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
- Business Value
- Other Considerations
- Conclusion
- References

Purpose

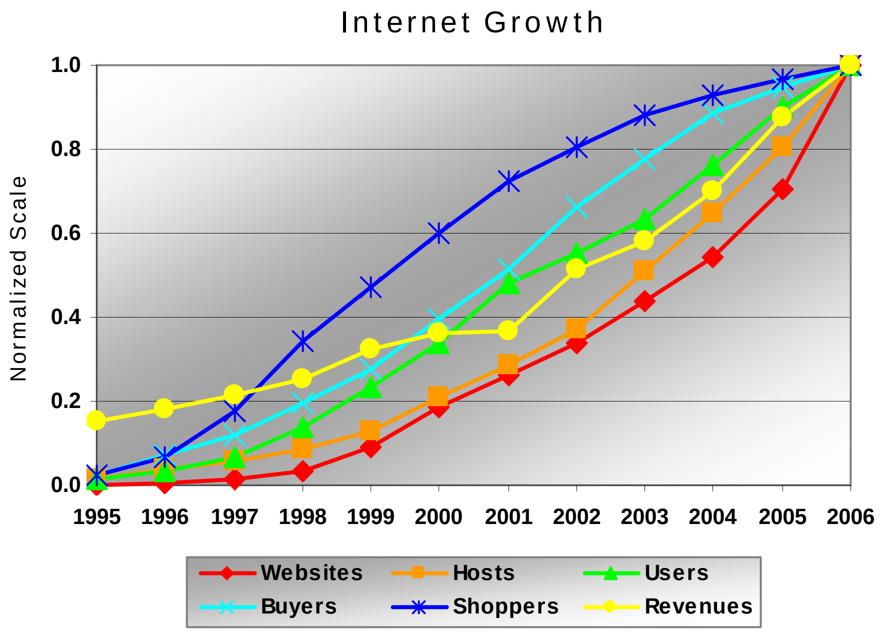
- Provide an overview of Agile Methods using examples, artifacts, benefits, and other data
  - *Agility is the ability to create and respond to change in order to profit in a turbulent business environment*
  - *Agility is prioritizing for maneuverability with respect to shifting requirements, technology, 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*

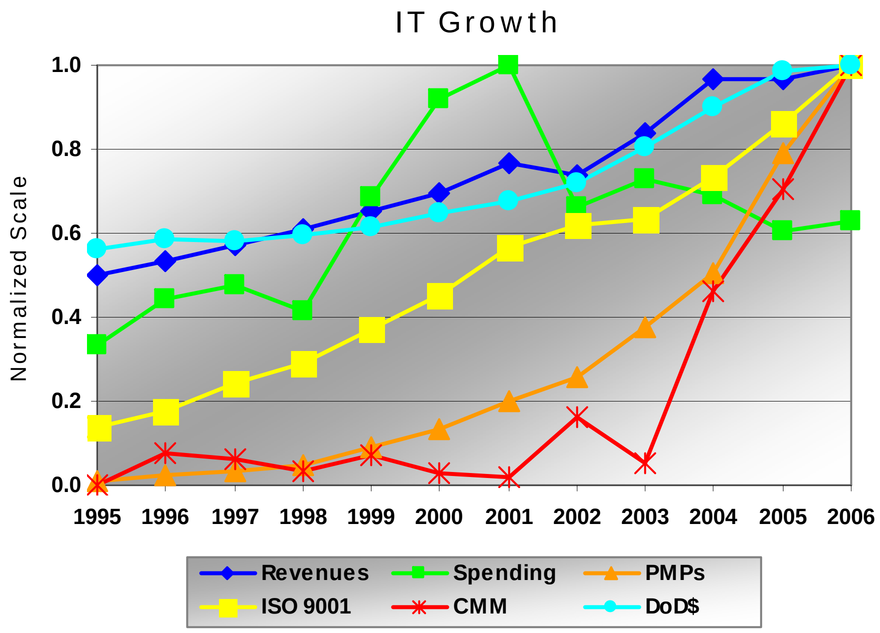
Key Terms

- 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.

IT Industry

- U.S. firms spent $700 billion on IT projects in 2006
- U.S. IT industry revenues reached $3 trillion in 2006
- U.S. used Agile Methods on 300,000 projects in 2006

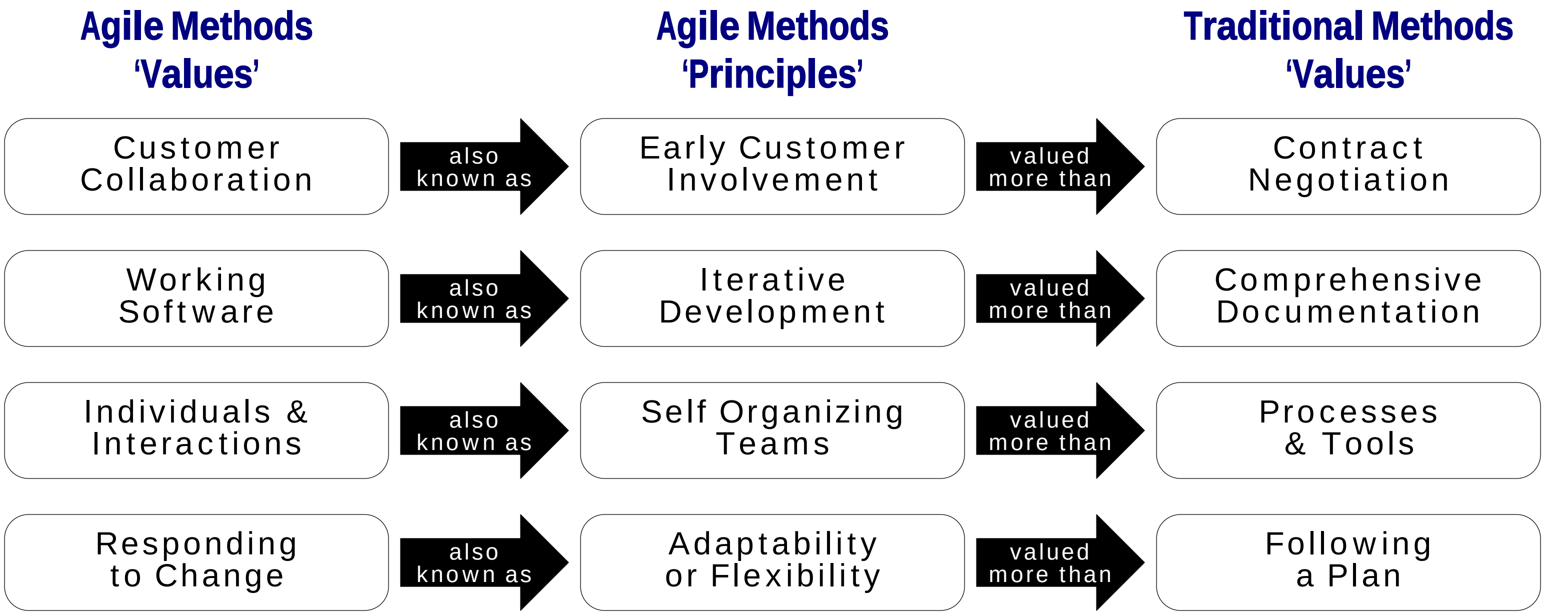




Rico, D. F. (2008). *Internet and information technology growth statistics*: *1995 to 2006*. Retrieved September 1, 2008, from http://davidfrico.com/it-stats.xls

What are Agile Methods?

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



Agile Methods ‘Values’		Agile Methods ‘Principles’		Traditional Methods ‘Values’
Customer Collaboration	also known as	Early Customer Involvement	valued more than	Contract Negotiation
Working Software	also known as	Iterative Development	valued more than	Comprehensive Documentation
Individuals & Interactions	also known as	Self Organizing Teams	valued more than	Processes & Tools
Responding to Change	also known as	Adaptability or Flexibility	valued more than	Following a Plan

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

Agile vs. Traditional Methods

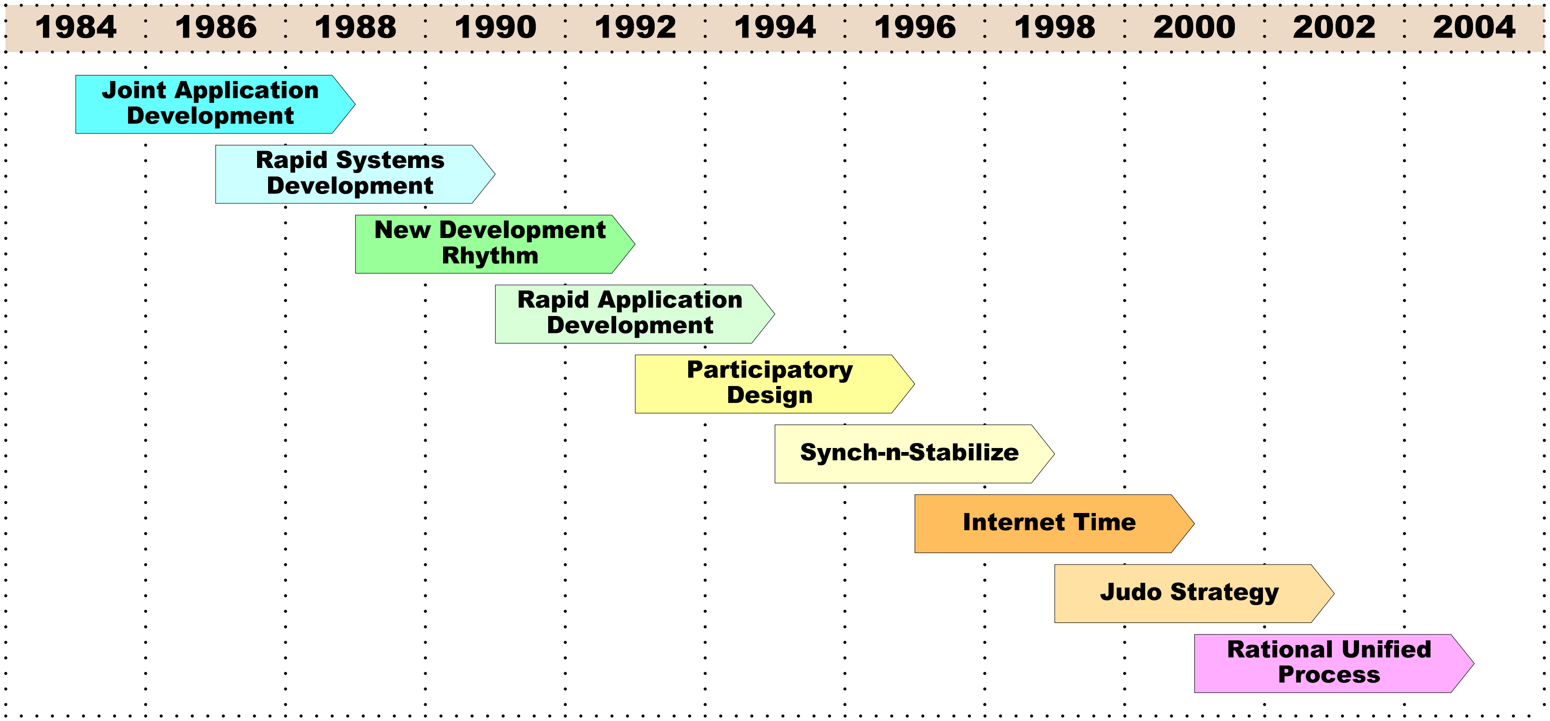
- Sloanism vs. Taylorism and Fordism
- Craft-industry vs. scientific management
- Personal vs. impersonal human interactions

Pine		Boehm		Nerur	
Agile	**Traditional**	**Agile**	**Traditional**	**Agile**	**Traditional**
• Job-shop oriented	• Manufacturing	• Unpredictable	• Predictable	• Unpredictable	• Predictable
• Mass customization	• Mass production	• Small projects	• Large projects	• People-centric	• Process-centric
• Micro-markets	• Mass market	• Turbulent	• Stability	• Egalitarian	• Authoritarian
• Sloanism	• Fordism	• Customer-centric	• Contract-centric	• Tacit knowledge	• Explicit knowledge
• Manufacturing cells	• Production lines	• Informal artifacts	• Formal documents	• Informal artifacts	• Formal documents
• Egalitarian	• Autocratic	• Tacit interaction	• Written interaction	• Multi-disciplinary	• Specialization
• Decentralized	• Centralized	• User stories	• Specifications	• Informal comm.	• Formal comm.
• Empowerment	• Division of labor	• Simple design	• Formal architecture	• Customer-critical	• Customer-important
• Multi-disciplinary	• Specialization	• Automated testing	• Formal test plans	• Feature-focused	• Activity-focused
• Collectivism	• Individualism	• Customer presence	• No customer	• Iterative process	• Linear process
• Economy of scope	• Economy of scale	• Developer-centric	• Analyst-centric	• Organic	• Mechanistic
• Value	• Cost	• Egalitarian	• Authoritarian	• Object-oriented	• Tech. agnostic
• Effectiveness	• Efficiency				
• Elegance	• Reliability				
• Capability-based	• Defect reduction				
• Customer satisfact.	• Quality control				

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

- Small well-structured multi-disciplinary team
- Adaptable processes and product technologies
- Customer feedback on working software releases

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
- Business Value
- Other Considerations
- Conclusion
- References

“Big 5” Agile Methods

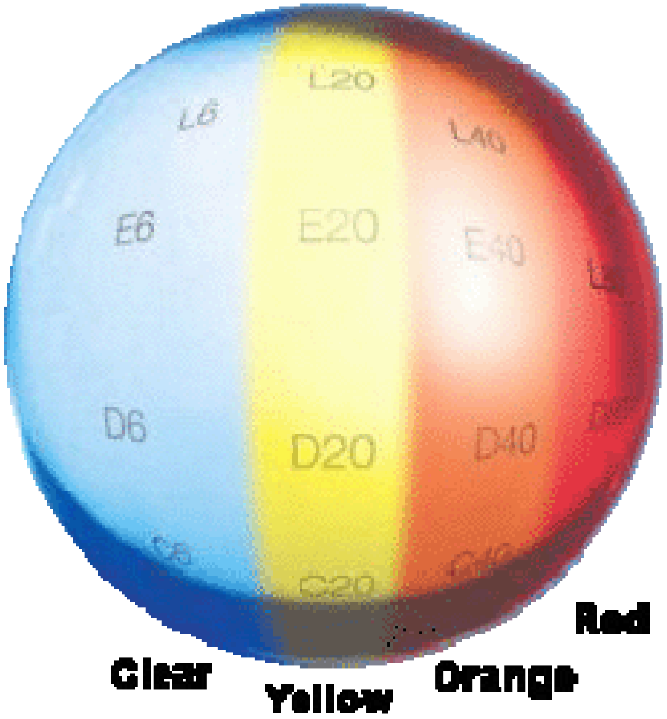
- The term “Agile Methods” was coined in 2001
- Extreme Programming was the first Agile Method
- Other Agile Methodologies came-to-light after 2001

Year	Method	Author	Firm	Process Elements	Major Features
1991	Crystal Clear	Cockburn	IBM	7 properties, 5 strategies, 7 stages, 9 tools, 22 artifacts	Use Cases, Domain Model
1993	Scrum	Sutherland	Easel	7 processes, 7 artifacts, 3 roles	Backlogs, Sprints, Daily Scrums
1993	Dynamic Systems Development	Millington	DSDM	9 principles, 5 stages, 15 tools, 12 roles, 23 artifacts	Iterations, Prototypes
1997	Feature-Driven Development	De Luca	Nebulon	5 processes, 8 practices, 14 roles, 29 tasks, 17 artifacts	Domain Model, Inspections
1999	Extreme Programming	Beck	Chrysler	Originally had 13 practices (now has 28 practices)	User Stories, Pair Programming, Tests

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

Crystal Methods

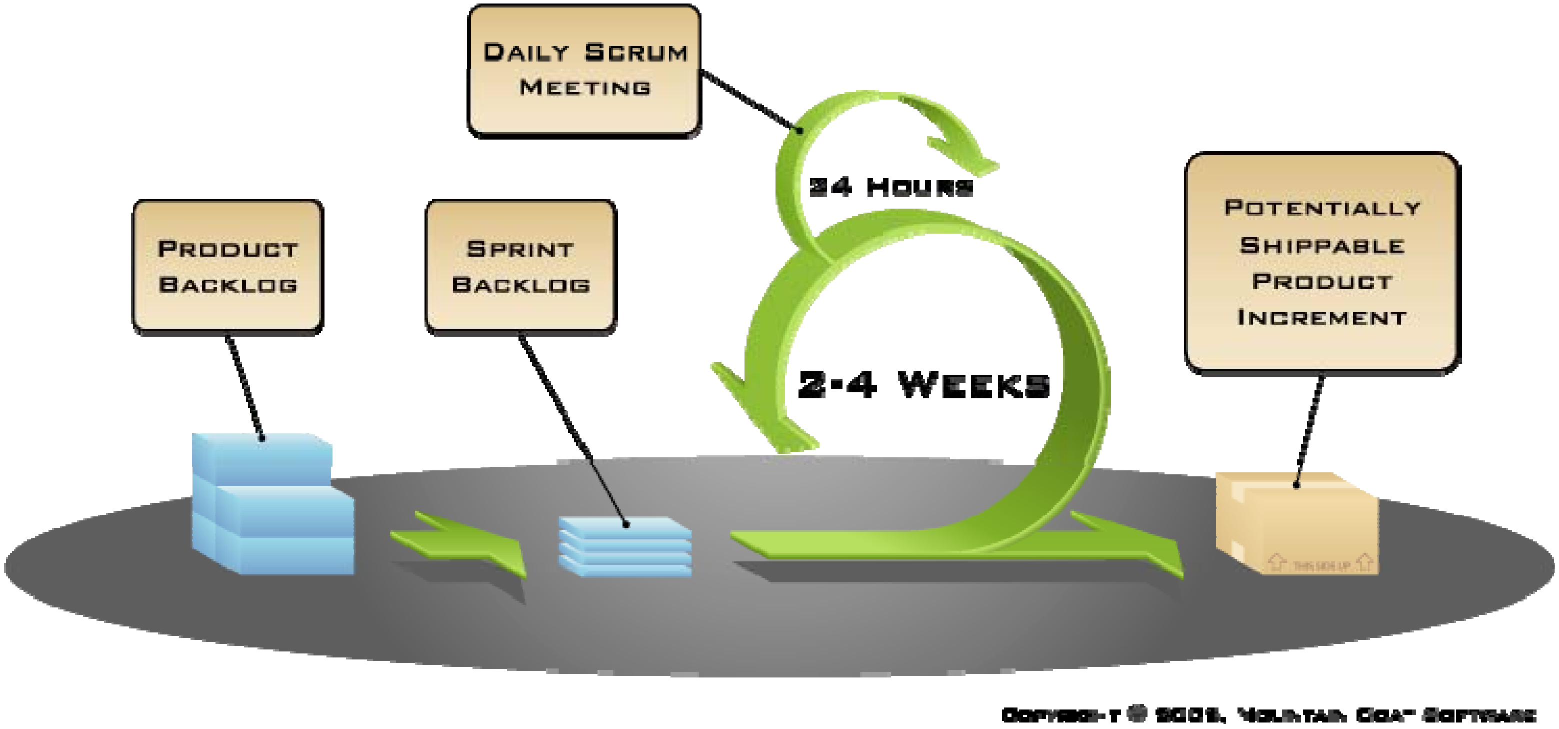
- Created by Alistair Cockburn in 1991
- Consists of 5 goals, 9 practices, and 8 roles
- Scalable family of techniques for critical systems



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

Scrum

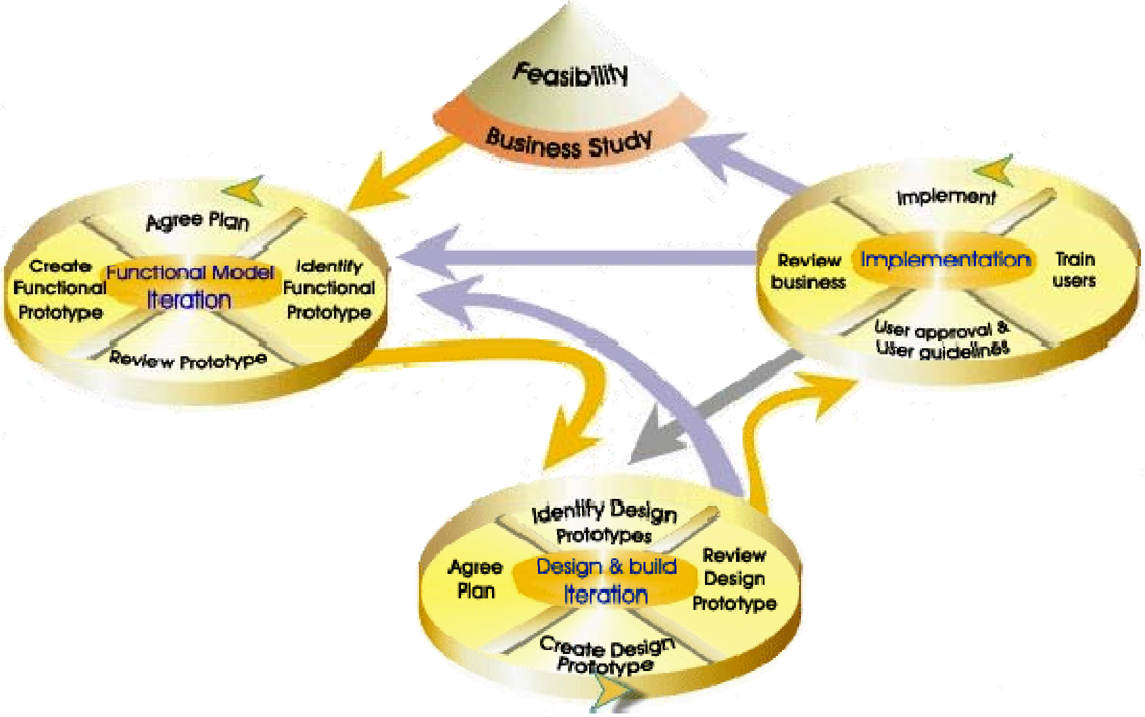
- Created by Jeff Sutherland at Easel in 1993
- Three basic phases—Planning, sprint, post-sprint
- Uses EVM to burn down backlog in 30-day iterations


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

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

Dynamic Systems Develop.

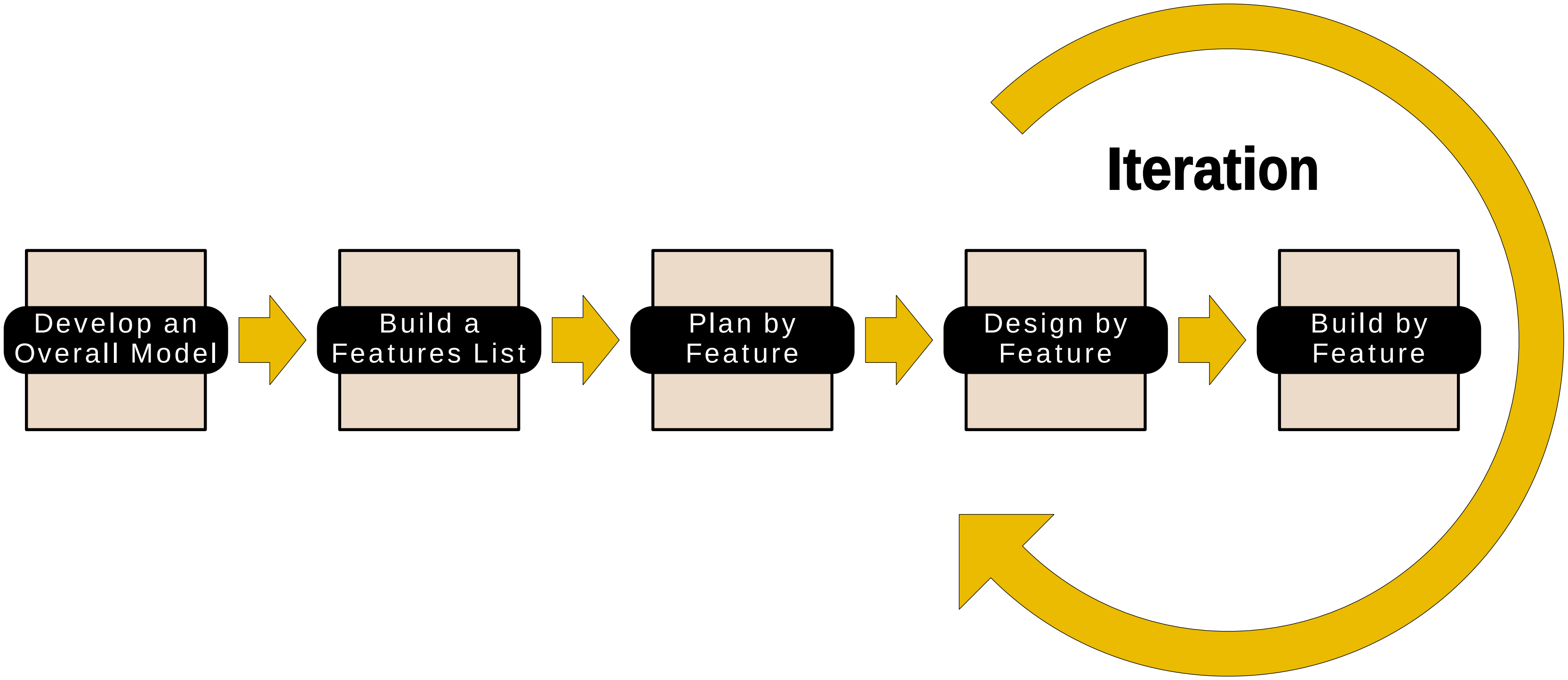
- Created by consortium of British firms in 1993
- Consists of 5 phases, 15 practices, and 12 roles
- Non-proprietary RAD approach from the early 1990s



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 Fagan inspections

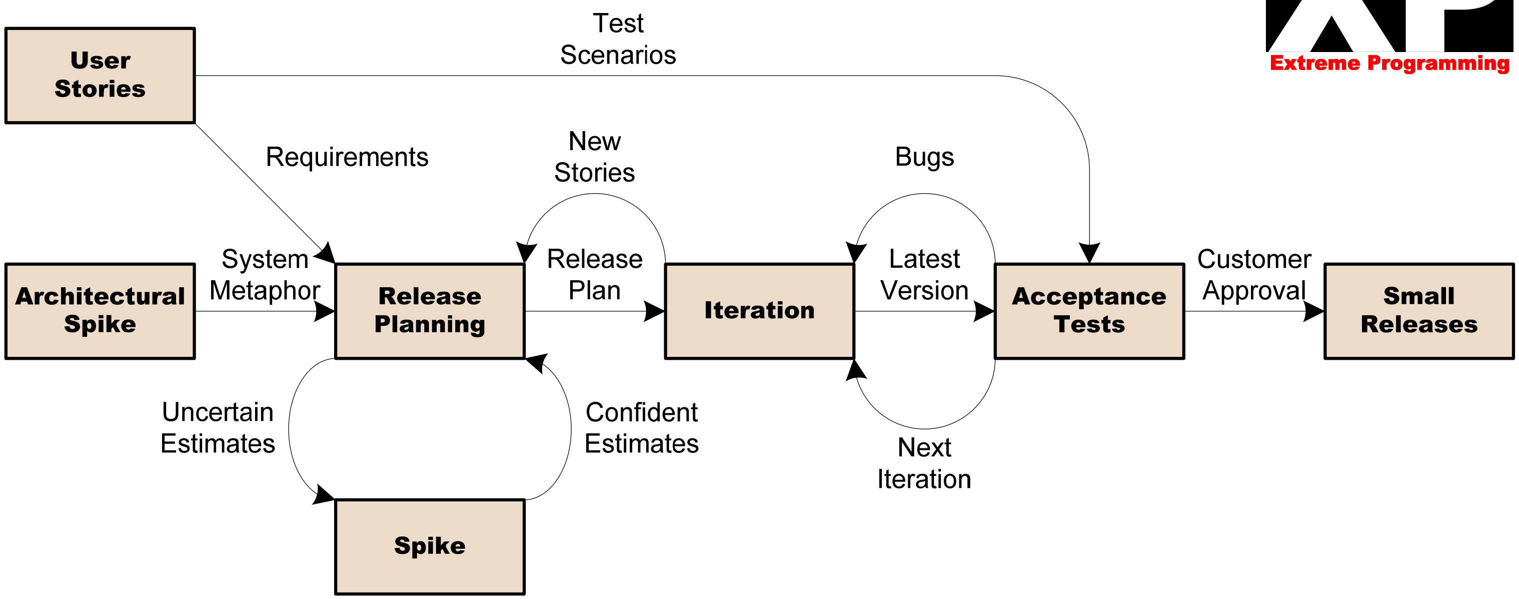


Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature driven development*. Upper Saddle River, NJ: Prentice-Hall.

Extreme Programming

- Created by Kent Beck at Chrysler in 1998
- Grown from 13 to more than 28 rules/practices
- Popularized pair programming and test-driven dev.

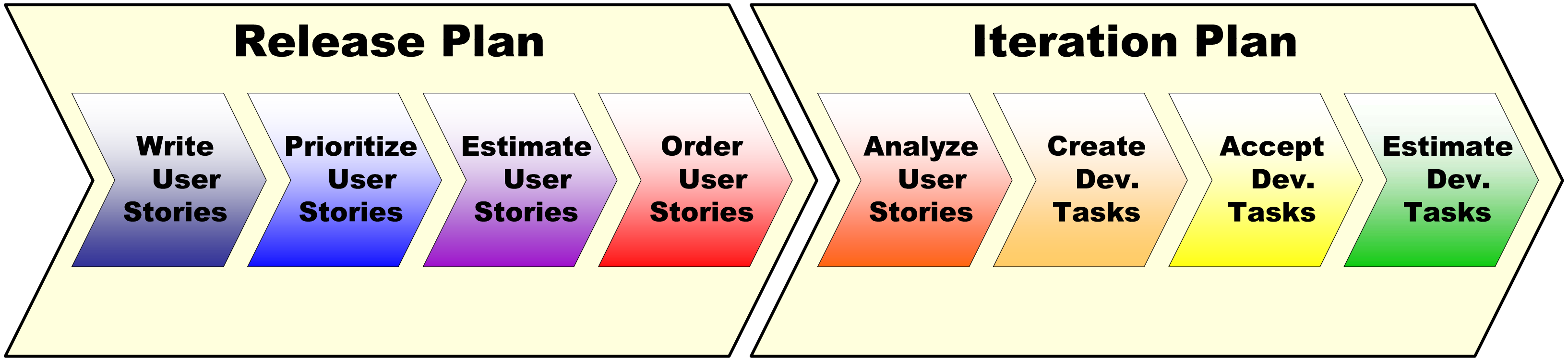
Extreme Programming



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

Release Planning

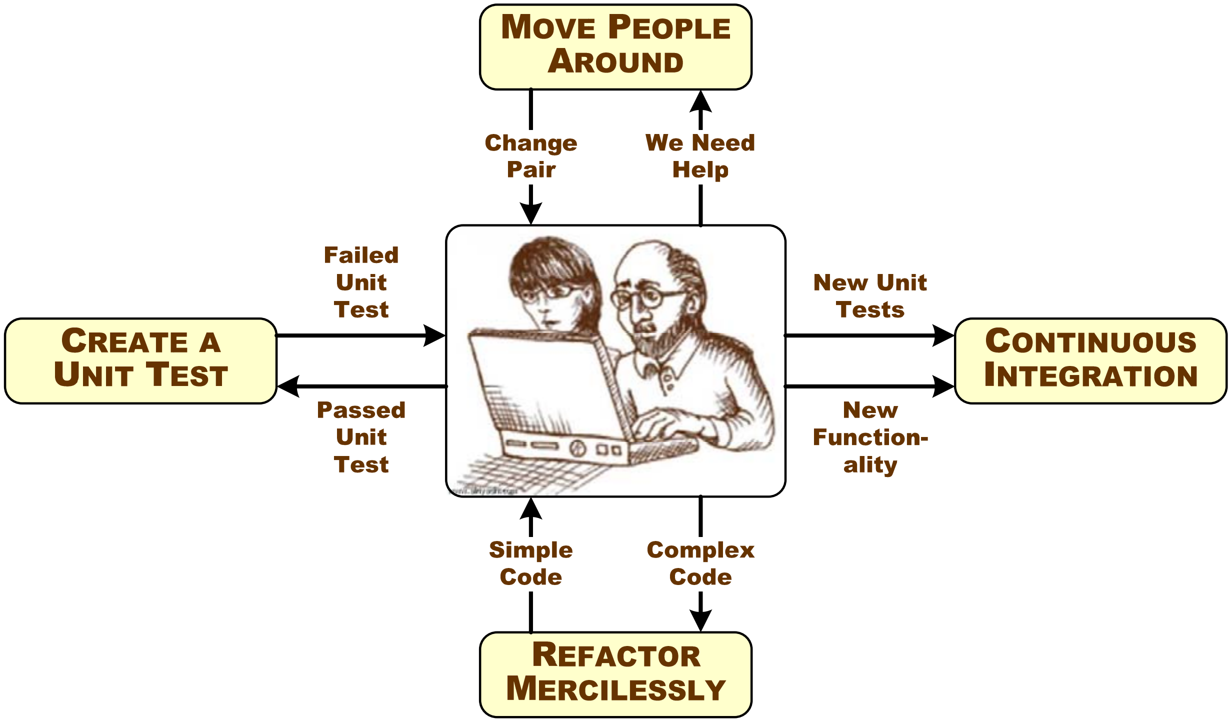
- Created by Kent Beck at Chrysler in 1998
- Consists of user stories and development tasks
- Used as project planning process for XP and Scrum



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

Pair Programming

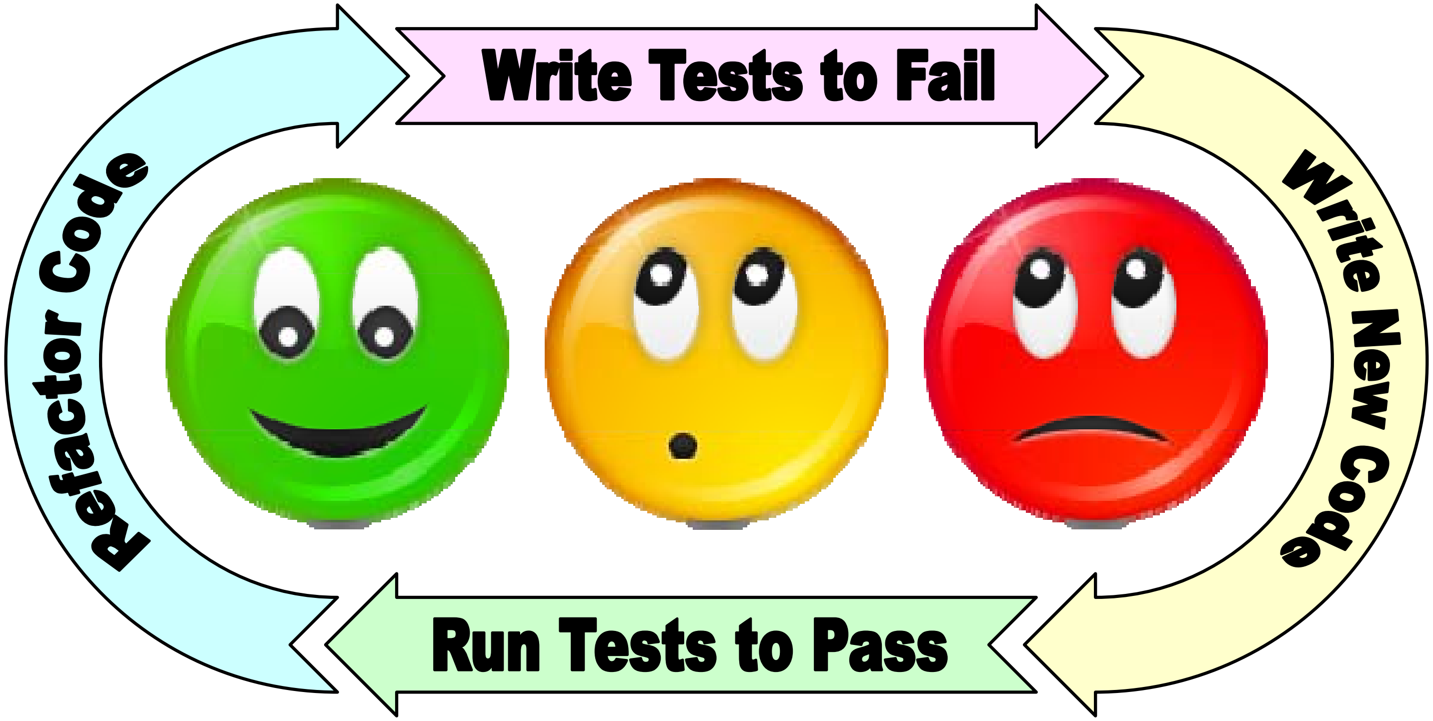
- Term coined by Jim Coplien in 1995
- Consists of two side-by-side programmers
- Considered an efficient problem solving technique



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

Test Driven Development

- Term coined by Kent Beck in 2003
- Consists of writing unit tests before coding
- Believed to be a primary means of quality control



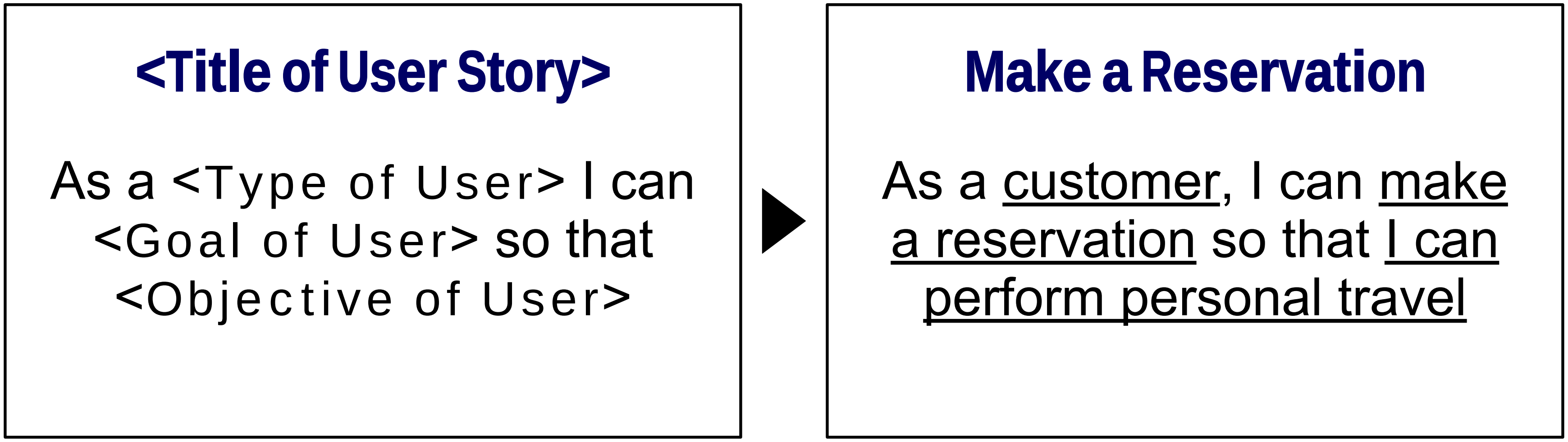
Beck, K. (2003). *Test-driven development: By example*. Boston, MA: Addison-Wesley.

Agenda

- Background
- Examples
- ☞ Deliverables
- Business Value
- Other Considerations
- Conclusion
- References

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>

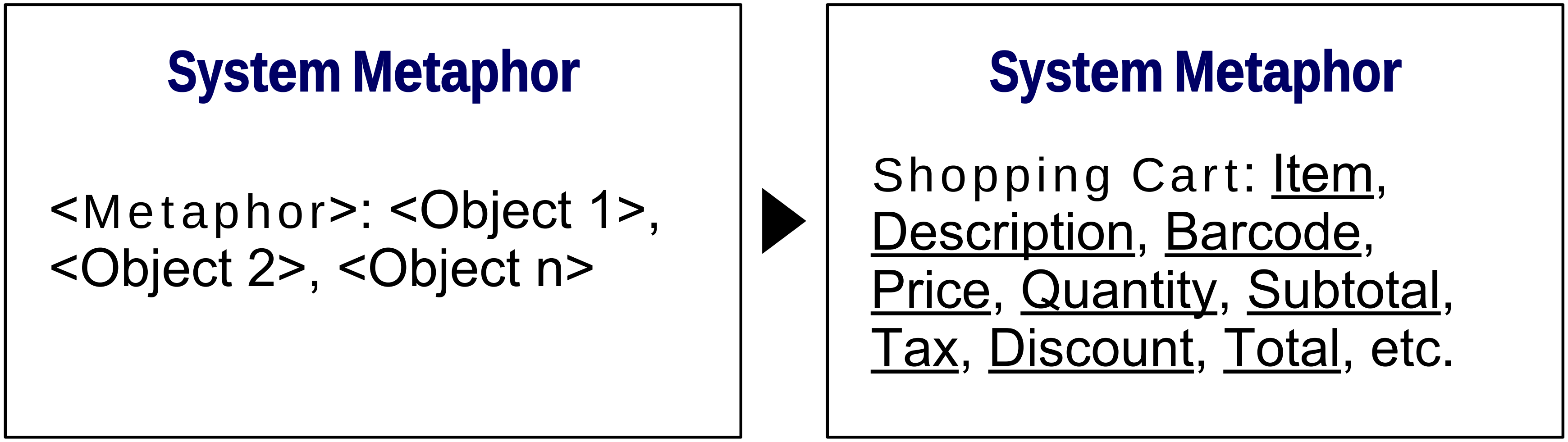
Make a Reservation

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.

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>

▶

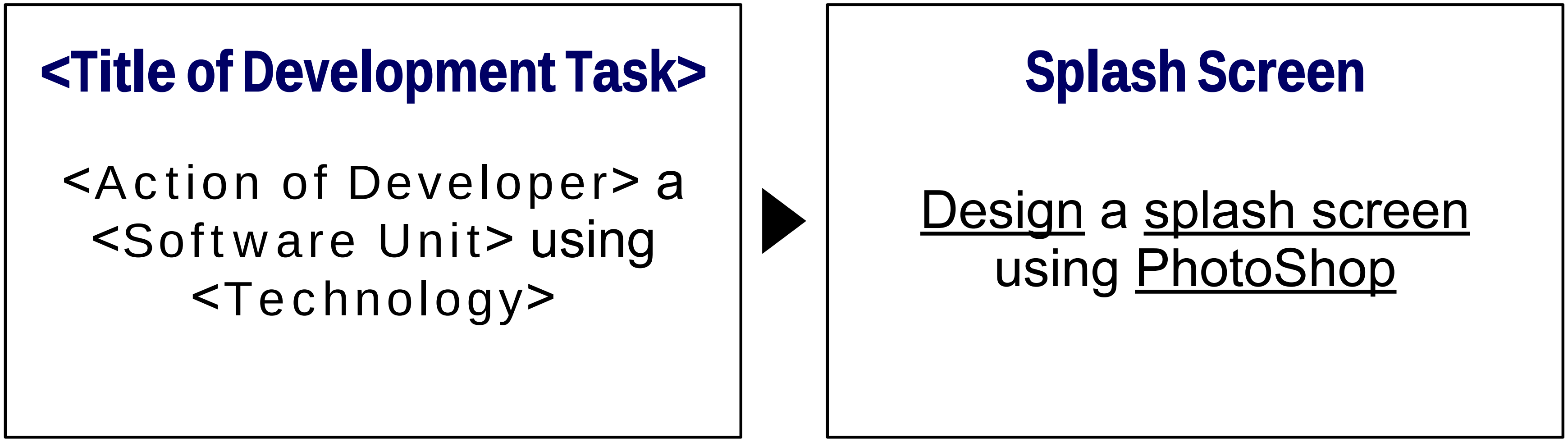
Release Plan

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.

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

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>

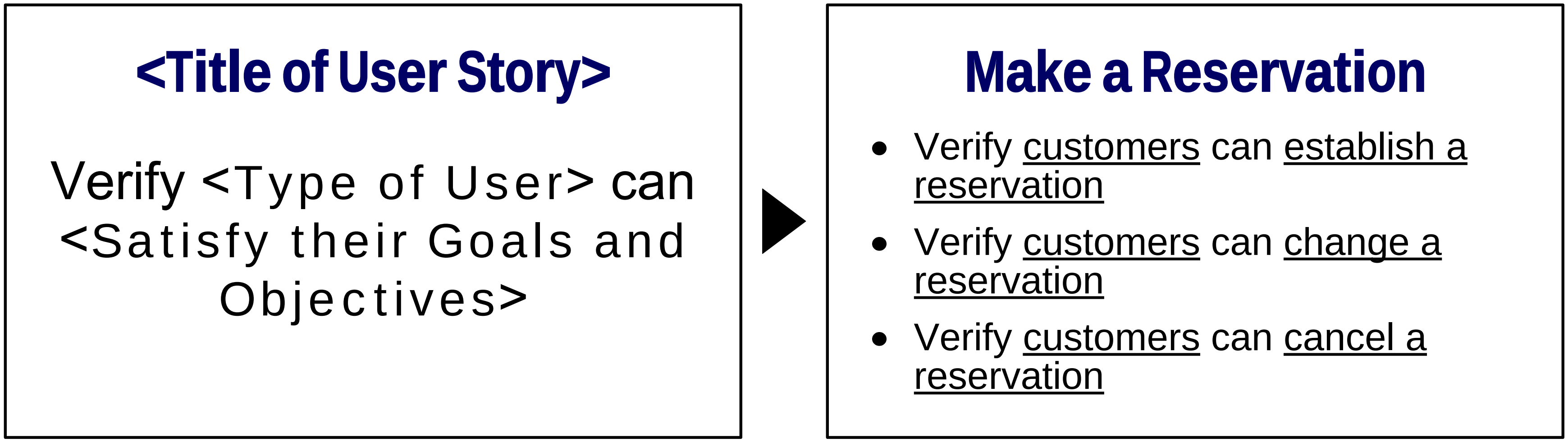
Iteration Plan

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.

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>

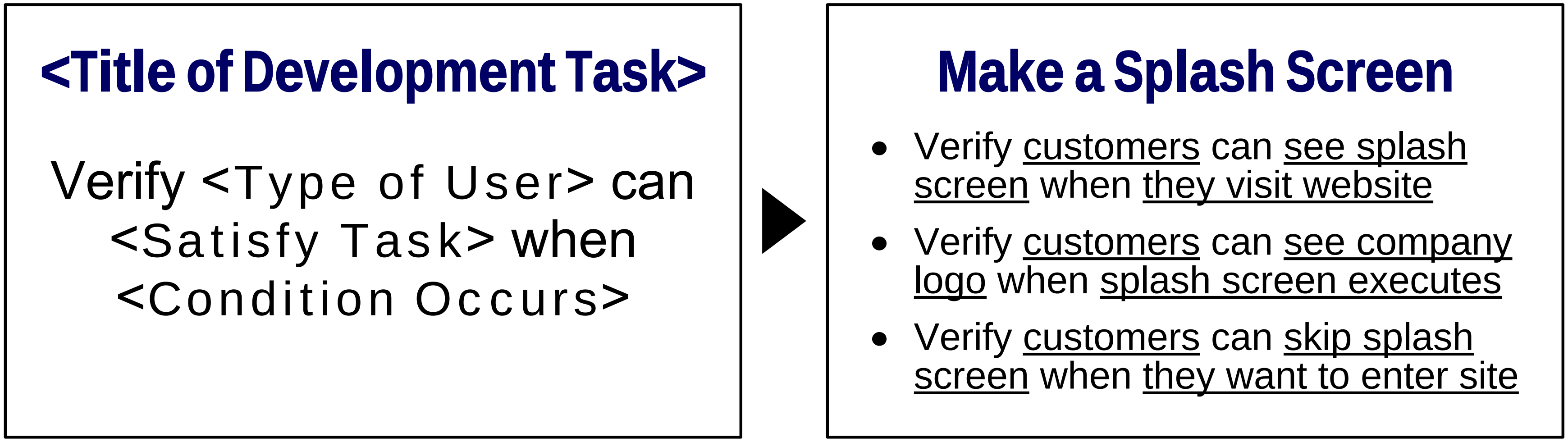
Make a Reservation

- 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

- Background
- Examples
- Deliverables
- ☞ 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)	$N(d_1) \times Benefits - N(d_2) \times Costs \times e^{-Rate \times Years}$

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

Rico, D. F. (2007). *Practical metrics and models for ROI with real options*. Retrieved September 3, 2008, from http://davidfrico.com/rico07b.pdf

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%

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

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	LOC ÷ 16.1575	619	0.7466 × KLOC × 100	747	$100	$136,548
TDD	10,000	LOC ÷ 29.2800	342	2.1550 × KLOC × 100	2,155	$100	$249,653
PP	10,000	LOC ÷ 33.4044	299	2.3550 × KLOC × 100	2,355	$100	$265,437
Scrum	10,000	LOC ÷ 05.4436	1,837	3.9450 × KLOC × 100	3,945	$100	$578,202
Agile	10,000	LOC ÷ 21.2374	471	1.7972 × KLOC × 100	1,797	$100	$226,805

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

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	(LOC × 10.51 – 6,666.67 × 9) × 100	$4,509,997	$136,548	$4,373,449
TDD	10,000	(LOC × 10.51 – 6,666.67 × 9) × 100	$4,509,997	$249,653	$4,260,344
PP	10,000	(LOC × 10.51 – 6,666.67 × 9) × 100	$4,509,997	$265,437	$4,244,560
Scrum	10,000	(LOC × 10.51 – 6,666.67 × 9) × 100	$4,509,997	$578,202	$3,931,795
Agile	10,000	(LOC × 10.51 – 6,666.67 × 9) × 100	$4,509,997	$226,805	$4,283,192

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 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
Agile	$226,805	$4,283,192	19:1	1,788%	$3,481,992	$12,010	$4,110,308
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

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

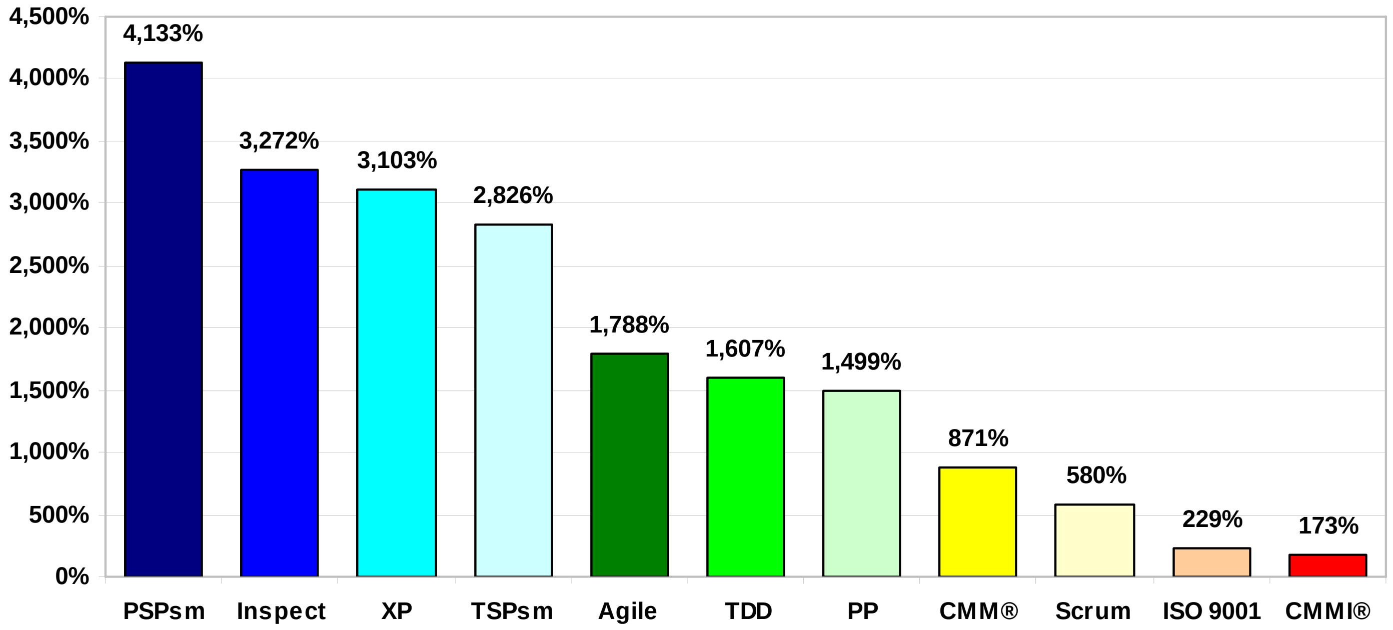
- Traditional Methods data was used for comparison
- All methods were ranked according to their ROI
- Methods with higher quality had higher ROI

Method	Costs	Benefits	B/CR	ROI	NPV	BEP	ROA
PSP^{sm}	$105,600	$4,469,997	42:1	4,133%	$3,764,950	$945	$4,387,756
Inspection	$82,073	$2,767,464	34:1	3,272%	$2,314,261	$51,677	$2,703,545
XP	$136,548	$4,373,449	32:1	3,103%	$3,650,401	$4,263	$4,267,105
TSP^{sm}	$148,400	$4,341,496	29:1	2,826%	$3,610,882	$5,760	$4,225,923
Agile	$226,805	$4,283,192	19:1	1,788%	$3,481,992	$12,010	$4,110,118
TDD	$249,653	$4,260,344	17:1	1,607%	$3,439,359	$14,629	$4,073,167
PP	$265,437	$4,244,560	16:1	1,499%	$3,409,908	$16,599	$4,048,404
SW-CMM®	$311,433	$3,023,064	10:1	871%	$2,306,224	$153,182	$2,828,802
Scrum	$578,202	$3,931,795	7:1	580%	$2,826,320	$85,029	$3,622,271
ISO 9001	$173,000	$569,841	3:1	229%	$320,423	$1,196,206	$503,345
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

Business Value

☞ Other Considerations

Conclusion

References

Strengths & Weaknesses

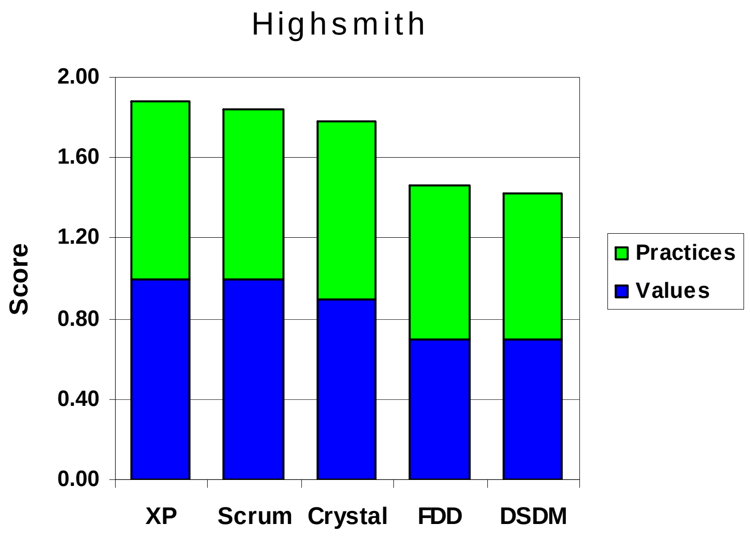
- Some follow the Agile Manifesto better than others
- Some have more process and document formality
- Often mistakenly compared to traditional methods

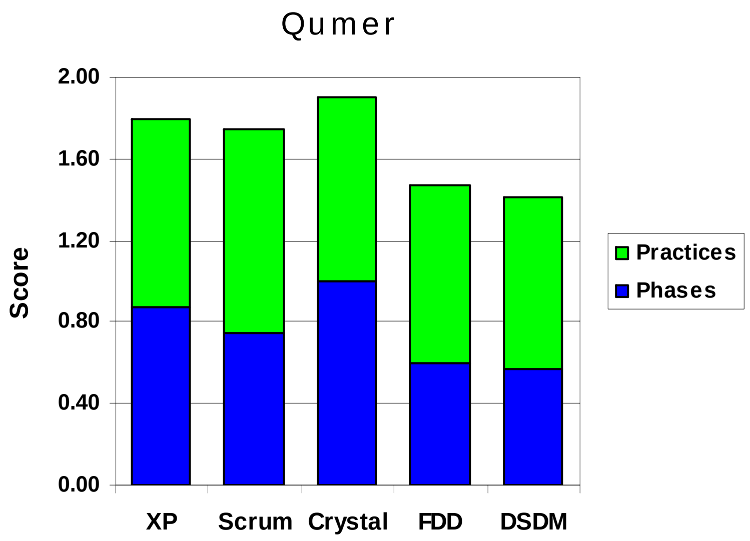
Method	Strengths	Weaknesses
XP	• Technical practices • Customer ownership • Frequent feedback • Widely known	• Onsite customer • Informal documentation • Little or no architecture
Scrum	• Self organizing teams • Customer participation • Focus on business value • Certification process	• No sub-disciplines • No technical practices • Feature prioritization
Crystal	• Scalable methodology • Support for safety-critical systems • Scalable project team size • Emphasis on testing	• Requires co-located teams • Backward and forward compatibility • Non-real time scalability
FDD	• Support for parallel teams • Product feature focused • Easy to adopt • Scales to large teams or projects	• Promotes individual code ownership • Release planning is not well-defined • Incompatible with other approaches
DSDM	• Emphasis on testing • Business focused • Prioritization of requirements • Sets stakeholder expectations early	• Most heavyweight approach • Continuous user involvement • Heavy documentation • Proprietary approach

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

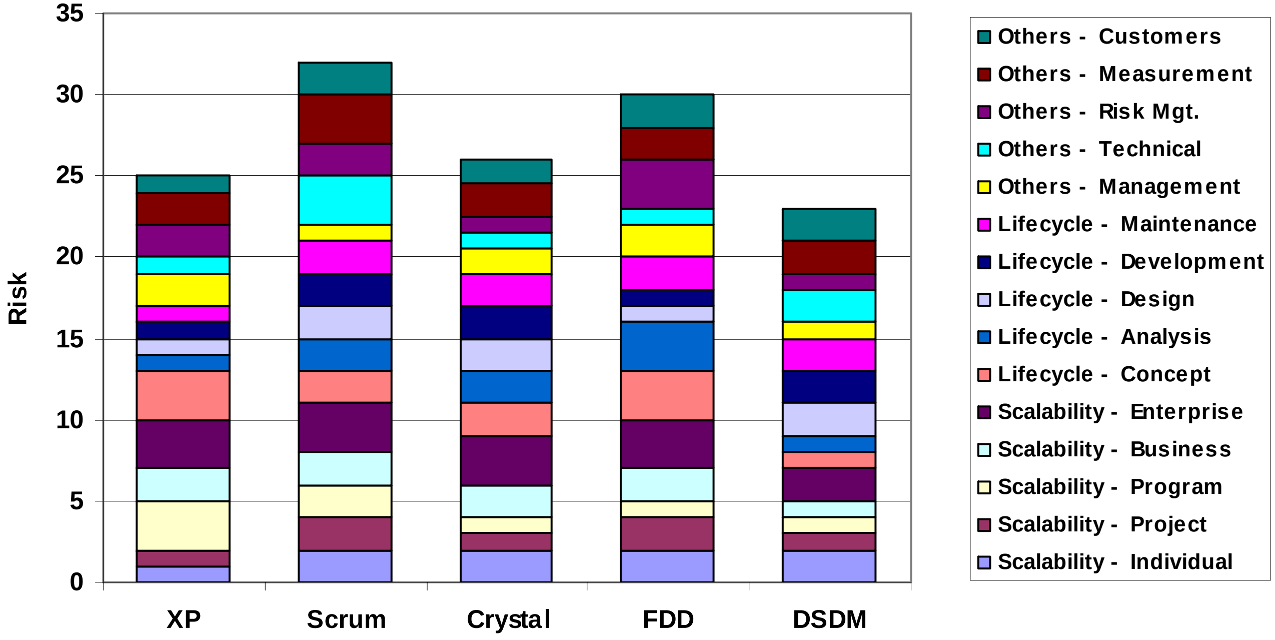




Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley.
Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods. *IS&T*, *50*(*4*), 280-295.

Degree of Risk

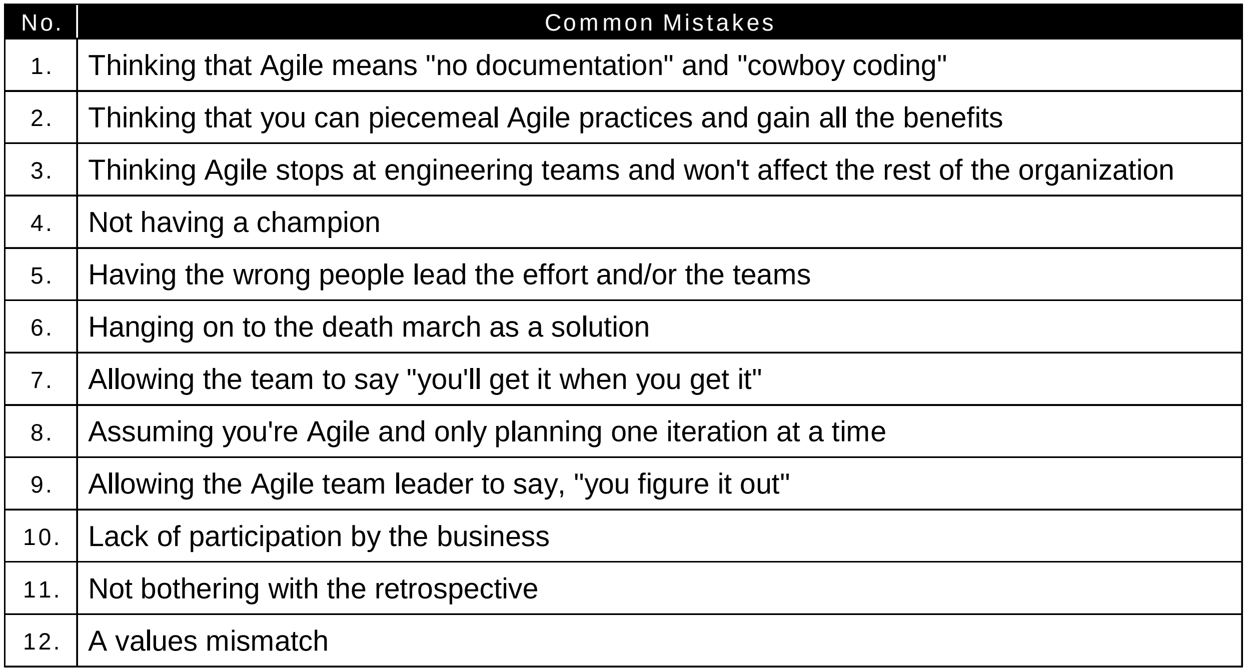
- Agile Manifesto should be used to measure risk
- Risk is often measured using Traditional Methods
- Some traditional factors may be considered (not all)



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	• Regular delivery of software	• Delivering important features first
Technical	• Well-defined coding standards • Pursuing simple design • Rigorous refactoring activities	• Right amount of documentation • Correct integration testing
Personnel	• High competence and expertise • Great motivation • Managers knowledgeable in agile	• Adaptive management style • Appropriate technical training
Management	• Agile requirements management • Agile project management • Agile configuration management	• Good progress tracking mechanism • Strong daily communication • Honoring regular working schedule
Teamwork	• Collocation of the whole team • Coherent self-organizing teamwork	• Projects with small team • No multiple independent teams
Customers	• Good customer relationship • Strong customer commitment	• 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**	• Promote software craftsmanship • Foster team collaboration • Form a guiding coalition • Cultivate informal communities of practice	• Identify the project community • Design a holographic formal structure • Get self-disciplined team players • Propose an adaptive IT enterprise
	Guiding vision	• Evolve a team vision • Align the team • Envision a bold future • Create and maintain shared expectations	• Discover business outcomes • Clearly delineate scope • Estimate level of effort • Design a vision box and elevator statement
Encourage emergence and self-organization	**Simple rules**	• Enlist the team for change • Focus on business value	• Assess the status quo and customize method • Develop a release/iteration plan/backlog • Facilitate design, code, test, and deployment • Conduct testing and manage release
	Open information	• Conduct a standup meeting daily • Encourage feedback • Build trust • Link language with action	• Collocate team members and practice pairing • Negotiate a customer representative on-site • Encourage the use of information radiators • Map the project's value stream
	Light touch	• 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	• Decentralize control • Establish a pull task management system • Manage the flow • Use action sprints
Institute leadership and adaptation	**Adaptive leadership**	• Cultivate an embodied presence • Practice embodied learning	• 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

	Level	Embrace Change	Frequent Delivery	Human Centricity	Technical Excellence	Customer Collaboration
5	Ambient	• Low ceremony	• Agile estimation	• Ideal physical setup	• Test driven dev. • Pair programming • Top performers	• Frequent interaction
4	Adapt	• Client-driven iteration • Continuous feedback	• Frequent releases • Adaptive planning		• Daily stand-ups • Agile documentation • User stories	• Accessible customer • Customer contract
3	Effective		• Risk-driven iterations • Feature-driven • Feature-tracking	• Self-organizing team • Collocated teams	• Continuous integ. • Continuous improve. • Unit testing • Good performers	
2	Evolve	• Evolutionary stories	• Continuous deliver • Multi-level planning		• Configuration mgt. • Iteration tracking • Evolutionary design	• Evolutionary contract
1	Collaborate	• Process reflection	• Collaborative planning	• Collaborative teams • Empowered teams	• Coding standards • Collaborative tools • Task volunteering	• 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

Business Value

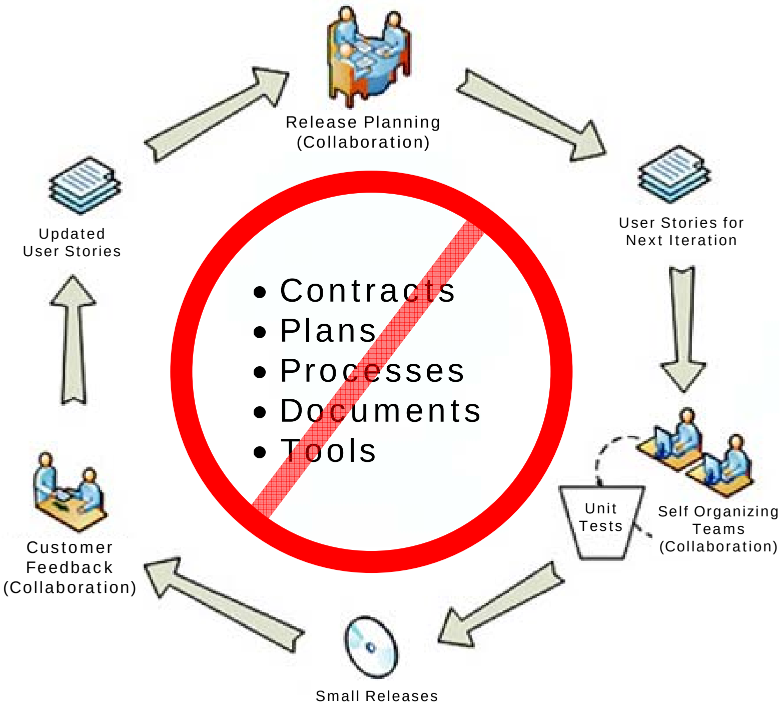
Other Considerations

☞ 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


Release Planning (Collaboration)
User Stories for Next Iteration
Self Organizing Teams (Collaboration)
Unit Tests
Small Releases
Customer Feedback (Collaboration)
Updated User Stories
• Contracts
• Plans
• Processes
• Documents
• Tools

Agenda

Background

Examples

Deliverables

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.
- Emery, P. (2002). *The dangers of extreme programming*. Retrieved September 3, 2008, from http://members.cox.net/cobbler/XPDangers.htm
- Highsmith, J. A. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley.
- Wake, W. C. (2002). *Extreme programming explored*. Upper Saddle River, NJ: Addison-Wesley.