



A Short Introduction to **Agile Methods**

A synopsis based on the background, examples, deliverables, costs, benefits, and unique features

Dr. David F. Rico, PMP, CSM



Agenda

 **Background**

Examples

Deliverables

Surveys

Business Value

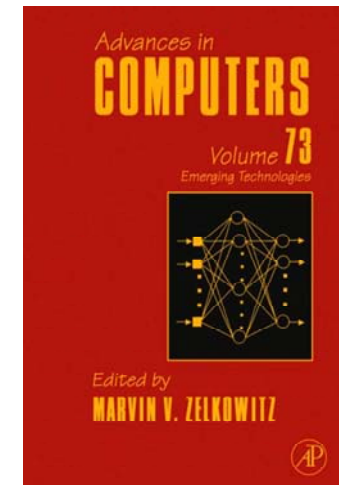
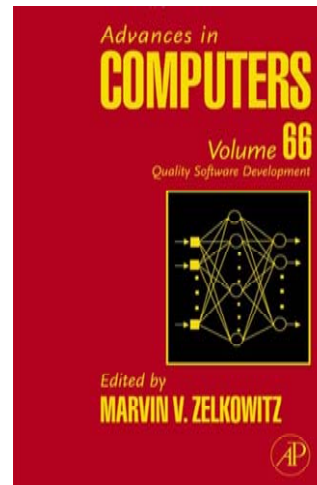
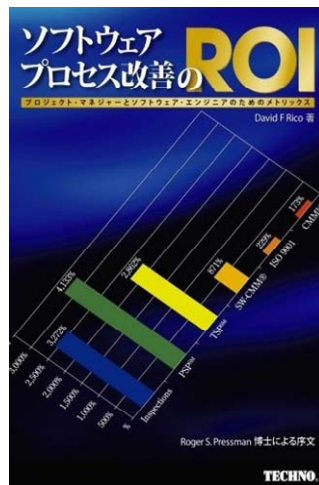
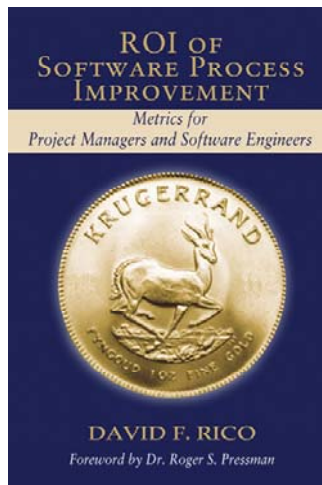
Other Considerations

Conclusion

References

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 Agile Methods using examples, artifacts, benefits, and other data
 - *Agility is the ability to create and respond to change in order to profit in a turbulent business environment*
 - *Agility is prioritizing for maneuverability with respect to shifting requirements, technology, and knowledge*
 - *Agile methods use time-boxed iterations, adaptability, and evolutionary delivery to promote rapid flexibility*
 - *Agile methods promote quick response to changes in requirements as well as collaboration with customers*
 - *Agile methods are a better way of developing software using teams, collaboration, iterations, and flexibility*

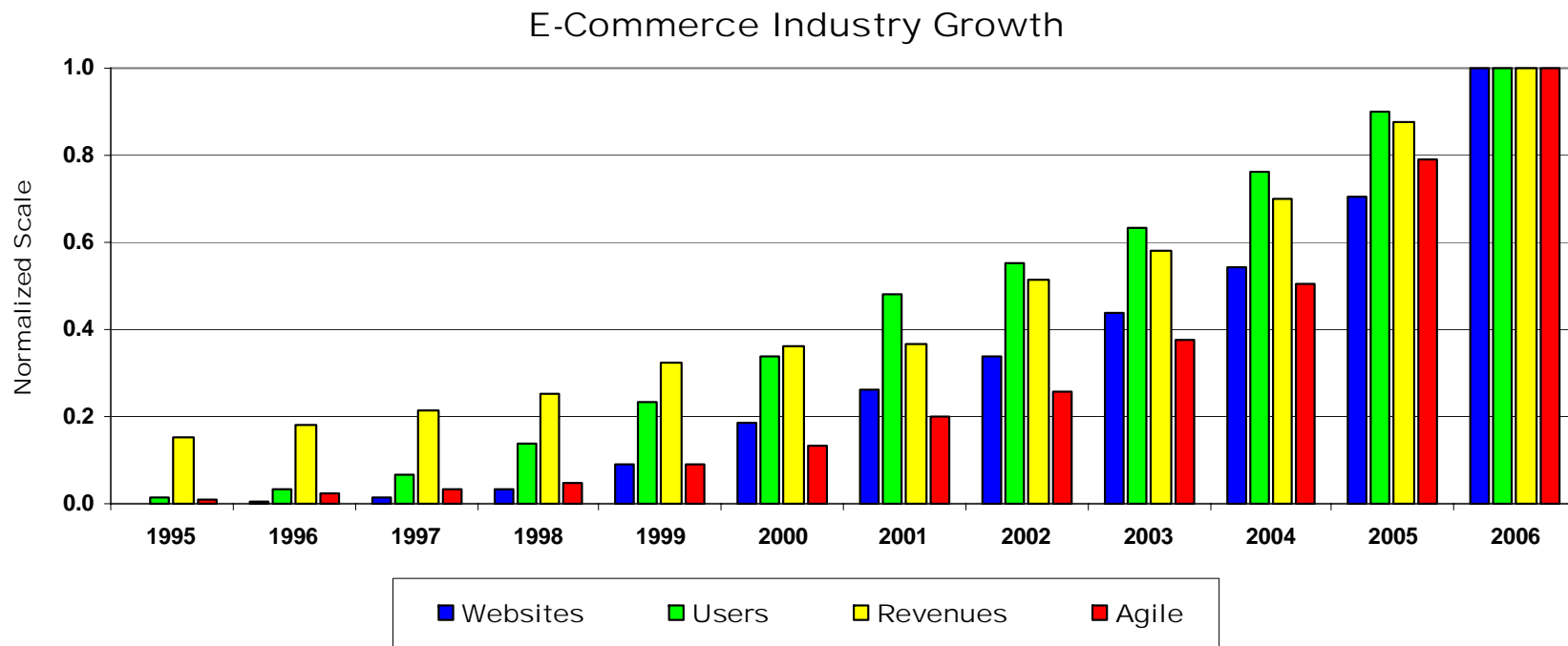


Key Terms

- Software method. An approach to the analysis, design, construction, and implementation of information systems.
- Traditional method. A software method with a focus on contracts, planning, processes, documentation, and tools.
- Agile method. A software method with a focus on teams, collaboration, working software, and responding to change.
- Software team. Small group responsible for making decisions, establishing needs, creating software, and ensuring success.
- Customer collaboration. A method of customer interaction and participation to obtain feedback and establish user needs.
- Iterative development. Creation of a large number of small, frequent, and time-boxed working operational software releases.
- Adaptability. A culture, attitude, process, and product enabling rapid, flexible, and easy adaptation to evolving customer needs.

E-Commerce Industry Growth

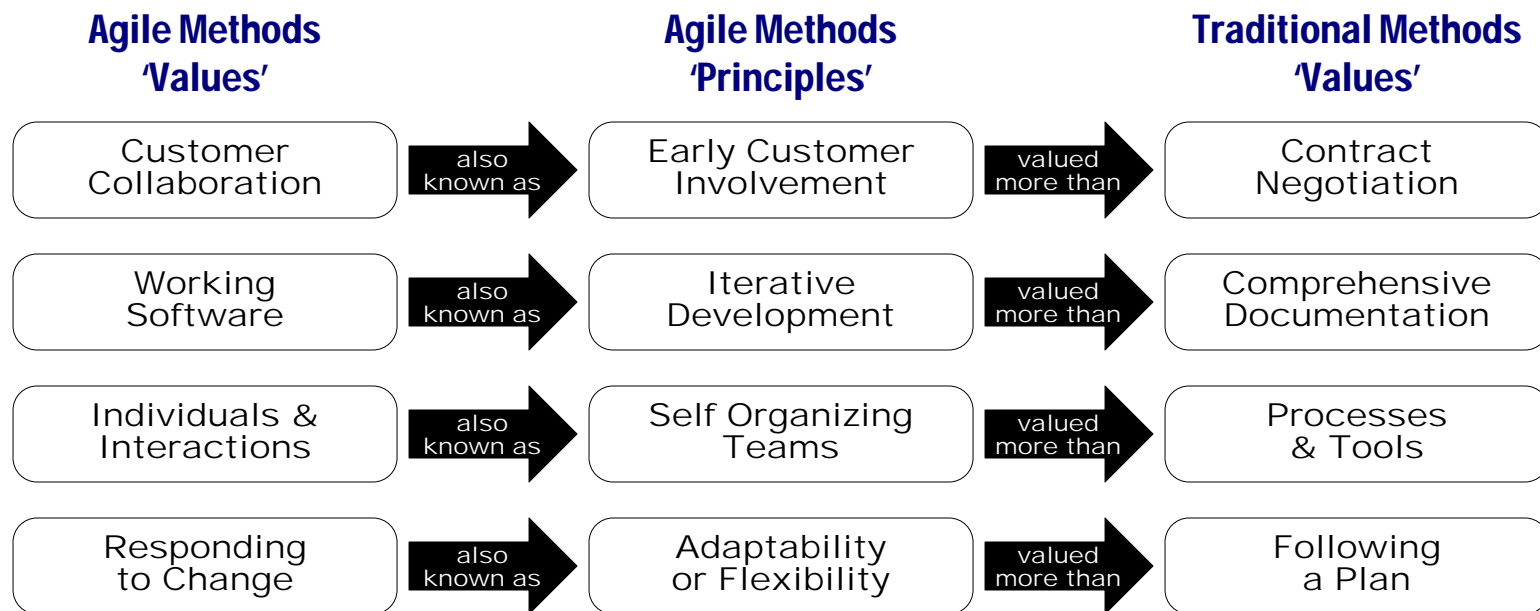
- No. of **websites** increased 4,609x (105 million)
- No. of Internet **users** increased 67x (1.07 billion)
- E-commerce **revenues** increased 7x (\$2.3 trillion)



Rico, D. F. (2008). *Internet and information technology growth statistics: 1995 to 2006*. Retrieved September 1, 2008, from <http://davidfrico.com/it-stats.xls>

What are Agile Methods?

- 'Lightweight' software development methodologies
- 'Human-centric' approach to creating business value
- 'Alternative' to heavy document-based methodologies



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

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 vs. Traditional Methods

- Sloanism vs. Taylorism and Fordism
- Craft-industry vs. scientific management
- Personal vs. impersonal human interactions

Pine		Boehm		Nerur	
Agile	Traditional	Agile	Traditional	Agile	Traditional
<ul style="list-style-type: none"> • Job-shop oriented • Mass customization • Micro-markets • Sloanism • Manufacturing cells • Egalitarian • Decentralized • Empowerment • Multi-disciplinary • Collectivism • Economy of scope • Value • Effectiveness • Elegance • Capability-based • Customer satisfact. 	<ul style="list-style-type: none"> • Manufacturing • Mass production • Mass market • Fordism • Production lines • Autocratic • Centralized • Division of labor • Specialization • Individualism • Economy of scale • Cost • Efficiency • Reliability • Defect reduction • Quality control 	<ul style="list-style-type: none"> • Unpredictable • Small projects • Turbulent • Customer-centric • Informal artifacts • Tacit interaction • User stories • Simple design • Automated testing • Customer presence • Developer-centric • Egalitarian 	<ul style="list-style-type: none"> • Predictable • Large projects • Stability • Contract-centric • Formal documents • Written interaction • Specifications • Formal architecture • Formal test plans • No customer • Analyst-centric • Authoritarian 	<ul style="list-style-type: none"> • Unpredictable • People-centric • Egalitarian • Tacit knowledge • Informal artifacts • Multi-disciplinary • Informal comm. • Customer-critical • Feature-focused • Iterative process • Organic • Object-oriented 	<ul style="list-style-type: none"> • Predictable • Process-centric • Authoritarian • Explicit knowledge • Formal documents • Specialization • Formal comm. • Customer-important • Activity-focused • Linear process • Mechanistic • Tech. agnostic

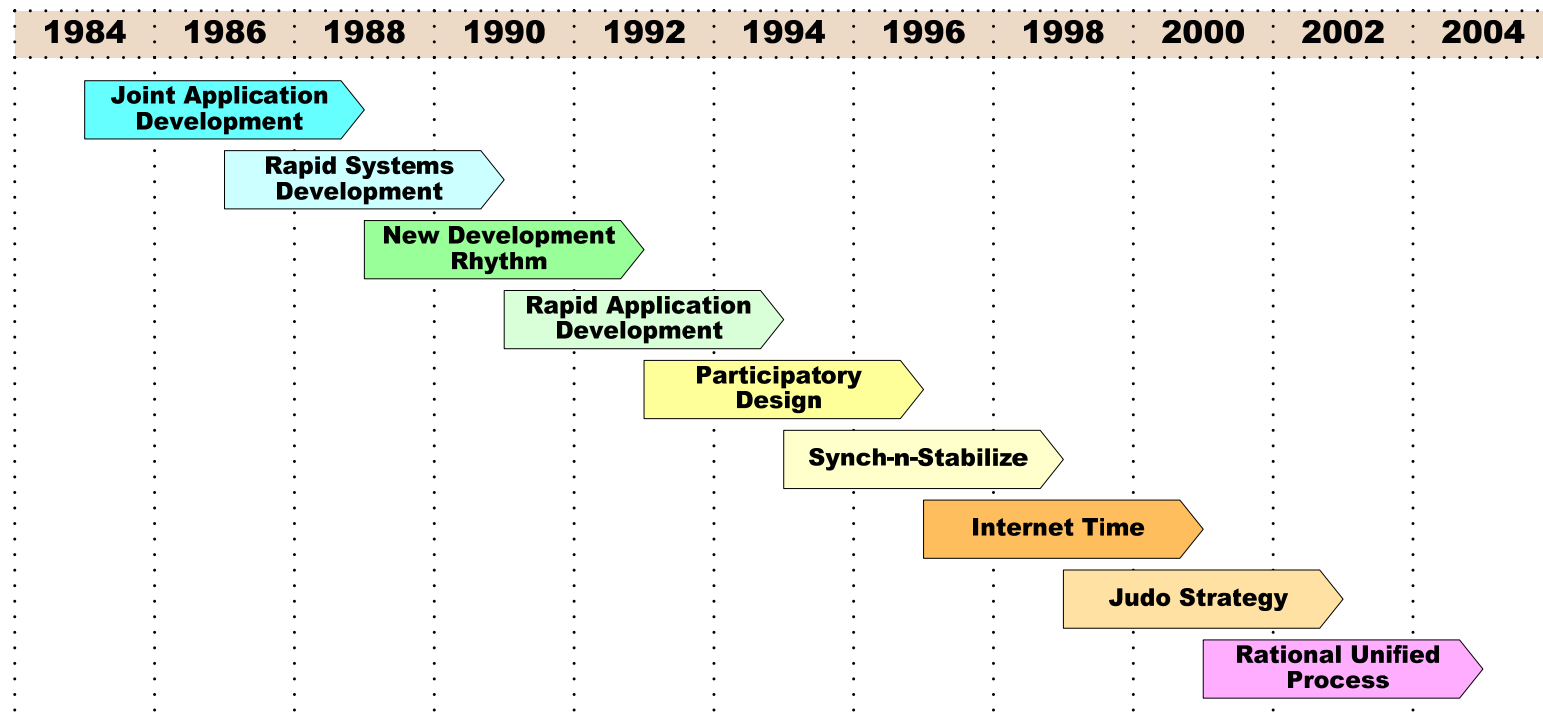
Pine, B. J. (1992). *Mass customization: The new frontier in business competition*. Boston, MA: Harvard Business School Press.

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Boston, MA: Addison-Wesley.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5), 73-78.

Antecedents of Agile Methods

- JAD involved customers in requirements analysis
- PD involved customers in architecture and designs
- Judo Strategy involved customers in implementation



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.

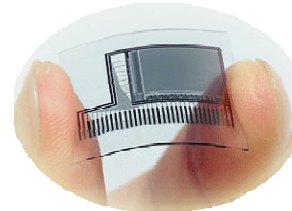
Essence of Agile Methods

- High degree of developer and customer interaction
- Just-enough process to get the job done on time
- Iterative product refinement to satisfy needs

Self Organizing Teams



Adaptability or Flexibility



Customer Involvement



Iterative Development

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



Agenda

Background

 **Examples**

Deliverables

Surveys

Business Value

Other Considerations

Conclusion

References

"Big 5" Agile Methods

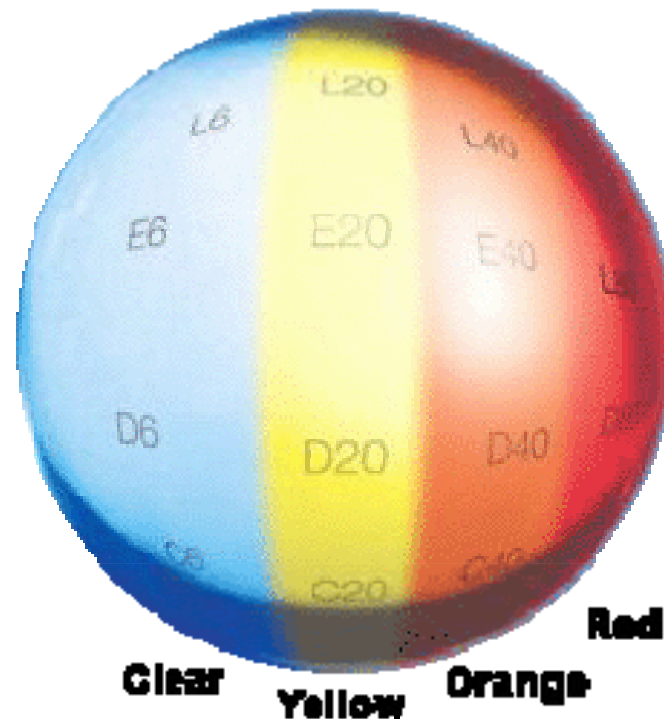
- ❑ Crystal Clear and Scrum first Agile Methods
- ❑ Extreme Programming swept the globe by 2001
- ❑ Scrum and Extreme Programming now most popular

Year	Method	Author	Firm	Major Features
1991	Crystal Clear	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)
1999	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 Clear

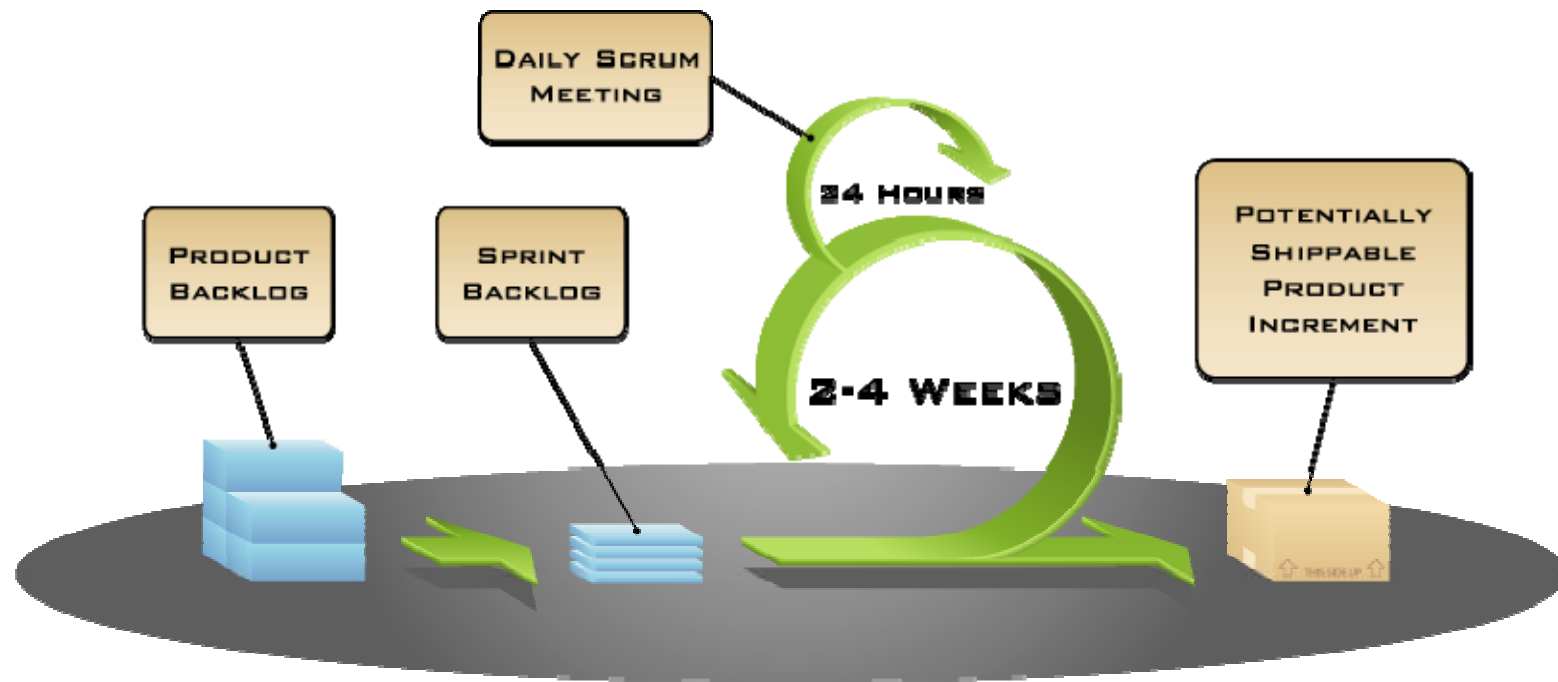
- ❑ Created by Alistair Cockburn in 1991
- ❑ Consists of 5 goals, 9 practices, and 8 roles
- ❑ Scalable family of techniques for critical systems



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

Scrum

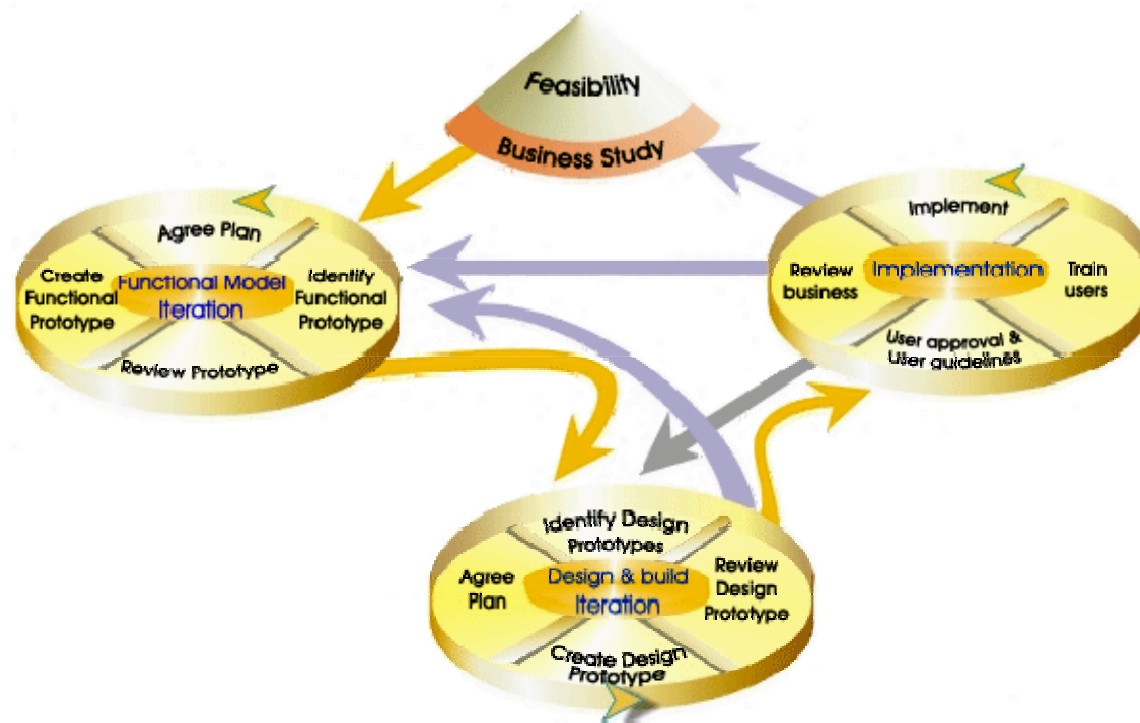
- Created by Jeff Sutherland at Easel in 1993
- Three basic phases—Planning, sprint, post-sprint
- Uses EVM to burn down backlog in 30-day iterations



Copyright © 2005, Mountain Goat Software

Dynamic Systems Develop.

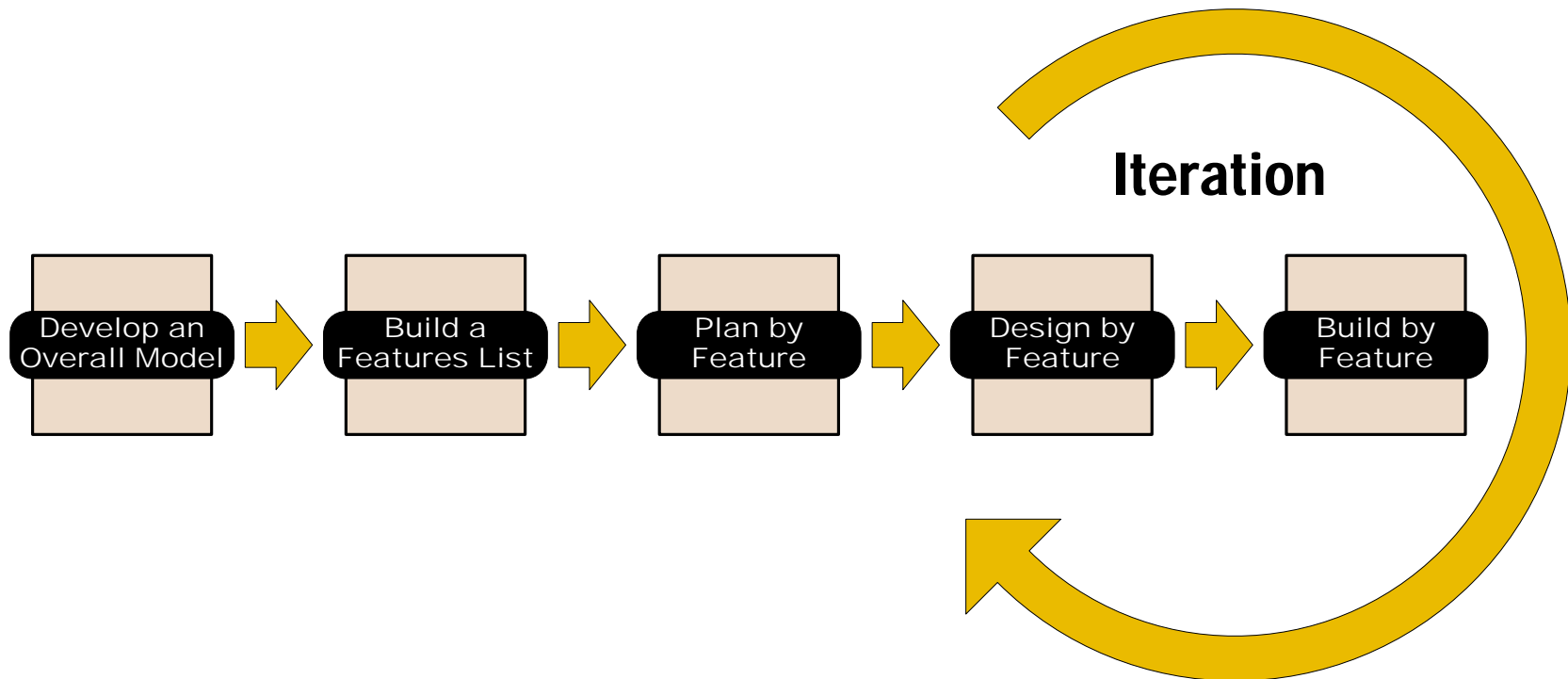
- Created by consortium of British firms in 1993
- Consists of 5 phases, 15 practices, and 12 roles
- Non-proprietary RAD approach from the early 1990s



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

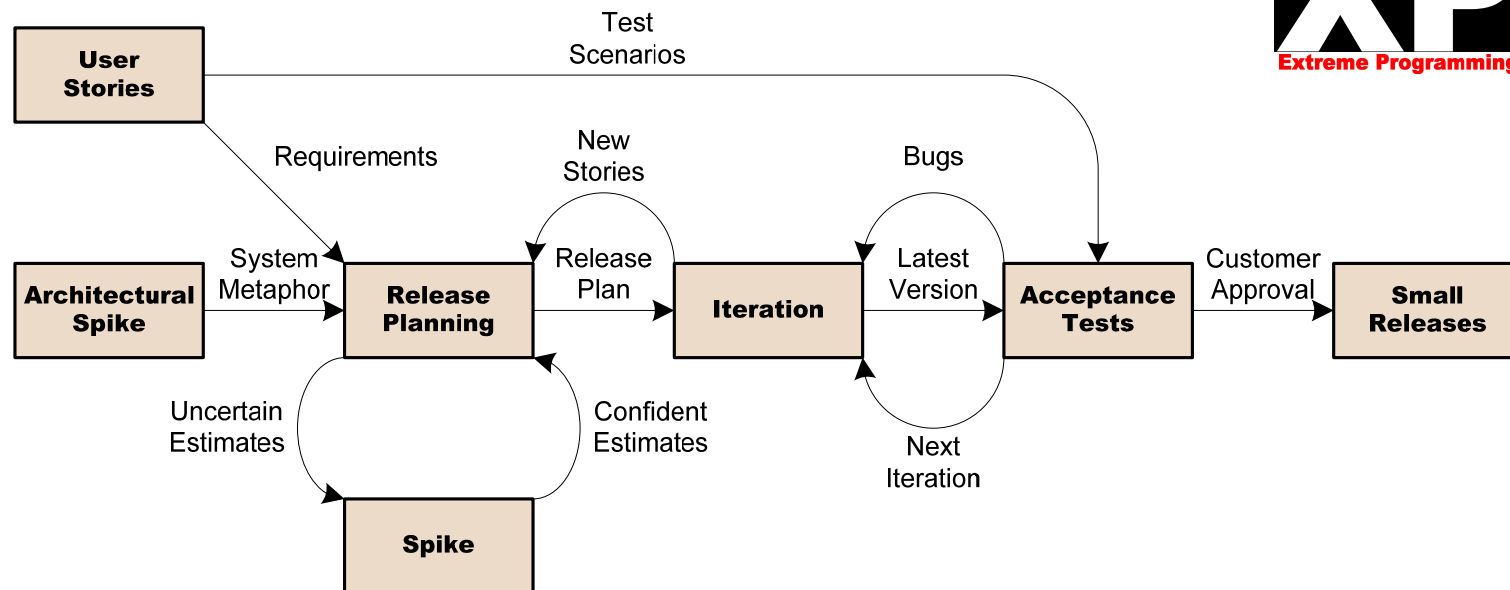
Feature Driven Development

- Created by Jeff De Luca at Nebulon in 1997
- Consists of 5 phases, 29 tasks, and 8 practices
- Uses object oriented design and peer/code reviews



Extreme Programming

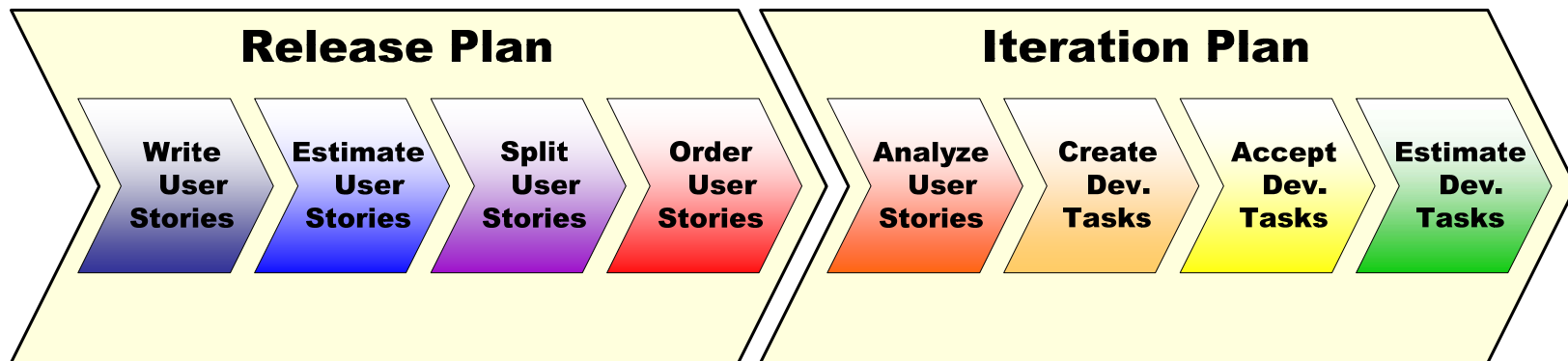
- Created by Kent Beck at Chrysler in 1999
- Grown from 13 to more than 28 rules/practices
- Popularized pair programming and test-driven dev.



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

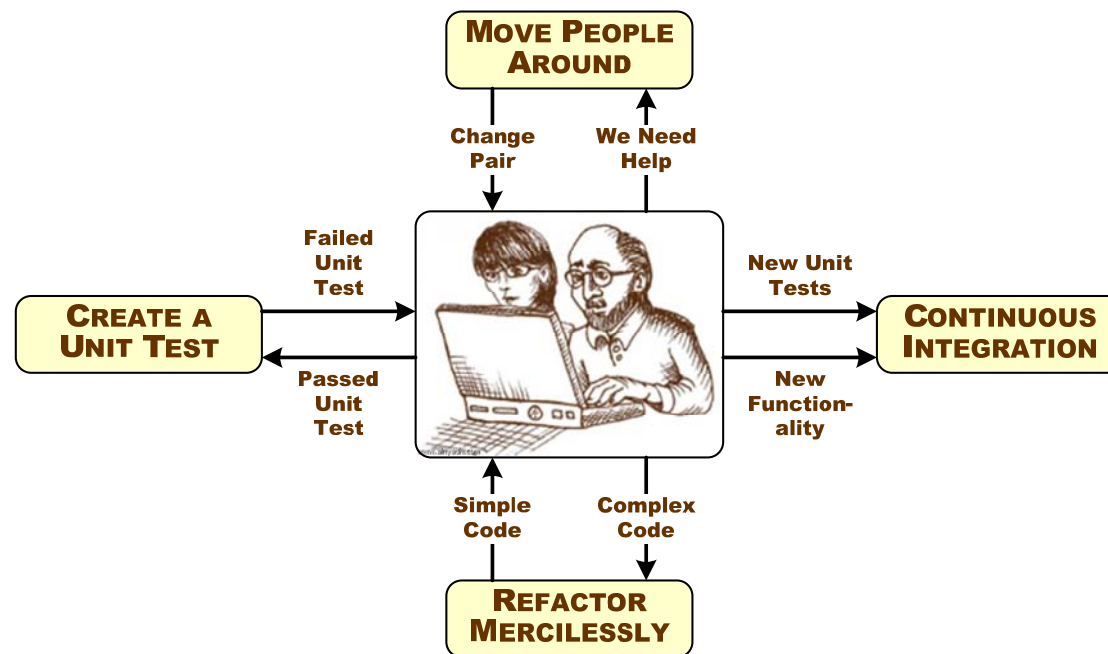
Extreme Programming (cont'd)

- **RELEASE PLANNING — Best Practice**
 - Created by Kent Beck at Chrysler in 1999
 - Consists of user stories and development tasks
 - Used as project planning process for XP and Scrum



Extreme Programming (cont'd)

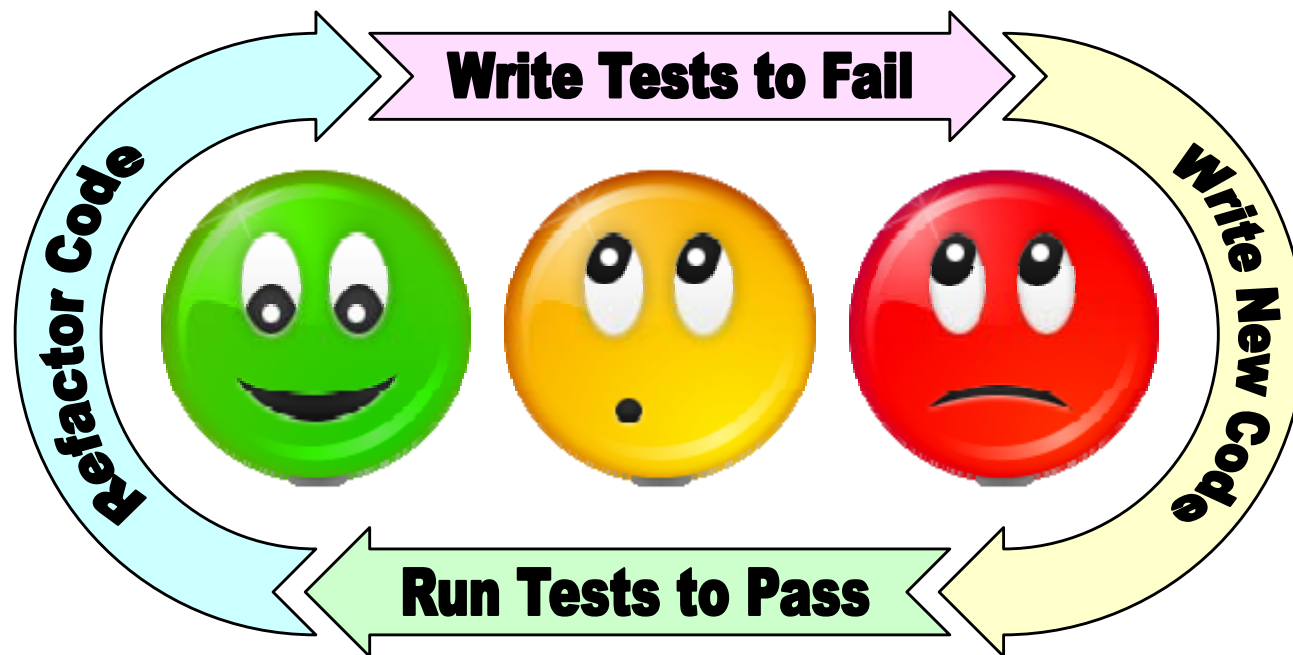
- **PAIR PROGRAMMING — Best Practice**
 - Term coined by Jim Coplien in 1995
 - Consists of two side-by-side programmers
 - Considered an efficient problem solving technique



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

Extreme Programming (cont'd)

- **TEST-DRIVEN DEVELOPMENT (TDD)**
 - Term coined by Kent Beck in 2003
 - Consists of writing unit tests before coding
 - Believed to be a primary means of quality control



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



Agenda

Background

Examples

 **Deliverables**

Surveys

Business Value

Other Considerations

Conclusion

References

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

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

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

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.

Development Tasks

- Customers read story to communicate expectations
- Developers brainstorm tasks to satisfy user stories
- Development tasks should 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

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

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



Agenda

Background

Examples

Deliverables

 **Surveys**

Business Value

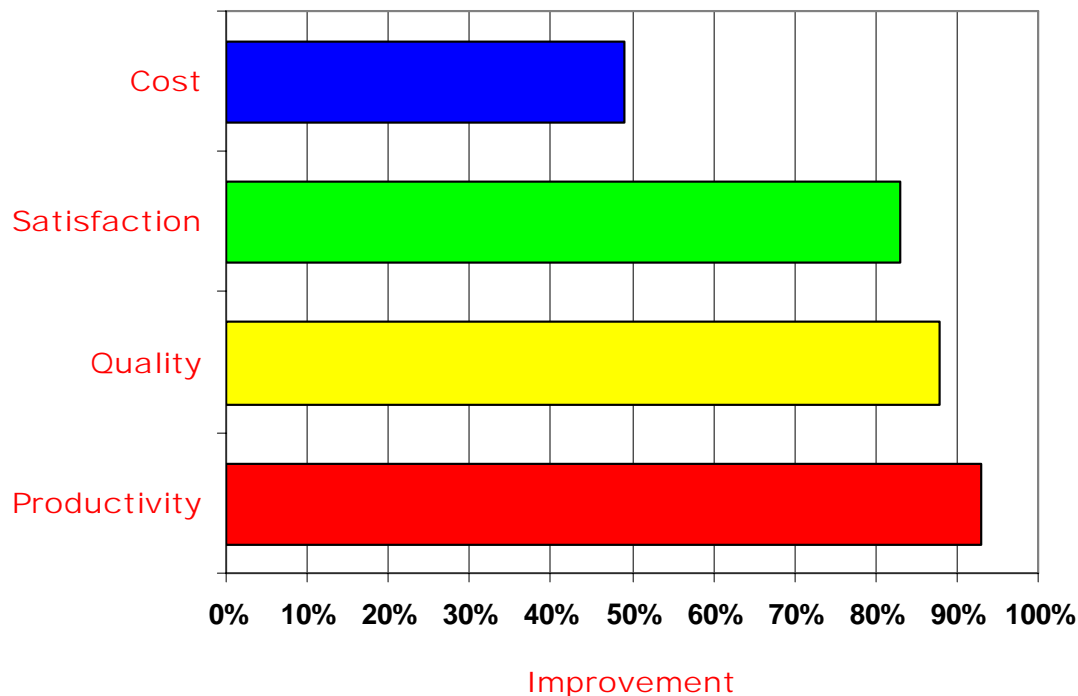
Other Considerations

Conclusion

References

Shine Technologies

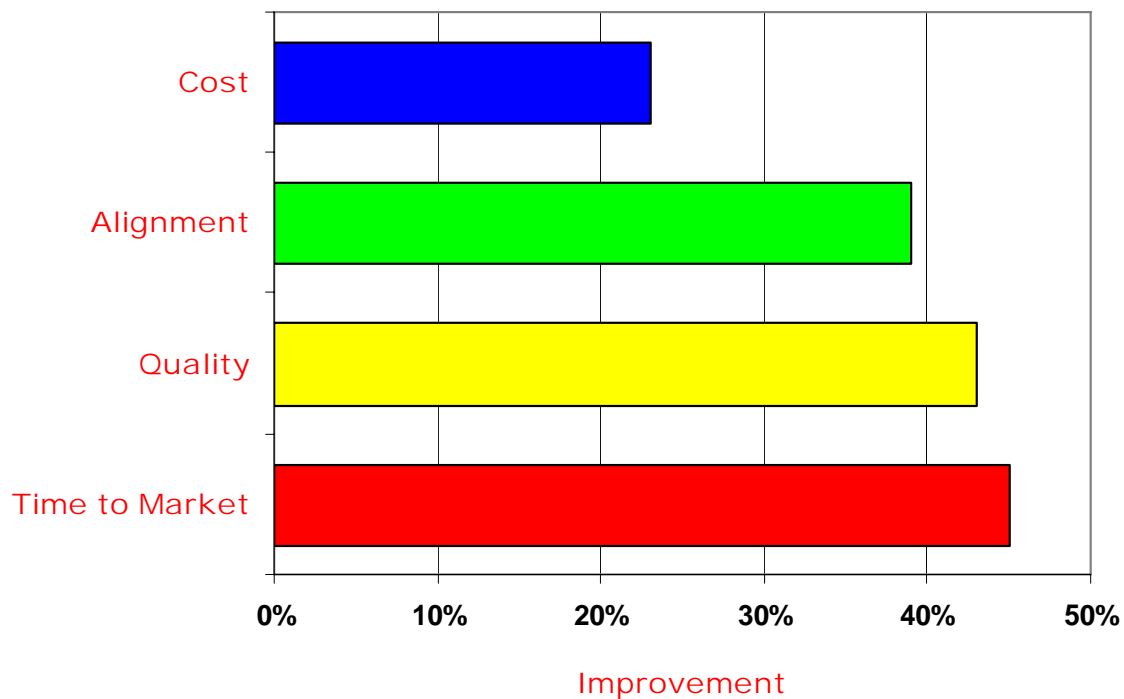
- Survey of 131 international respondents
- Extreme Programming (58%) and Scrum (8%)
- 85% of respondents were experts in Agile Methods



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

Agile Journal

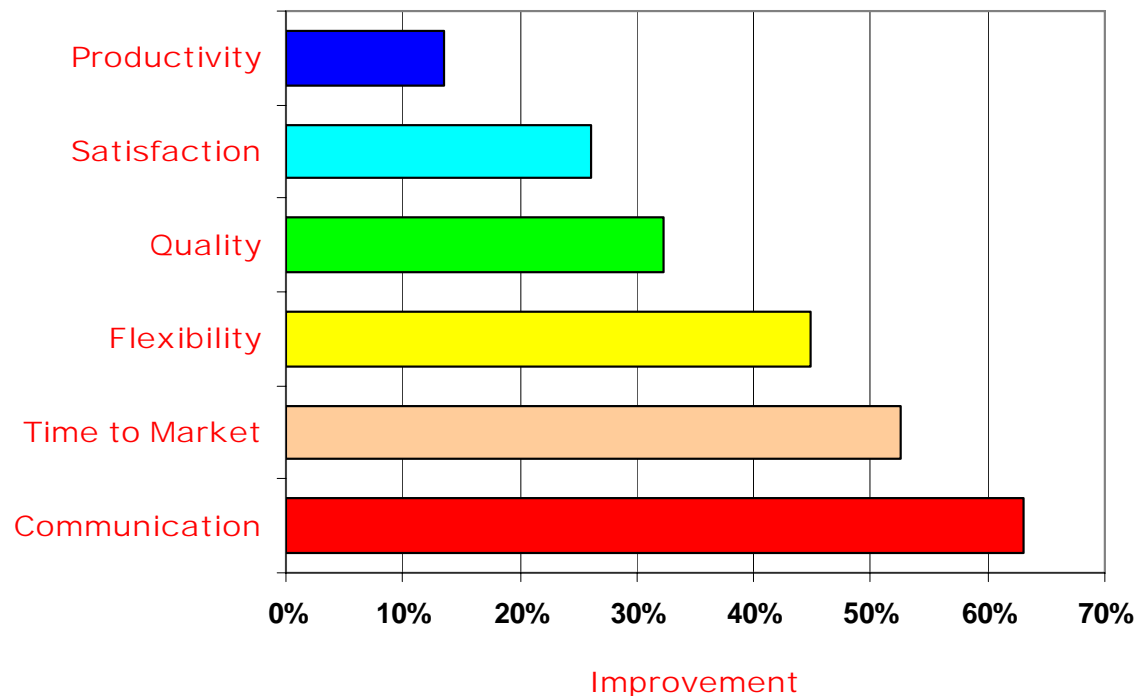
- Survey of 400 international respondents
- Extreme programming (28%) and Scrum (20%)
- 80% using Agile Methods to deliver maximum value



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

Microsoft

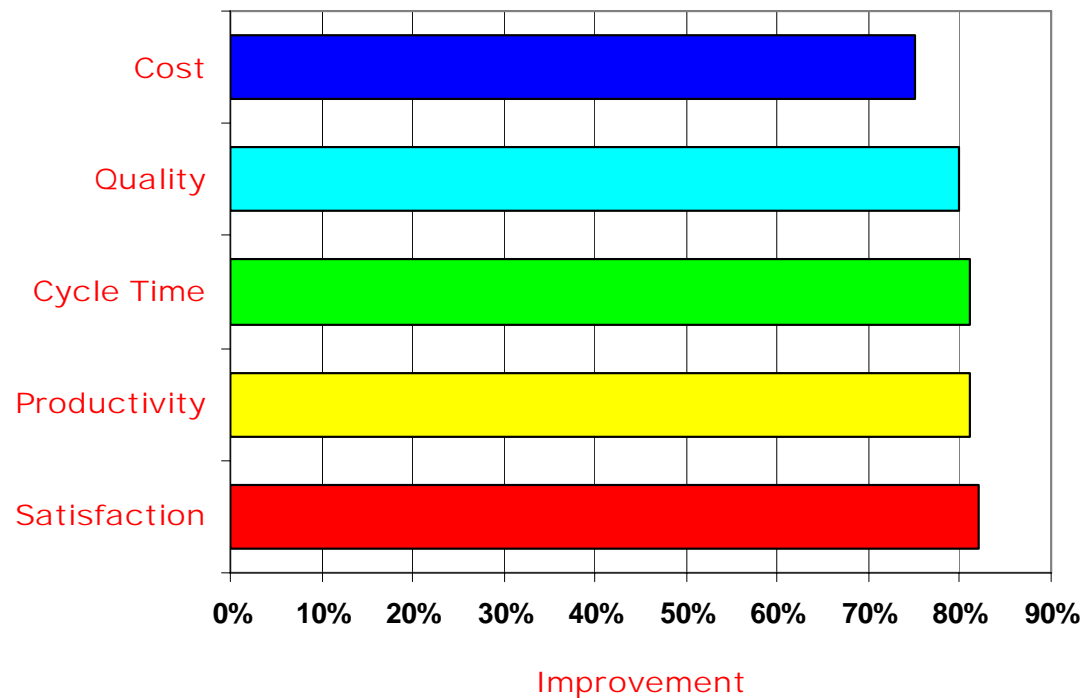
- Survey of 492 Microsoft respondents
- Scrum (65%) and Extreme Programming (5%)
- 65% using Agile Methods in virtual distributed teams



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

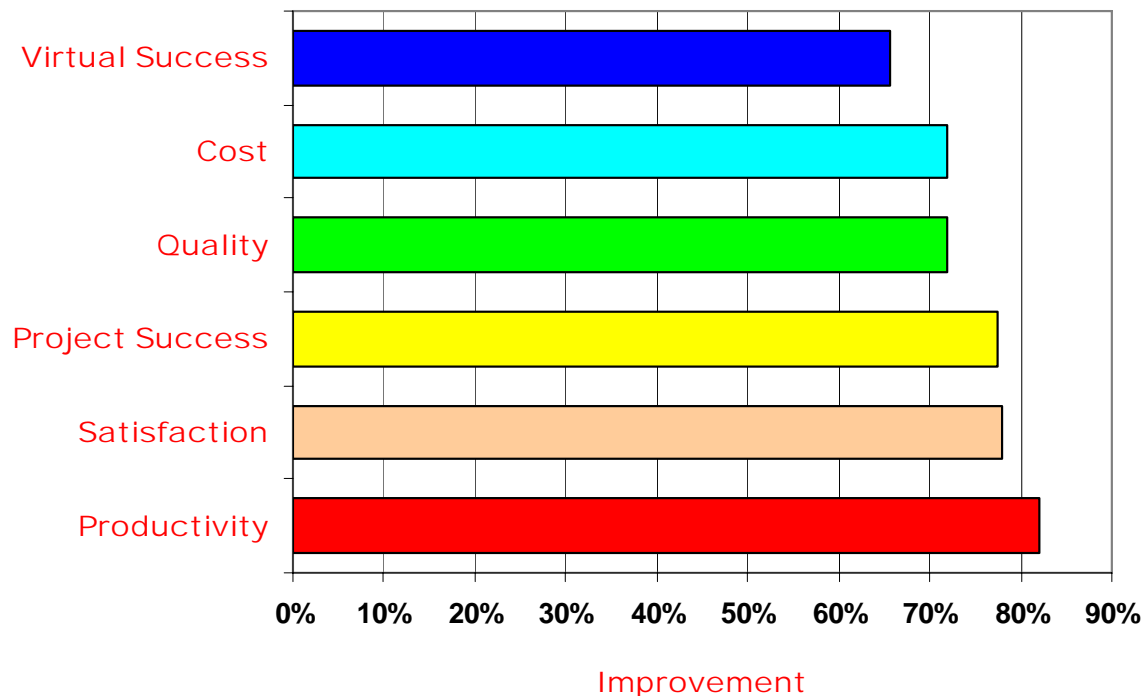
- Survey of 250 international respondents
- 70% of respondents using Agile Methods
- 83% of were from small-to-medium sized firms



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.

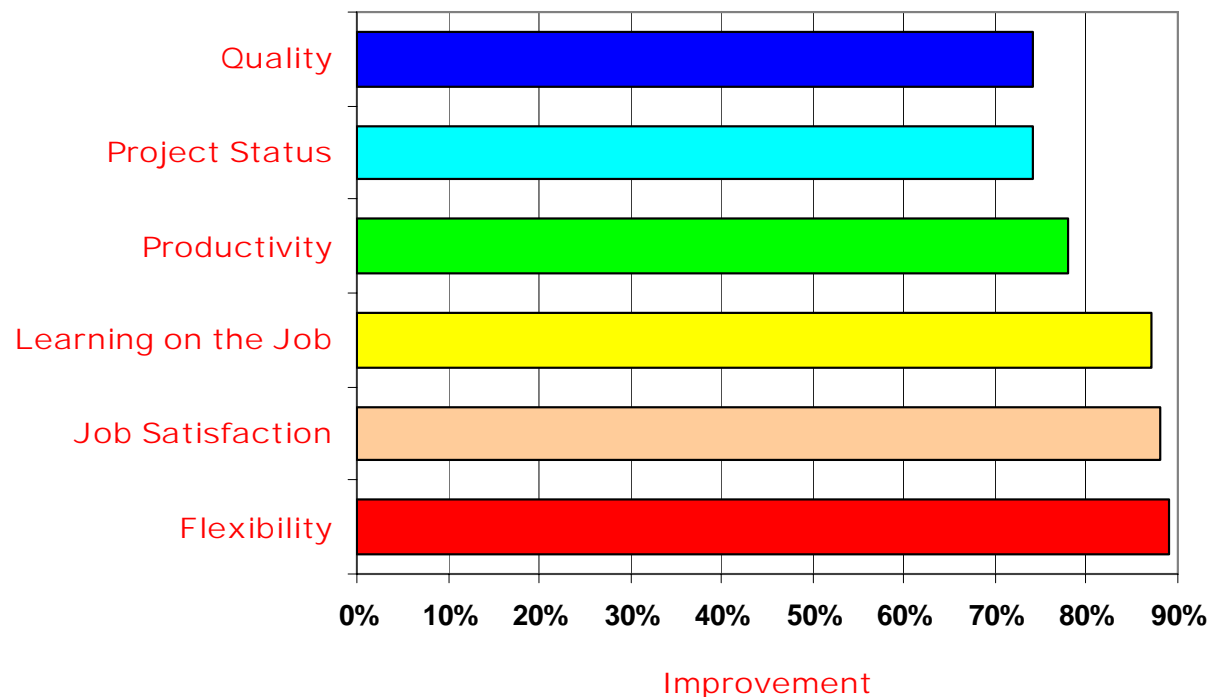
AmbySoft

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



IT Agile

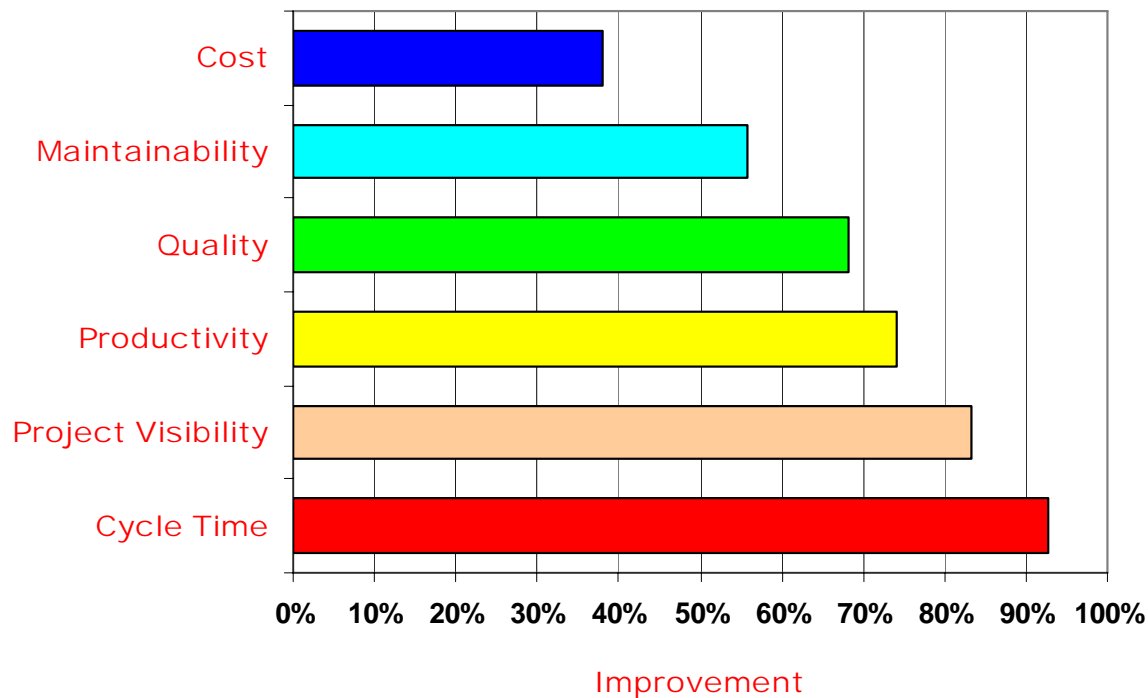
- Survey of 207 respondents in Germany
- Scrum (21%) and Extreme Programming (14%)
- 97% of respondents are satisfied with Agile Methods



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

Version One

- Survey of 3,061 respondents from 80 countries
- Scrum (49%), Scrum/XP (22%), and XP (8%)
- 68% from small firms and 57% distributed



Version One. (2008). *The state of agile development: Third Annual Survey*. Alpharetta, GA: Author.



Agenda

Background

Examples

Deliverables

Surveys

 **Business Value**

Other Considerations

Conclusion

References

ROI Metrics for Agile Methods

- A major principle of Agile Methods is creating value
- ROI is the measure of value within Agile Methods
- Costs and benefits are the basic inputs to ROI

ROI Metric	ROI Formula
Costs	$\sum_{i=1}^n Cost_i$
Benefits	$\sum_{i=1}^n Benefit_i$
Benefit to Cost Ratio (B/CR)	$\frac{Benefits}{Costs}$
Return on Investment (ROI)	$\frac{Benefits - Costs}{Costs} \times 100\%$
Net Present Value (NPV)	$\sum_{i=1}^{Years} \frac{Benefits_i}{(1 + Discount\ Rate)^{Years}} - Costs_0$
Break Even Point (BEP)	$\frac{Costs}{NPV} \times 60\ Months$
Real Options Analysis (ROA)	$N(d_1) \times Benefits - N(d_2) \times Costs \times e^{-Rate \times Years}$

$$d1 = [\ln(Benefits \div Costs) + (Rate + 0.5 \times Risk^2) \times Years] \div Risk \times \sqrt{Years}, \quad d2 = d1 - Risk \times \sqrt{Years}$$

Studies of Agile Methods

- Based on a recent study of Agile Methods
- Represents 109 data points from 69 studies
- Agile is 459% better than Traditional Methods

Agile Methods

Category	Low	Median	High
Cost	10%	26%	70%
Schedule	11%	71%	700%
Productivity	14%	122%	712%
Quality	10%	70%	1,000%
Satisfaction	70%	70%	70%
ROI	240%	2,633%	8,852%



Traditional Methods

Category	Low	Median	High
Cost	3%	20%	87%
Schedule	2%	37%	90%
Productivity	9%	62%	255%
Quality	7%	50%	132%
Satisfaction	-4%	14%	55%
ROI	200%	470%	2,770%

Costs of Agile Methods

- Represents 47 data points from 29 studies
- Based on average productivity and quality data
- Better quality is related to lower total lifecycle costs

Agile Methods — Total Lifecycle Cost Models

Method	LOC	Development	Hours	Maintenance	Hours	Rate	Total Cost
XP	10,000	$\text{LOC} \div 16.1575$	619	$0.7466 \times \text{KLOC} \times 100$	747	\$100	\$136,548
TDD	10,000	$\text{LOC} \div 29.2800$	342	$2.1550 \times \text{KLOC} \times 100$	2,155	\$100	\$249,653
PP	10,000	$\text{LOC} \div 33.4044$	299	$2.3550 \times \text{KLOC} \times 100$	2,355	\$100	\$265,437
Scrum	10,000	$\text{LOC} \div 05.4436$	1,837	$3.9450 \times \text{KLOC} \times 100$	3,945	\$100	\$578,202
Average	10,000	$\text{LOC} \div 21.2374$	471	$1.7972 \times \text{KLOC} \times 100$	1,797	\$100	\$226,805

Benefits of Agile Methods

- Traditional costs based on quality and productivity
- Test benefits are subtracted from traditional cost
- Agile costs are subtracted from traditional costs

Agile Methods — Total Lifecycle Benefit Models

Method	LOC	Traditional Methods	Trad. Cost	Agile Cost	Benefits
XP	10,000	$(\text{LOC} \times 10.51 - 6,666.67 \times 9) \times 100$	\$4,509,997	\$136,548	\$4,373,449
TDD	10,000	$(\text{LOC} \times 10.51 - 6,666.67 \times 9) \times 100$	\$4,509,997	\$249,653	\$4,260,344
PP	10,000	$(\text{LOC} \times 10.51 - 6,666.67 \times 9) \times 100$	\$4,509,997	\$265,437	\$4,244,560
Scrum	10,000	$(\text{LOC} \times 10.51 - 6,666.67 \times 9) \times 100$	\$4,509,997	\$578,202	\$3,931,795
Average	10,000	$(\text{LOC} \times 10.51 - 6,666.67 \times 9) \times 100$	\$4,509,997	\$226,805	\$4,283,192

ROI of Agile Methods

- Costs and benefits were input to ROI metrics
- Agile Methods were ranked according their ROI
- Agile Methods with higher quality had higher ROI

Method	Costs	Benefits	B/CR	ROI	NPV	BEP	ROA
XP	\$136,548	\$4,373,449	32:1	3,103%	\$3,650,401	\$4,263	\$4,267,105
TDD	\$249,653	\$4,260,344	17:1	1,607%	\$3,439,359	\$14,629	\$4,074,506
PP	\$265,437	\$4,244,560	16:1	1,499%	\$3,409,908	\$16,599	\$4,050,918
Scrum	\$578,202	\$3,931,795	7:1	580%	\$2,826,320	\$85,029	\$3,660,805
Average	\$226,805	\$4,283,192	19:1	1,788%	\$3,481,992	\$12,010	\$4,110,308

Rico, D. F. (2008). *What is the ROI of agile vs. traditional methods?* Retrieved September 3, 2008, from <http://davidfrico.com/rico08b.pdf>

ROI of Agile vs. Traditional

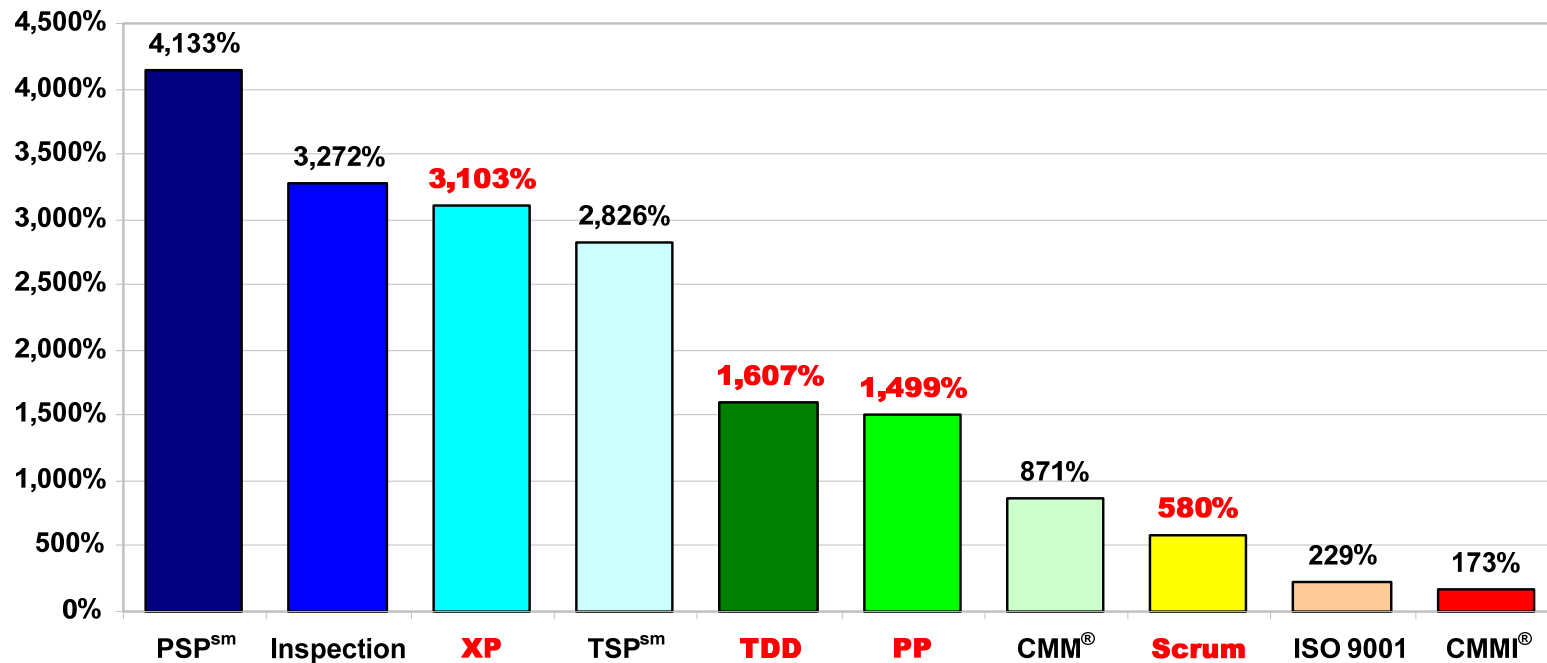
- Traditional Methods data was used for comparison
- All methods were ranked according to their ROI
- Methods with higher quality had higher ROI

Type	Method	Costs	Benefits	B/CR	ROI	NPV	BEP	ROA
Traditional	PSP sm	\$105,600	\$4,469,997	42:1	4,133%	\$3,764,950	\$945	\$4,387,756
Traditional	Inspection	\$82,073	\$2,767,464	34:1	3,272%	\$2,314,261	\$51,677	\$2,703,545
Agile	XP	\$136,548	\$4,373,449	32:1	3,103%	\$3,650,401	\$4,263	\$4,267,105
Traditional	TSP sm	\$148,400	\$4,341,496	29:1	2,826%	\$3,610,882	\$5,760	\$4,225,923
Agile	TDD	\$249,653	\$4,260,344	17:1	1,607%	\$3,439,359	\$14,629	\$4,073,167
Agile	PP	\$265,437	\$4,244,560	16:1	1,499%	\$3,409,908	\$16,599	\$4,048,404
Traditional	SW-CMM [®]	\$311,433	\$3,023,064	10:1	871%	\$2,306,224	\$153,182	\$2,828,802
Agile	Scrum	\$578,202	\$3,931,795	7:1	580%	\$2,826,320	\$85,029	\$3,622,271
Traditional	ISO 9001	\$173,000	\$569,841	3:1	229%	\$320,423	\$1,196,206	\$503,345
Traditional	CMMI [®]	\$1,108,233	\$3,023,064	3:1	173%	\$1,509,424	\$545,099	\$2,633,052

Rico, D. F. (2008). *What is the ROI of agile vs. traditional methods?* Retrieved September 3, 2008, from <http://davidfrico.com/rico08b.pdf>

ROI of Individual Methods

- Data for all methods was used for comparison
- Best Agile and Traditional Methods had top ROI
- Agile Methods better than big Traditional Methods



Rico, D. F. (2008). *What is the ROI of agile vs. traditional methods?* Retrieved September 3, 2008, from <http://davidfrico.com/rico08b.pdf>



Agenda

Background

Examples

Deliverables

Surveys

Business Value

 **Other Considerations**

Conclusion

References

Strengths & Weaknesses

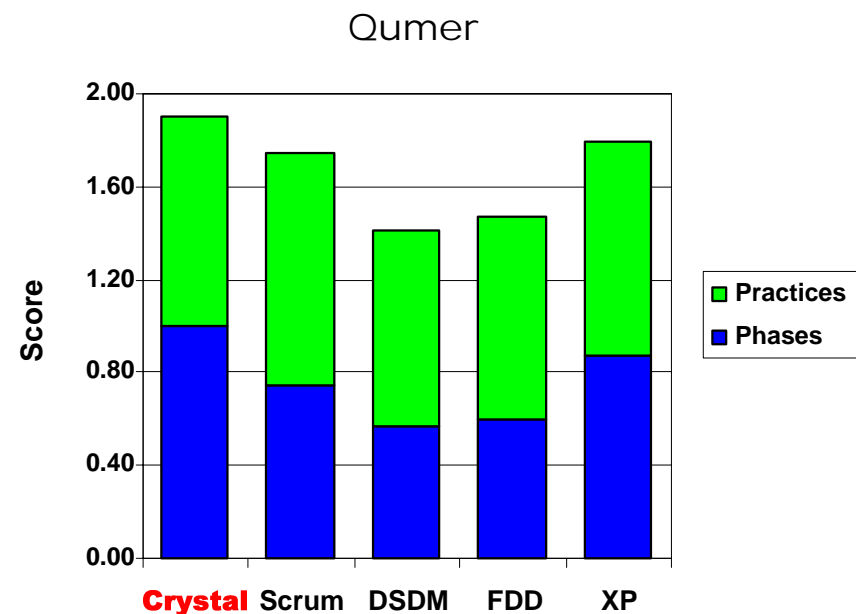
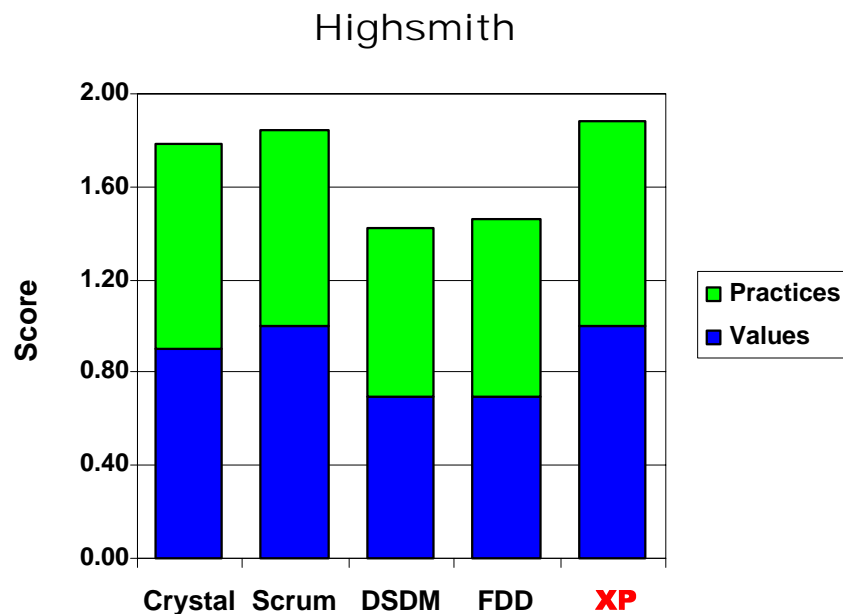
- Some follow the Agile Manifesto better than others
- Some have more process and document formality
- Often mistakenly compared to traditional methods

Method	Strengths	Weaknesses
Crystal	<ul style="list-style-type: none">• Scalable methodology• Support for safety-critical systems• Scalable project team size• Emphasis on testing	<ul style="list-style-type: none">• Requires co-located teams• Backward and forward compatibility• Non-real time scalability
Scrum	<ul style="list-style-type: none">• Self organizing teams• Customer participation• Focus on business value• Certification process	<ul style="list-style-type: none">• No sub-disciplines• No technical practices• Feature prioritization
DSDM	<ul style="list-style-type: none">• Emphasis on testing• Business focused• Prioritization of requirements• Sets stakeholder expectations early	<ul style="list-style-type: none">• Most heavyweight approach• Continuous user involvement• Heavy documentation• Proprietary approach
FDD	<ul style="list-style-type: none">• Support for parallel teams• Product feature focused• Easy to adopt• Scales to large teams or projects	<ul style="list-style-type: none">• Promotes individual code ownership• Release planning not defined• Incompatible with other approaches
XP	<ul style="list-style-type: none">• Technical practices• Customer ownership• Frequent feedback• Widely known	<ul style="list-style-type: none">• Onsite customer• Informal documentation• Little or no architecture

Coffin, R., & Lane, D. (2006). *A practical guide to seven agile methodologies: Part 2*. Retrieved September 3, 2008, from <http://www.devx.com/architect/Article/32836/1954>

Degree of Agility

- Agile Manifesto is a great way to measure agility
- Some do have a high degree of process rigidity
- These tend to be more of a popularity contest

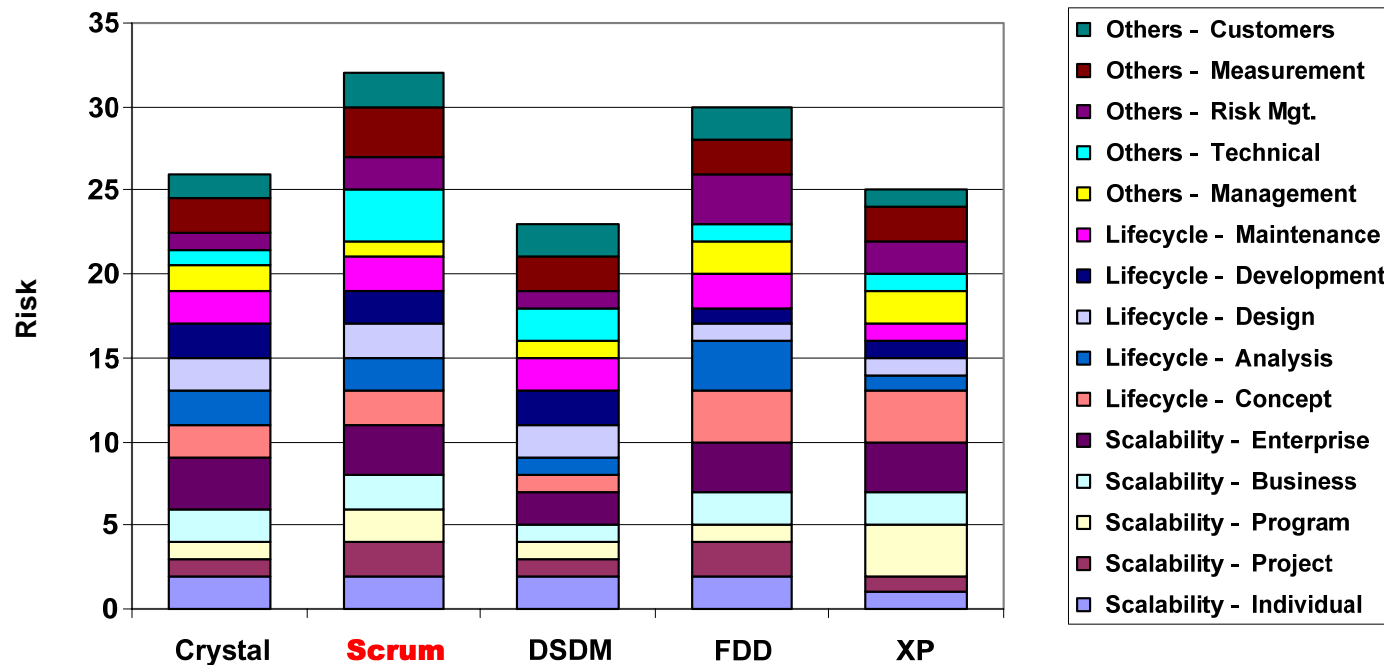


Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley.

Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods. *IS&T*, 50(4), 280-295.

Degree of Risk

- Agile Manifesto should be used to measure risk
- Risk is often measured using Traditional Methods
- Some traditional factors may be considered (not all)



Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Boston, MA: Addison-Wesley.

Common Mistakes

- ❑ Laissez-faire attitude to Agile Methods is a mistake
- ❑ Agile Methods require a measure of commitment
- ❑ Involve resources, training, and compliance

No.	Common Mistakes
1.	Thinking that Agile means "no documentation" and "cowboy coding"
2.	Thinking that you can piecemeal Agile practices and gain all the benefits
3.	Thinking Agile stops at engineering teams and won't affect the rest of the organization
4.	Not having a champion
5.	Having the wrong people lead the effort and/or the teams
6.	Hanging on to the death march as a solution
7.	Allowing the team to say "you'll get it when you get it"
8.	Assuming you're Agile and only planning one iteration at a time
9.	Allowing the Agile team leader to say, "you figure it out"
10.	Lack of participation by the business
11.	Not bothering with the retrospective
12.	A values mismatch

Sliger, M., & Broderick, S. (2008). *The software project manager's bridge to agility*. Boston, MA: Addison-Wesley.

Critical Success Factors

- Agile Methods-specific studies starting to emerge
- New studies focusing on values of Agile Manifesto
- Training, adherence, culture, and leadership are key

Category	Critical Success Factors	
Delivery	<ul style="list-style-type: none"> • Regular delivery of software 	<ul style="list-style-type: none"> • Delivering important features first
Technical	<ul style="list-style-type: none"> • Well-defined coding standards • Pursuing simple design • Rigorous refactoring activities 	<ul style="list-style-type: none"> • Right amount of documentation • Correct integration testing
Personnel	<ul style="list-style-type: none"> • Formal training in Agile Methods • High competence and expertise • Great motivation 	<ul style="list-style-type: none"> • Managers knowledgeable in agile • Adaptive management style • Appropriate technical training
Management	<ul style="list-style-type: none"> • Agile requirements management • Agile project management • Agile configuration management 	<ul style="list-style-type: none"> • Good progress tracking mechanism • Strong daily communication • Honoring regular working schedule
Teamwork	<ul style="list-style-type: none"> • Coherent self-organizing teams • Collocation of the whole team 	<ul style="list-style-type: none"> • Projects with small team • No multiple independent teams
Customers	<ul style="list-style-type: none"> • Good customer relationship • Strong customer commitment 	<ul style="list-style-type: none"> • Customer having full authority

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6), 961-971.

Project Management

- Project management differs for Agile Methods
- Focuses on enhancing the performance of teams
- Agile project management is related to agile values

Principle	Practice	Leadership	Management
Foster Alignment and Cooperation	Organic teams	<ul style="list-style-type: none"> • Promote software craftsmanship • Foster team collaboration • Form a guiding coalition • Cultivate informal communities of practice 	<ul style="list-style-type: none"> • Identify the project community • Design a holographic formal structure • Get self-disciplined team players • Propose an adaptive IT enterprise
	Guiding vision	<ul style="list-style-type: none"> • Evolve a team vision • Align the team • Envision a bold future • Create and maintain shared expectations 	<ul style="list-style-type: none"> • Discover business outcomes • Clearly delineate scope • Estimate level of effort • Design a vision box and elevator statement
Encourage Emergence and Self Organization	Simple rules	<ul style="list-style-type: none"> • Enlist the team for change • Focus on business value 	<ul style="list-style-type: none"> • Assess the status quo and tailor method • Develop a release/iteration plan/backlog • Facilitate design, code, test, and deployment • Conduct testing and manage release
	Open information	<ul style="list-style-type: none"> • Conduct a standup meeting daily • Encourage feedback • Build trust • Link language with action 	<ul style="list-style-type: none"> • Promote good internal and external com. • Negotiate a customer representative on-site • Encourage the use of information radiators • Map the project's value stream
	Light touch	<ul style="list-style-type: none"> • Fit your style to the situation • Support roving leadership • Go with the flow and maintain quality of work life • Build on personal strengths and commitments 	<ul style="list-style-type: none"> • Decentralize control • Establish a pull task management system • Manage the flow • Use action sprints
Institute Leadership and Adaptation	Adaptive leadership	<ul style="list-style-type: none"> • Cultivate an embodied presence • Practice embodied learning 	<ul style="list-style-type: none"> • Get plus-delta feedback daily • Monitor and adapt to simple rules/practices • Conduct regular project reflections • Conduct scenario planning

Augustine, S. (2005). *Managing agile projects*. Upper Saddle River, NJ: Prentice-Hall.

Adoption Framework

- Models exist for measuring degree of agile adoption
- Lowest levels focus on basic tools and techniques
- Highest level focus on advanced agile practices

No.	Level	Embrace Change	Frequent Delivery	Human Centricity	Technical Excellence	Customer Collaboration
5	Ambient	<ul style="list-style-type: none"> • Low ceremony 	<ul style="list-style-type: none"> • Agile estimation 	<ul style="list-style-type: none"> • Ideal physical setup 	<ul style="list-style-type: none"> • Test driven development • Pair programming • Top performers 	<ul style="list-style-type: none"> • Frequent interaction
4	Adapt	<ul style="list-style-type: none"> • Client-driven iteration • Continuous feedback 	<ul style="list-style-type: none"> • Frequent releases • Adaptive planning 		<ul style="list-style-type: none"> • Daily stand-ups • Agile documentation • User stories 	<ul style="list-style-type: none"> • Accessible customer • Customer contract
3	Effective		<ul style="list-style-type: none"> • Risk-driven iterations • Feature-driven • Feature-tracking 	<ul style="list-style-type: none"> • Self organizing teams • Collocated teams 	<ul style="list-style-type: none"> • Continuous integ. • Continuous improve. • Unit testing • Good performers 	
2	Evolve	<ul style="list-style-type: none"> • Evolutionary stories 	<ul style="list-style-type: none"> • Continuous delivery • Multi-level planning 		<ul style="list-style-type: none"> • Configuration mgt. • Iteration tracking • Evolutionary design 	<ul style="list-style-type: none"> • Evolutionary contract
1	Collaborate	<ul style="list-style-type: none"> • Process reflection 	<ul style="list-style-type: none"> • Collaborative planning 	<ul style="list-style-type: none"> • Collaborative teams • Empowered teams 	<ul style="list-style-type: none"> • Coding standards • Collaborative tools • Task volunteering 	<ul style="list-style-type: none"> • Committed customer

Sidky, A., Arthur, J., & Bohner, S. (2007). A disciplined approach to adopting agile practices: The agile adoption framework. *Innovations in Systems and Software Engineering*, 3(3), 203-216.



Agenda

Background

Examples

Deliverables

Surveys

Business Value

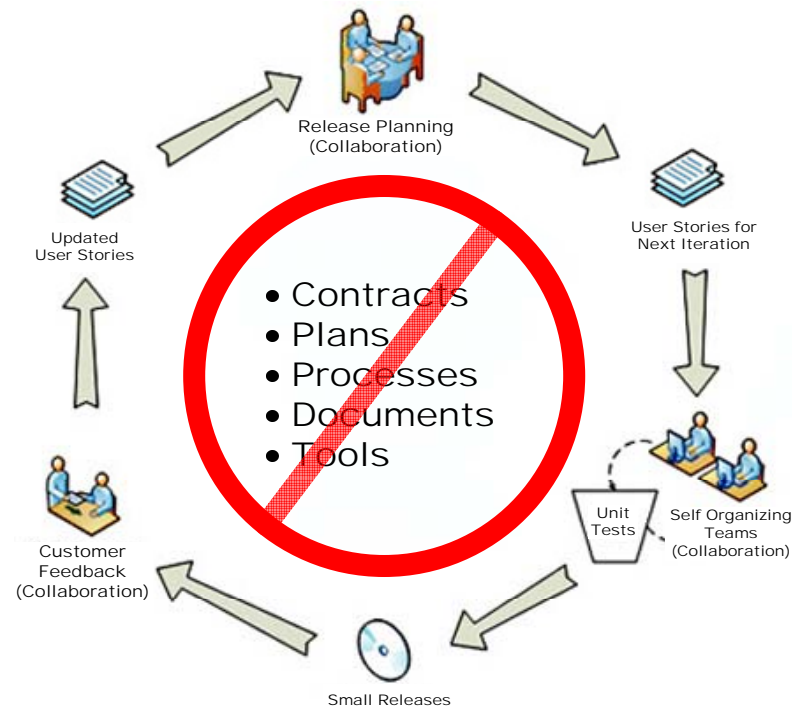
Other Considerations

 **Conclusion**

References

Conclusion

- Agile Methods are a fundamentally new paradigm
- Agile Methods are “not” lighter Traditional Methods
- They should not be viewed through a Traditional lens



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

Common Questions

- Many still have basic questions about Agile Methods
- Often about quality, documents, and maintenance
- Based on parochial views of Traditional Methods

No.	Subject	Question
1.	Quality	Do Agile Methods result in poor software quality?
2.	Documentation	Do Agile Methods lack proper software documentation?
3.	Maintenance	Do Agile Methods result in poor software maintainability?
4.	Requirements	Do Agile Methods lack proper software requirements?
5.	Customers	Do Agile Methods require on-site customers?
6.	Planning	Do Agile Methods lack proper long-term strategic plans?
7.	Virtual Teams	Do Agile Methods work with globally distributed teams?
8.	Implementation	Do Agile Methods have to be used in their entirety to be Agile?
9.	Customization	Do Agile Methods need to be mixed with other Agile Methods?
10.	Uniqueness	Do Agile Methods need to be mixed with Traditional Methods?

Rico, D. F. (2008). *Agile methods and software maintenance*. Retrieved October 20, 2008, from <http://davidfrico.com/rico08f.pdf>

Rico, D. F. (2008). *Agile methods and software documentation*. Retrieved October 20, 2008, from <http://davidfrico.com/rico08e.pdf>

Rico, D. F. (2008). *Agile methods and virtual distributed teams*. Retrieved October 20, 2008, from <http://davidfrico.com/rico08d.pdf>



Agenda

Background

Examples

Deliverables

Surveys

Business Value

Other Considerations

Conclusion

 **References**

References

- Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved September 3, 2008, from <http://www.agilemanifesto.org>
- Beck, K. (2001). *Extreme programming: Embrace change*. Upper Saddle River, NJ: Addison-Wesley.
- Beck, K. (2003). *Test-driven development: By example*. Boston, MA: Addison-Wesley.
- Beck, K., & Fowler, M. (2004). *Planning extreme programming*. Upper Saddle River, NJ: Addison-Wesley.
- Cohn, M. (2004). *User stories applied: For agile software development*. Boston, MA: Addison-Wesley.
- Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley.
- Schwaber, K., & Beedle, M. (2001). *Agile software development with scrum*. Upper Saddle River, NJ: Prentice-Hall.
- Schwaber, K. (2004). *Agile project management with scrum*. Redmond, WA: Microsoft Press.

Contact Information

- Website: <http://davidfrico.com>
- Biography: <http://www.linkedin.com/in/davidfrico>
- Capabilities: <http://davidfrico.com/rico-capability.pdf>

Dr. David F. Rico, PMP, CSM

Cost Estimates • Metrics • Project Plans
ROI • NPV • Breakeven Pt. • Real Options
Agile Methods • Six Sigma • CMMI • ISO 9001
System Lifecycles • Policies & Procedures • IDEF0

NASA • DARPA • DISA • Air Force • Navy • Army • MITI
Washington, DC • Tokyo, Japan • Silicon Valley

B.S. Computer Sci. • M.S. Software Eng. • D.M. Information Tech.
dave1@davidfrico.com • 410-924-5210 • <http://davidfrico.com>