total, etc.

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Development Tasks

- Detailed steps for implementing user stories
- User stories are decomposed into technical tasks
- Brainstormed by developers to last two to three days

<Title of Development Task>

<Action of Developer> a <Software Unit> using <Technology>

Splash Screen

Design a splash screen using PhotoShop

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Acceptance Tests

- Black-box, functional tests to be performed
- Specified by customers during iteration planning
- Run when user stories and unit tests are completed

<Title of User Story>

Verify <Type of User> can <Satisfy their Goals and Objectives>

Make a Reservation

- Verify customers can establish a reservation
- Verify customers can change a reservation
- Verify customers can cancel a reservation

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Unit Tests

- A test written from the developer's perspective
- Each task is implemented by two programmers
- Unit tests are developed prior to implementation

<Title of Development Task>

Verify <Type of User> can <Satisfy Task> when <Condition Occurs>

Make a Splash Screen

- Verify customers can see splash screen when they visit website
- Verify customers can see company logo when splash screen executes
- Verify customers can skip splash screen when they want to enter site

Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.

Agenda

- Introduction
- Sources of Business Value
- Planning for Business Value
- ☞ Surveys of Business Value
- Measures of Business Value
- Models of Business Value
- Estimation of Business Value
- Comparison of Business Value
- Summary of Business Value

Surveys of Agile Methods

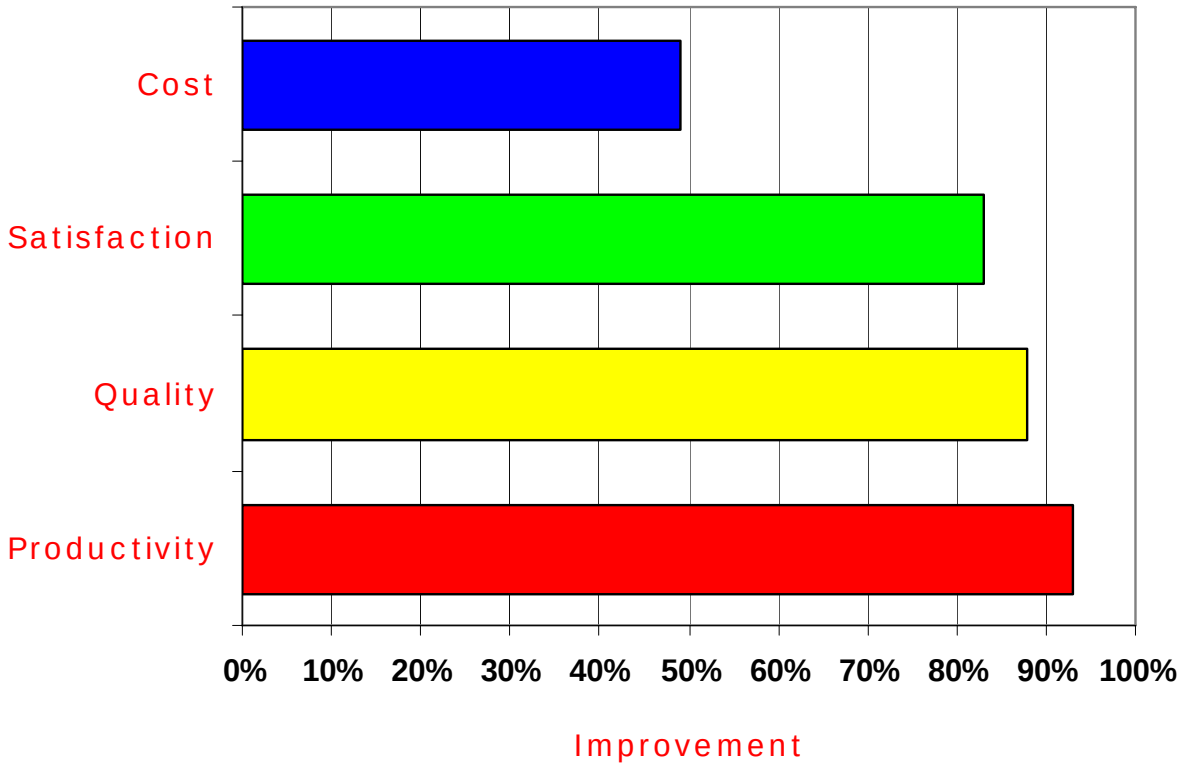
- Numerous surveys of Agile Methods since 2003
- AmbySoft and Version One collect annual data
- Generally include both hard and soft benefits

Year	Organization	Author	Size	Productivity	Quality	Cost
2003	Shine	Johnson	131	93%	88%	49%
2006	Agile Journal	Barnett	400	45%	43%	23%
2007	**Microsoft**	**Begel, et al.**	**492**	**14%**	**32%**	**16%**
2007	**UMUC**	**Rico, et al.**	**250**	**81%**	**80%**	**75%**
2008	AmbySoft	Ambler	642	82%	72%	72%
2008	IT Agile	Wolf, et al.	207	78%	74%	72%
2008	Version One	Hanscom	3,061	74%	68%	38%
Average				**67%**	**65%**	**49%**

Rico, D. F. (2008). *What is the return-on-investment of agile methods*? Retrieved February 3, 2009, from http://davidfrico.com/rico08a.pdf

Shine Technologies

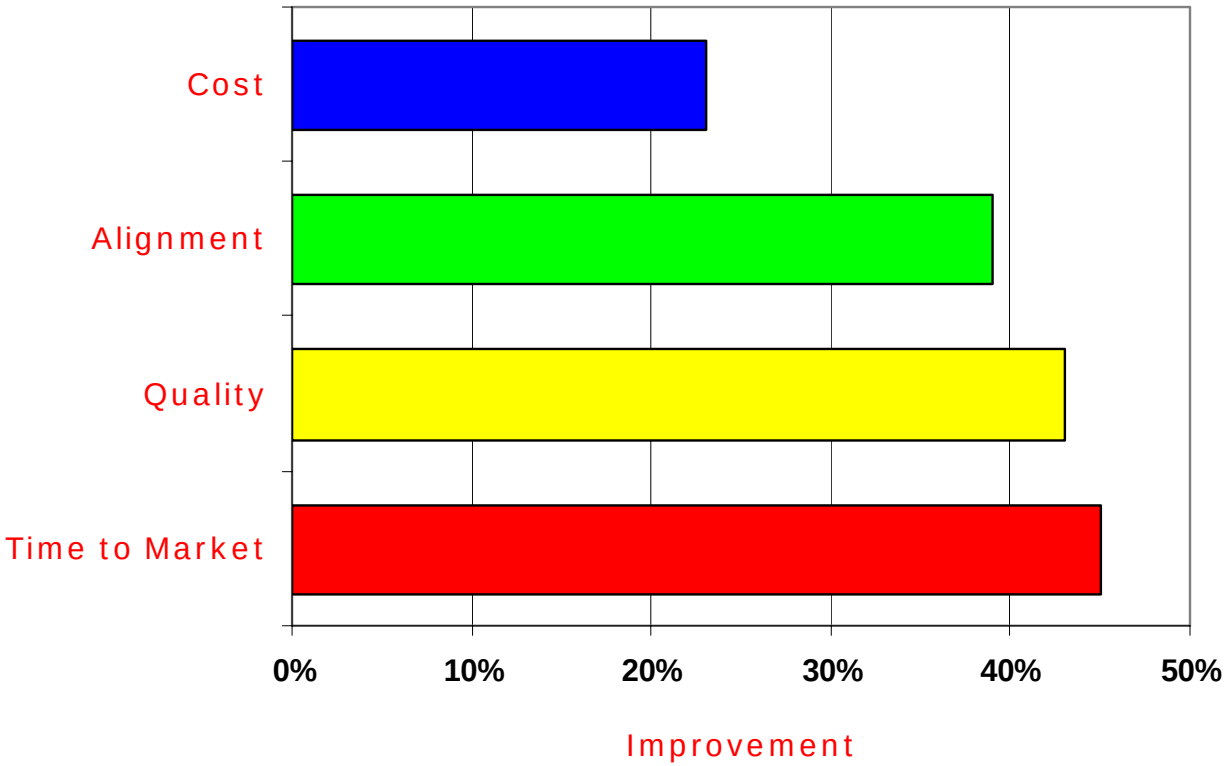
- Survey of 131 international respondents
- Extreme Programming (58%) and Scrum (8%)
- 85% of respondents were experts in Agile Methods



Johnson, M. (2003). *Agile methodologies*: *Survey results*. Victoria, Australia: Shine Technologies.

Agile Journal

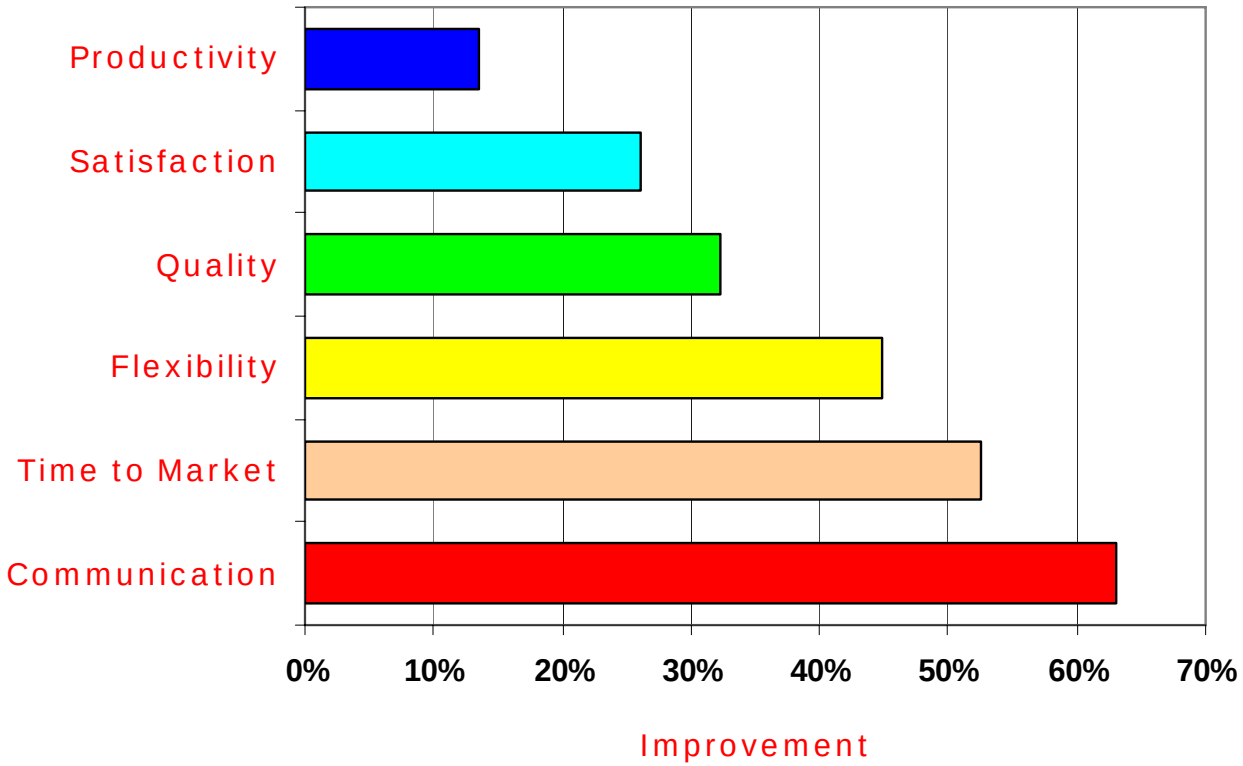
- Survey of 400 international respondents
- Extreme programming (28%) and Scrum (20%)
- 80% using Agile Methods to deliver maximum value



Barnett, L. (2006). And the agile survey says. *Agile Journal*, *1*(1).

Microsoft

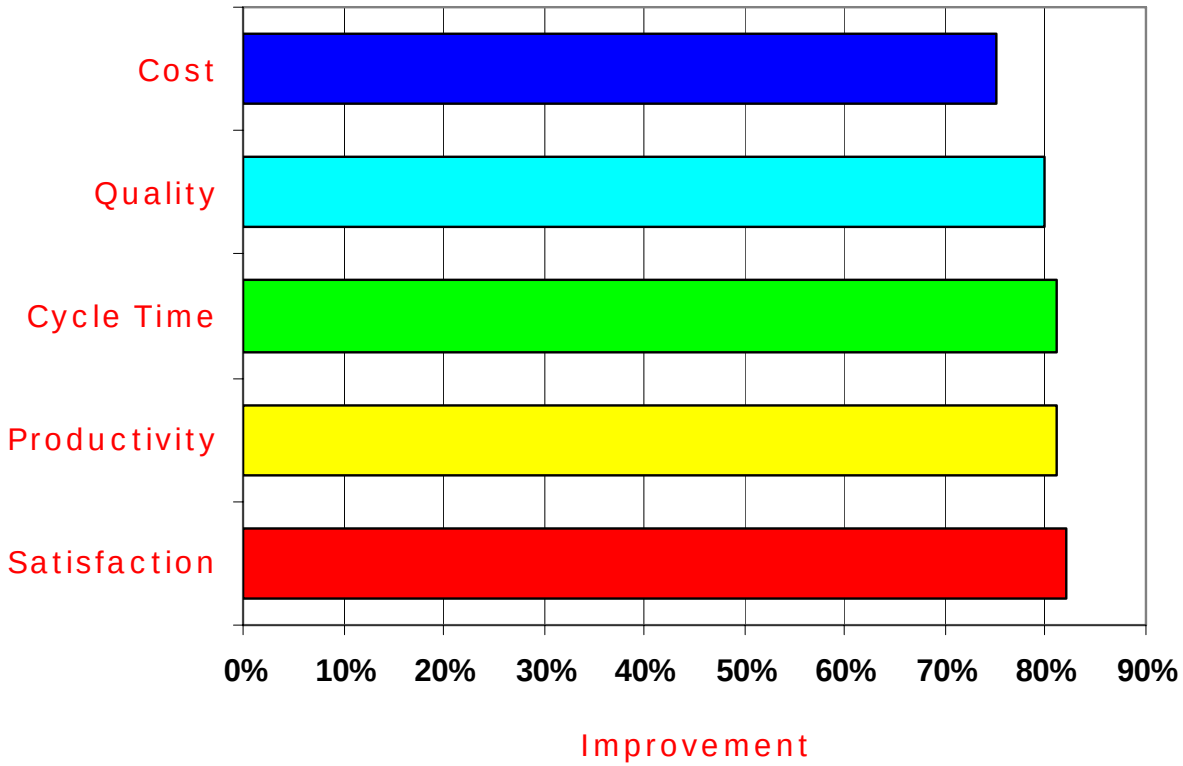
- Survey of 492 Microsoft respondents
- Scrum (65%) and Extreme Programming (5%)
- 65% using Agile Methods in virtual distributed teams



Begel, A., & Nagappan, N. (2007). Usage and perceptions of agile software development in an industrial context: An exploratory study. *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, Madrid, Spain*, 255-264.

UMUC

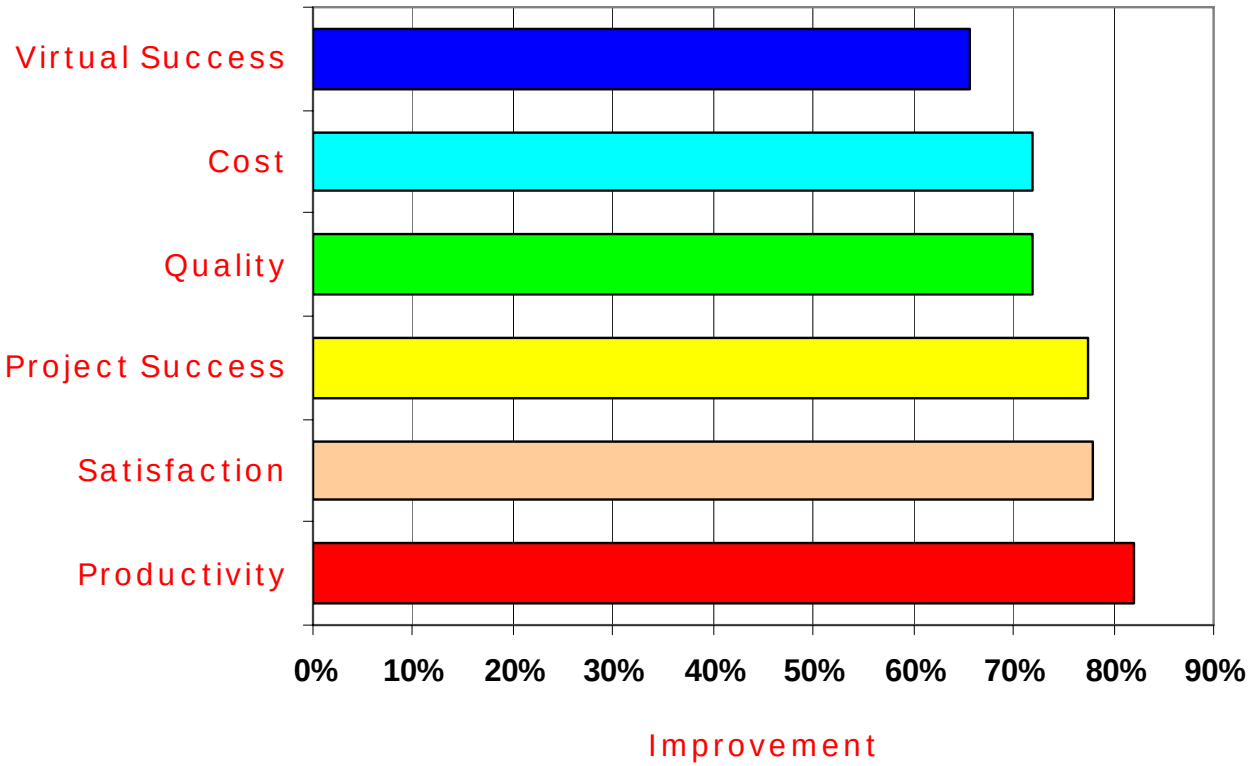
- Survey of 250 international respondents
- 70% of respondents using Agile Methods
- 83% of were from small-to-medium sized firms



Rico, D. F., Sayani, H. H., Stewart, J. J., & Field, R. F. (2007). A model for measuring agile methods and website quality. *TickIT International*, *9*(3), 3-15.

AmbySoft

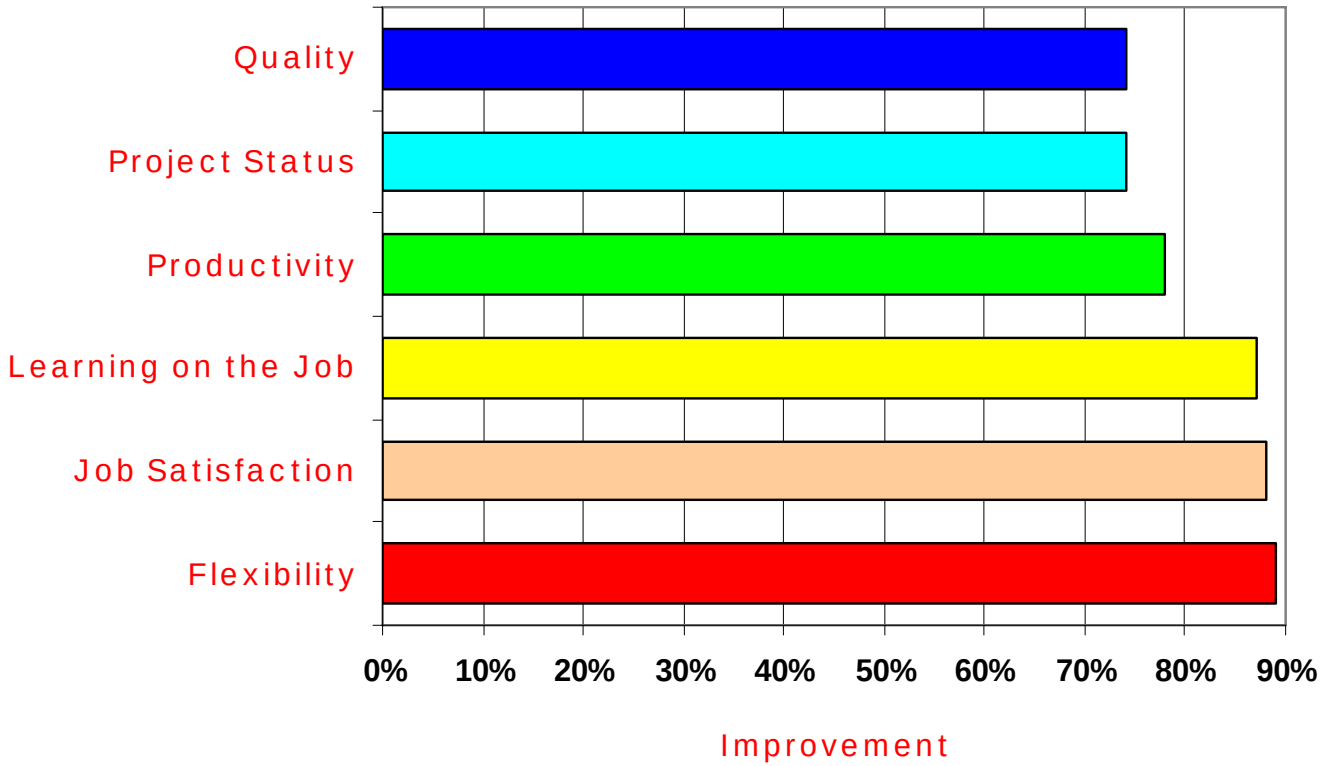
- Survey of 642 international respondents
- 69% of firms had adopted an Agile Method
- 62% were from firms with less than 1,000 people



Ambler, S. W. (2008). *Agile adoption survey*. Retrieved October 17, 2008, from http://www.ambysoft.com/downloads/surveys/AgileAdoption2008.ppt

IT Agile

- Survey of 207 respondents in Germany
- Scrum (21%) and Extreme Programming (14%)
- 97% of respondents are satisfied with Agile Methods



Wolf, H., & Roock, A. (2008). Agile becomes mainstream: Results of an Online Survey. *Object Spektrum*, *15*(3), 10-13.