



Making Agile Work for You

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Timothy A. Chick
April 2011



Topics

Agile Overview

Selecting the Appropriate Agile Methods

Maintaining Consistency and Agility using TSP



Agile or agile

Agile – a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

agile –

1. Characterized by quickness, lightness, and ease of movement; nimble.
2. Mentally quick or alert.



Agile Manifesto

Manifesto for Agile Software Development

February 2001

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



The Twelve Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



Agile Methods

There is no single *Agile* method, but there are several methods that are shaped by Agile principles.

- Extreme Programming (XP)
- SCRUM
- Dynamic Systems Development Method (DSDM)
- Adaptive Software Development
- Crystal
- Feature-Driven Development
- Pragmatic Programming
- TSP

Today, any method that is an alternative to documentation driven, heavyweight software development processes is probably an *Agile* method.



Extreme Programming -1

The Twelve Practices of XP

1. The Planning Game - XP follows an iterative development process and the planning game is used to develop an overall plan, the release plan, and the iteration plan.
2. On-site customer - A requirement of XP is to include a customer on the team and the customer must be available full-time to answer questions as they arise, preferably face to face, and preferably on site.
3. Small releases - The product is built in small releases with very few features implemented during each iteration (time boxed).
4. Metaphor - XP discourages defining an architecture or creating a framework for the product being developed, instead, XP uses something called a metaphor. The metaphor provides an idea or a model for the system, which developers can use as guidance during the entire project.
5. Simple design - The design is kept as simple as possible. You only do enough design to implement the next feature.
6. Testing - Programmers write automated unit tests for every feature. No new feature is implemented without writing the unit test first. The customer writes acceptance tests. Acceptance tests are used to test the functionality of individual features of a product.



Extreme Programming -2

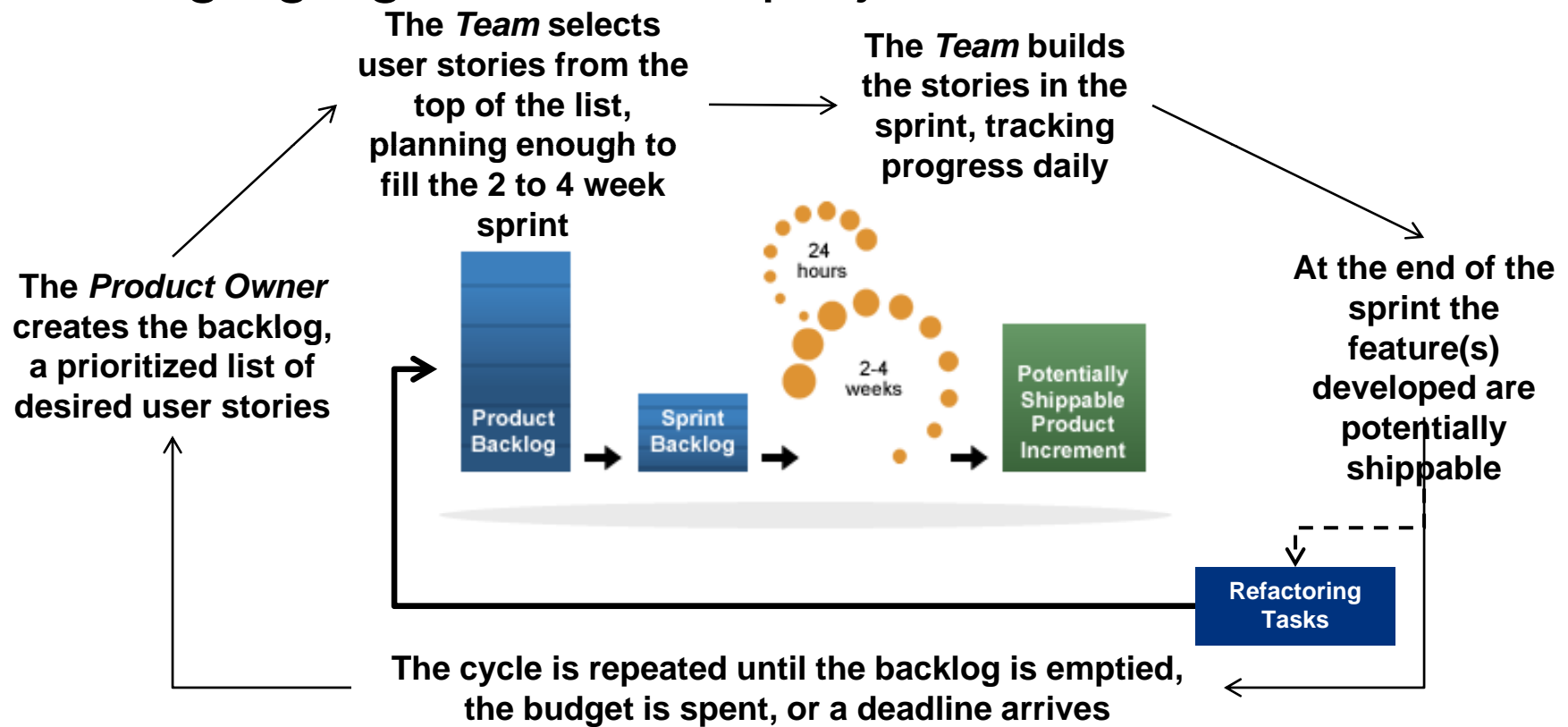
The Twelve Practices of XP (continued)

7. Refactoring - Refactoring improves the design of existing code by making it simpler, faster, more robust, cleaner, or by eliminating duplication, without changing the function of the code.
8. Pair programming - All production code is written with two people using one machine, one keyboard, and one mouse. Pairs change constantly. Pair programming also includes pair designing and pair unit testing.
9. Collective ownership - Any developer can change any code at any time. There is no concept of code ownership. There is no concept of a system architect or a keeper of the vision.
10. Continuous integration - The system is integrated, built, and unit-tested several times a day. Each time a pair finishes a task, they must integrate and release their changes to the team. The guidelines are that no one must go longer than a day before they integrate.
11. 40-hour week - XP teams work 40-hour weeks. Teams will never work more than two back-to-back weeks that exceed 40 hours per week.
12. Coding standards - Since anyone can make changes to any code at any time, the team must use a coding standard. Teams define and evolve their own coding standard.



SCRUM

Scrum is an iterative, incremental methodology for managing agile software projects.



Development Activities Within an Agile Sprint¹

Team members

- select the next user story from the top of the sprint queue.
- write the unit tests for the user story, then write and test the code.²
- interact with the product owner and demo working code.

At the end of the sprint

- a retrospective is held to document lessons learned
- any refactoring work is added to the next sprint
- unfinished user stories slip to the next sprint

1. Adapted from CMU/SEI-2010-TN-002; *Considerations for Using Agile in DoD Acquisition*

2. Two team members if pair programming



Measures and Progress in Agile Methods

Working software – the potentially releasable features created during each sprint.

Burn-down chart – tracks the work completed and work remaining.

- two views – the current sprint and the overall project
- includes the hours of effort remaining and the number of tasks remaining in for the sprint or project.

Velocity – measures the number of user stories completed in a sprint and can be used to predict the completion date range.

Cyclomatic complexity – a measure of the “goodness” of the software that is used to establish the need for refactoring.



Topics

Agile Overview

Selecting the Appropriate Agile Methods

Maintaining Consistency and Agility using TSP



Traditional and Agile Perspective on Software Development

	Traditional View	Agile Perspective
Design Process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving Process	Selection of the best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the Environment	Stable, predictable	Turbulent, difficult to predict
Type of Learning	Single-loop/adaptive	Double-loop/generative
Key Characteristics	Control and direction Avoid conflict Formalizes innovation Manager is controller Design precedes implementation	Collaboration and communication; integrates different worldviews Embraces conflict and dialectics Encourages exploration and creativity; opportunistic Manager is facilitator Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical and/or Philosophical Roots	Logical positivism, scientific method	Action learning, John Dewey's pragmatism, phenomenology

Dyba, Tore; Torgeir Dingsoyr, *What Do We Know about Agile Software Development*, Published by the IEEE Computer Society, 0740-7459/09, 2009

Symbiotic Relationship

Agile and traditional methods have a symbiotic relationship, in which the process selected is based on:

- Size of project/application
- Application domain
- Criticality
- Innovativeness

So how does an organization identify the agile practices that improve the development process without causing any harmful side-effects to the product, project, system, or organization?

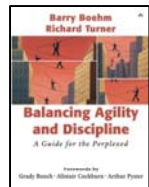


Five Critical Factors

Boehm and Turner identify five critical factors to guide method selection and/or tailoring.

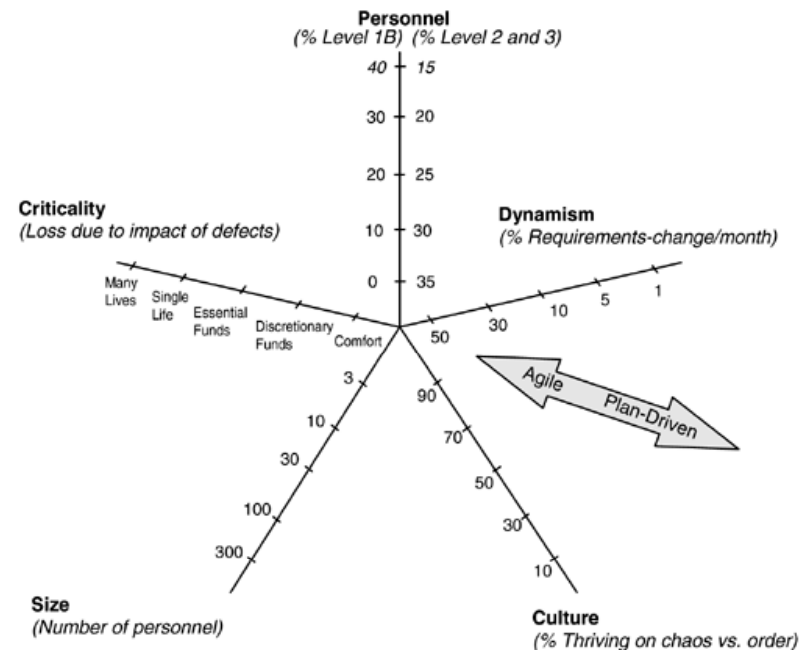
- Seniority of personnel
- Dynamism
- Organization’s Culture
- Project size
- Criticality

“...there will be a higher premium on having methods available that combine discipline and agility in situation-tailorable ways.”



1. Boehm, B.; R. Turner (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*.

Figure 6-1. Home Ground Polar Chart



Three Fundamental Factors

1. Identifying the ability of the organization to adopt agile practices
2. Determining the suitability of agile practices in the development of a given product or system
3. Determining the suitability of agile practices for the organization developing the product or system.

Adapted from Sidky, Ahmed; James Arther, *Determining the Applicability of Agile Practices to Mission and Life-critical Systems*, Proceedings of the 31st IEEE Software Engineering Workshop (SEW 2007). pp 3-12



Ability of the Organization to Adopt Agile

1. Agile is like any other Process Improvement Initiative
 - Successful adoption of agile practices requires the absorption of associated costs, as well as expending the required time and effort.
2. Method for reaching a decision to proceed or not
 - a. Identify success factors, such as
 - Level of executive sponsorship
 - Organizational desire to improve
 - Availability of resources
 - b. Assess the extent to which the success factors are present
 - c. Make a Go/No-go decision based on assessment results

Adapted from Sidky, Ahmed; James Arther, *Determining the Applicability of Agile Practices to Mission and Life-critical Systems*, Proceedings of the 31st IEEE Software Engineering Workshop (SEW 2007). pp 3-12



Suitability of Agile Practices

1. Development and product characteristics play a large role in determining the suitability of a particular agile methods
 - For example, life-critical systems are usually characterized as being large, complex, and having long development periods
2. The desired product qualities also play a role in determining appropriate agile methods.
 - For example, Maintainability and Reliability

An agile practice is considered unsuitable and must be discarded when it conflicts with integral product/development characteristics, or precludes the attainment of desirable qualities

Adapted from Sidky, Ahmed; James Arther, *Determining the Applicability of Agile Practices to Mission and Life-critical Systems*, Proceedings of the 31st IEEE Software Engineering Workshop (SEW 2007). pp 3-12



When is this unsuitable or impractical? - 1

1. On-site customer - the customer must be available full-time to answer questions as they arise, preferably face-to-face, and preferably on-site.

Many stakeholders who are unable to speak to one another's requirements and priorities (need for a traditional System's Engineer or Requirements Analysis)

2. Simple design - The design is kept as simple as possible. You only do enough design to implement the next feature.

Refactoring is most difficult without an architecture. It is important to distinguish between basic infrastructure services and application functionality. The former must be stable very fast and experience very little change, because in infrastructure code can have serious ripple effects that result in cost prohibitive changes

Source of examples: Emam, Khaled El; *Finding Success in Small Software Projects*, Cutter Consortium, Agile Project Management, Vol. 4, No. 11.



When is this unsuitable or impractical? - 2

1. Minimal documentation – The discouragement of writing requirements and design because of the cost associated with developing and maintaining them and the fact they do not deliver value to the customer, only working software does
 1. **Staffing flux** results in the tacit knowledge of the developers being lost, which can result in a considerable slowdown in the work
 2. The lack of documentation demonstrating the origins of some technical material can be costly when faced with an **intellectual property lawsuit**
2. Refactoring – making existing code simpler, faster, more robust, cleaner, or eliminating duplication, without changing the function of the code.

Refactoring a Database Schema - because system would already be in place, migration of all data would need to be migrated to new database, which is not a trivial exercise. Also, all modules that interact with the database would require modification or at least inspection. Back to the need for a stable design up front

Source of examples: Emam, Khaled El; *Finding Success in Small Software Projects*, Cutter Consortium, Agile Project Management, Vol. 4, No. 11.



Suitability of Agile for the Organization

1. When mismatches between the agile method and the organization are identified before the adoption effort, the probability of failing during the adoption effort or introducing undesirable agile practices is reduced.

Is this a mismatch?

1. assuming **Test-First Programming (TFP)** is a suitable method in meeting the required product/development characteristics and enables the attainment of desirable qualities. It has also been determined that **your organization already has a dedicated integration and system test team**

No, as TFP addresses how the developers do unit testing and has no effect on the dedicated test team

2. assuming **Pair Programming** is a suitable method in that it meets the required product/development characteristics and enables the attainment of desirable qualities. It has also been determined **that the organization has major collaborative issues**

Yes, as Pair Programming works best in a collaborative environment

Adapted from Sidky, Ahmed; James Arther, *Determining the Applicability of Agile Practices to Mission and Life-critical Systems*, Proceedings of the 31st IEEE Software Engineering Workshop (SEW 2007). pp 3-12



Usage Varies - Microsoft Agile Study

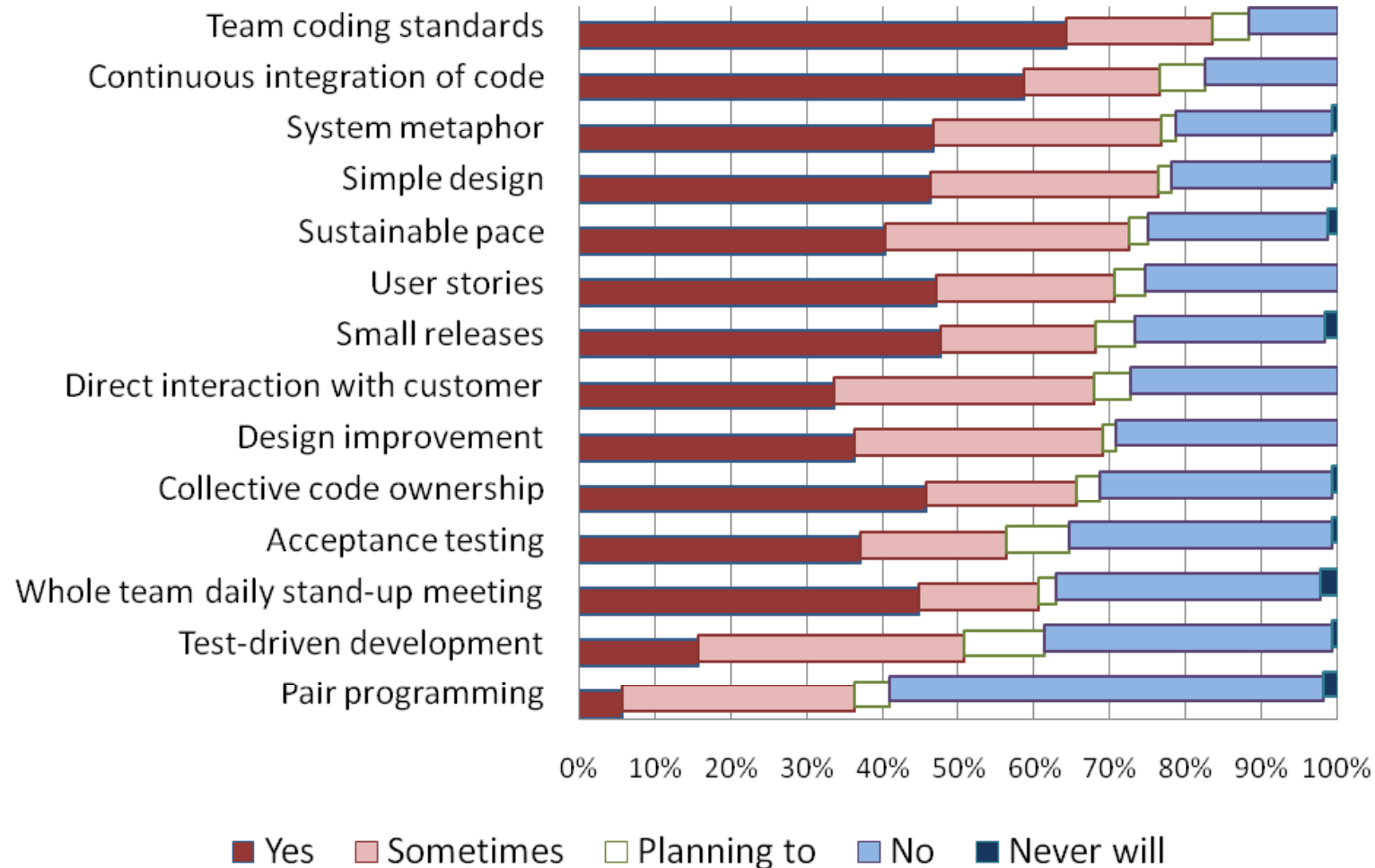
An agile practices survey of 2,800 developers, testers, and managers was conducted at Microsoft.

- Nearly 500 (17%) responded to the survey.
- Respondents had an average of 10 years of experience
- Average time on their current team was 2.5 years
- Testers and developers represented 73% of the respondents
- Managers, and managers of managers represented 24% of respondents
- More than 60% of the respondents work on legacy products

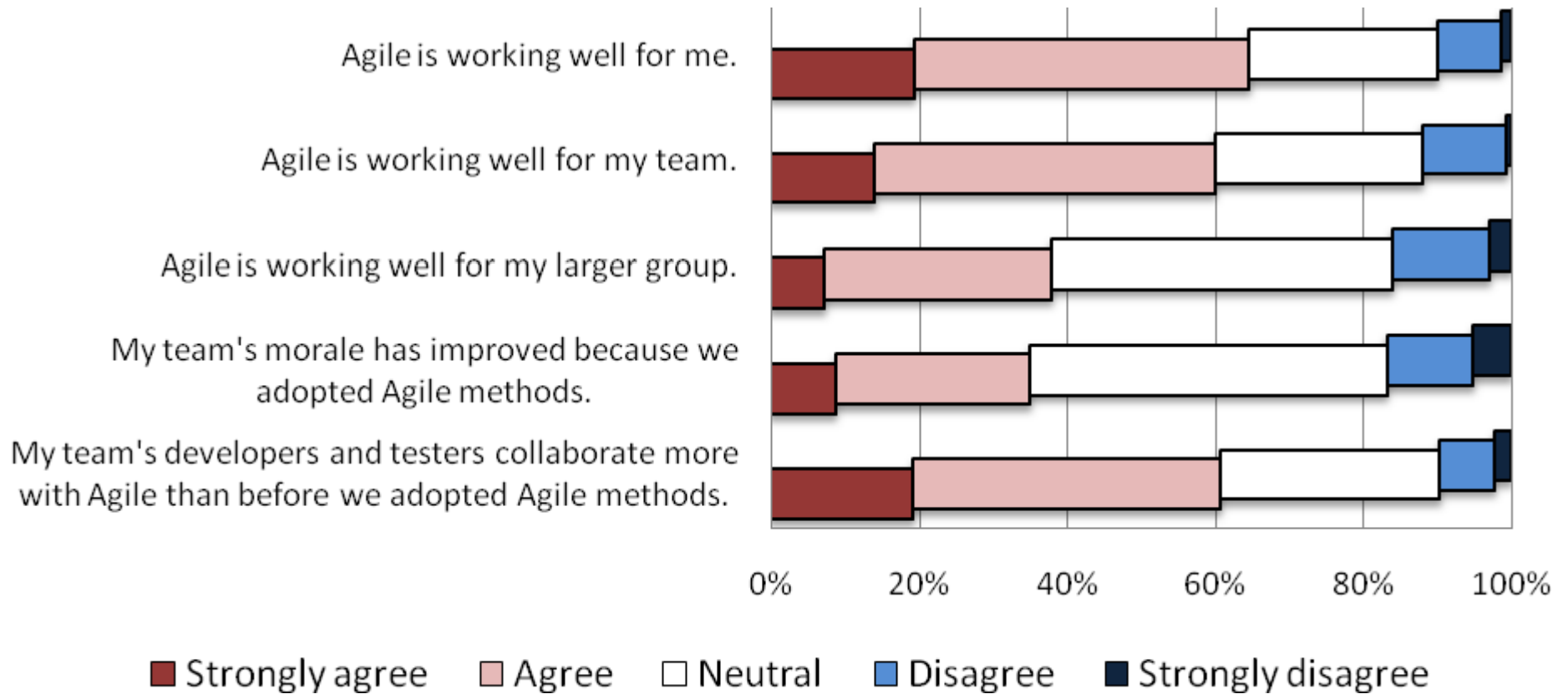
Nagappan, N; Begel, A, *Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study*, 2007



Microsoft – Agile Adoption by Practice



Microsoft – Attitude towards Agile



Critical Agile Implementation Questions -1

Agile Principle	Question
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable [assume working] software.	How would you measure the software to establish that it is both working and valuable?
2. Welcome changing requirements, even late in development . Agile processes harness change for the customer's competitive advantage .	How would you evaluate late requirements change to ensure that it provides competitive advantage for the customer?
3. Deliver working software frequently , from a couple of weeks to a couple of months, with a preference to the shorter timescale.	How do you determine the optimal frequency? Can you deliver too frequently?
4. Business people and developers must work together daily throughout the project.	Is daily availability realistic? What if there is more than one stakeholder position?
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done .	Trust is earned, not given. How will you establish and sustain a trusting relationship with the software development team?
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Is face-to-face practical? What if the teams are large or distributed? What happens to the tacit system knowledge that the team has after the team is dissolved?



Critical Agile Implementation Questions -2

Agile Principle	Discussion Question
7. Working software is the primary measure of progress.	How would you use “working software” as a progress measure?
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely .	What does maintain a constant pace indefinitely mean?
9. Continuous attention to technical excellence and good design enhances agility.	
10. Simplicity--the art of maximizing the amount of work not done--is essential.	What’s the danger here?
11. The best architectures, requirements, and designs emerge from self-organizing teams.	Working code is the progress measure but the architecture, requirements, and designs emerge. What does this imply about the working software progress measure?
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	



Topics

Agile Overview

Selecting the Appropriate Agile Methods

Maintaining Consistency and Agility using TSP



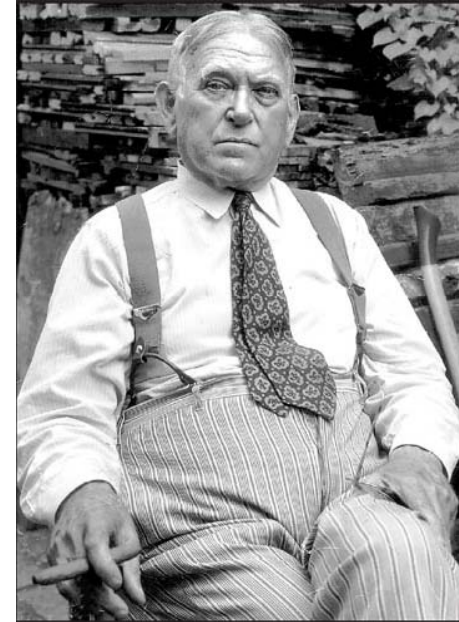
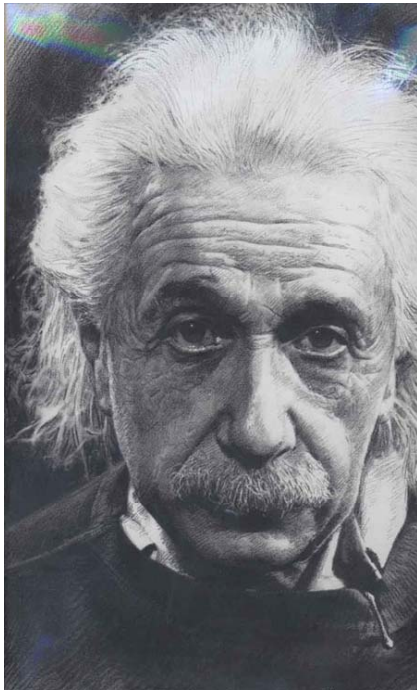
Maintaining Consistency and Scalability with Agility

How does the developing organization empower teams to make these process selections, while maintaining consistency and agility throughout the organization?



Simple But Not Easy

"Everything should be made as simple as possible, but not simpler."



"For every complex problem, there is a solution that is simple, neat, and wrong."



Team Software Process

The Team Software Process (TSP) is a process framework that was specifically designed for software teams.

It's purpose is to help teams achieve their best performance by showing them how to

- accurately estimate and plan their work
- negotiate their commitments with management
- manage and track projects to a successful conclusion
- manage quality to produce better products in less time



TSP and Agile - 1

Individuals and interactions over processes and tools

- TSP holds that individuals are the key to product quality and effective member interactions are necessary for the team to succeed
 - Project launches strive to create jelled teams
 - Weekly meetings and communication are essential to sustain them
 - Teams define their own processes in the launch
 - Sharing leadership responsibilities helps interaction between members

Working software over comprehensive documentation

- TSP teams can choose evolutionary or iterative lifecycle models to deliver early functionality – the focus is on high quality working software from the start. TSP does not require heavy documentation
 - Documentation should merely be sufficient to facilitate effective reviews and information sharing
 - TSP teams can determine the level of documentation produced based on organizational standards, customer needs, and system attributes.

Source: Lussier, Frederick, *CMMI Agile Processes – PSP/TSP*. Alcyonix Inc., <http://www.slideshare.net/flussier/psp-tsp-agile-3-0-en>



TSP and Agile - 2

Customer collaboration over contract negotiation

- Learning what the customer wants is a key focus of the TSP
 - The customer or a representative is required to be present when TSP teams start planning the project and first cycle.
 - The customer or a representative can also attend each re-launch to guide plans for the next cycle.
 - The TSP has a team member role, the Customer Interface Manager, for the purpose of collaboration with the customer.

Responding to change over following a plan

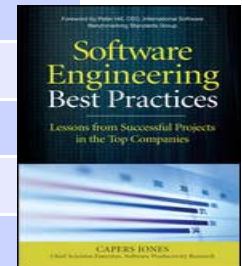
- TSP teams expect and plan for change
 - TSP teams re-plan whenever the plan is no longer a useful guide
 - New tasks are added as they are discovered
 - They dynamically rebalance the team workload as required to finish faster
 - The team's process is adjusted as needed to improve performance
 - The team continuously manages risk and uses early-warning indicators to identify and avoid technical and planning issues.

Source: Lussier, Frederick, *CMMI Agile Processes – PSP/TSP*. Alcyonix Inc., <http://www.slideshare.net/flussier/psp-tsp-agile-3-0-en>



TSP is Agile and More

Features TSP and Agile Methodologies Support	
Agile	TSP
Team organization	Team organization
Project management – planning and estimating	Project management – planning and estimating
Change control	Change control
Requirements	Requirements
Design	Design
Code development	Code development
Configuration control	Configuration control
Testing	Testing
	Specialization of team members
	Project management – tracking and control
	Reusability
	Quality assurance
	Inspections
	Static analysis
	Security
	Documentation and training



Software Engineering Best Practices, C. Jones, 2010

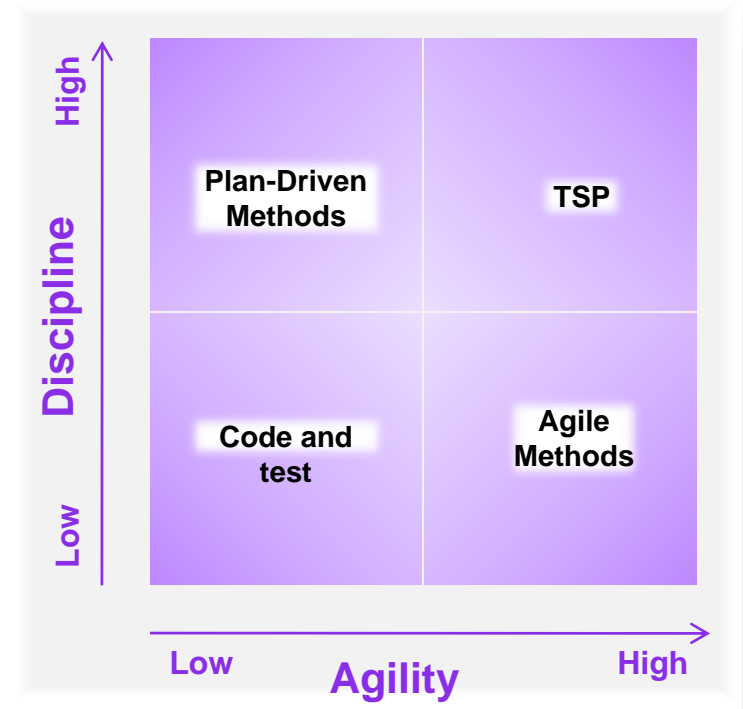


Balancing Agility and Discipline with TSP

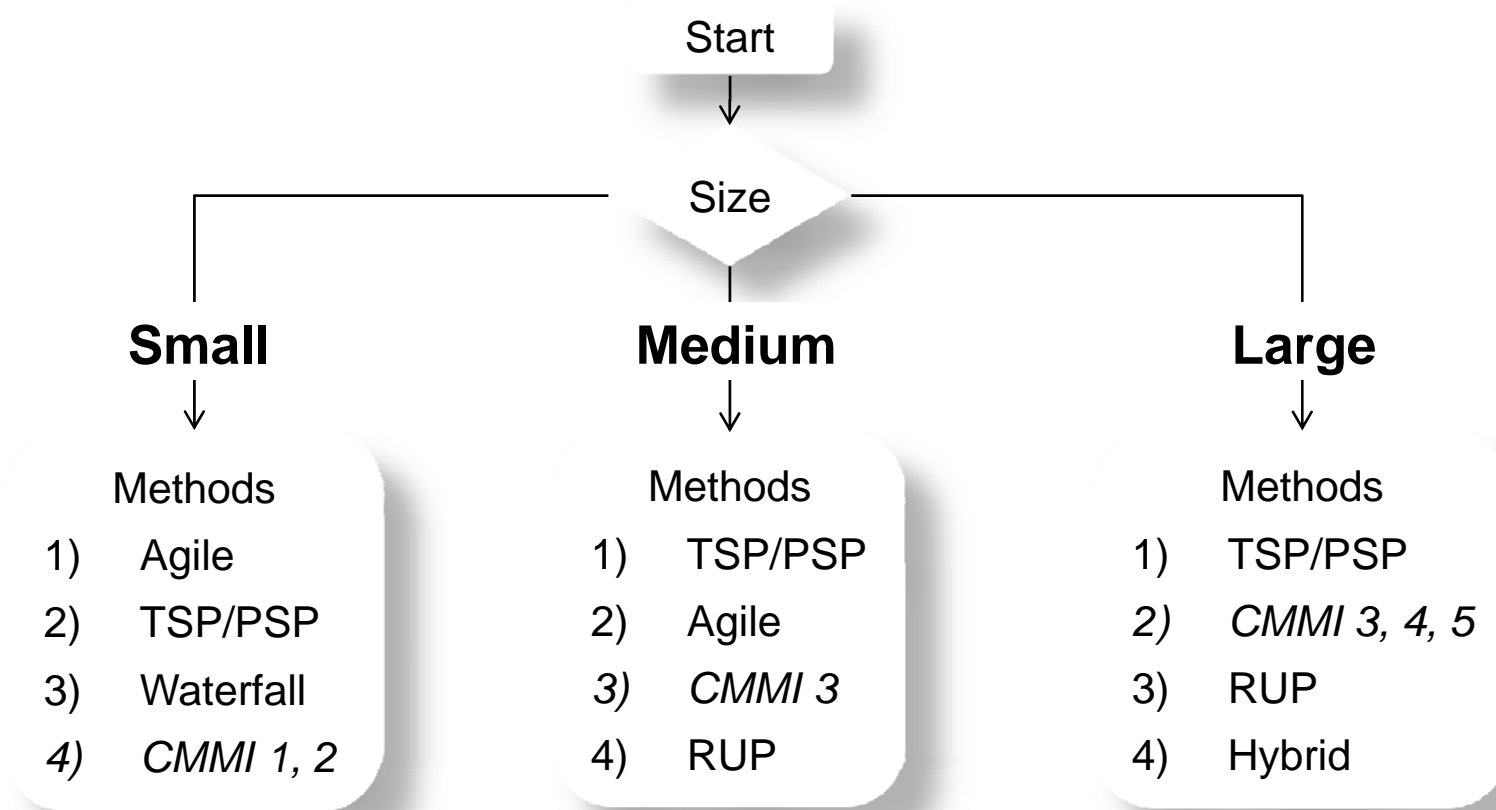
Agility and discipline are both important.

TSP

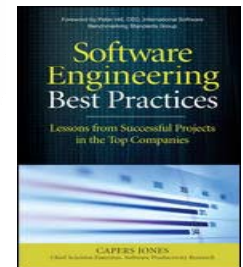
- integrates agile principles with management and engineering discipline.
- uses short iterations and delivers working code
- enables quick response to changing requirements
- scales-up in a linear fashion without sacrificing quality or performance



TSP is a Software Engineering Best Practice



Development practices by size of application^[1]



Software Engineering Best Practices, C. Jones, 2010



Summary

Agile is a means to an end.

Decide upon your goals, how you will measure success, then determine what strategy, process, and activities help you achieve those goals.

TSP can provide the disciplined, yet flexible, planning, quality, and process frameworks needed to successfully scale other Agile methods across your organization.





Attend the TSP Symposium 2011 to learn more:

- Visit www.sei.cmu.edu/tsp_symposium and sign up to receive email updates
- Join the *TSP/PSP Heroes* group on LinkedIn
- Follow @SEINews on Twitter: www.twitter.com/SEINews
- **SAVE \$100.00** using discount code: **TSPBZSN**

For information on user experience and results

- <http://www.sei.cmu.edu/tsp>
- http://www.sei.cmu.edu/tsp_symposium/2011/proceedings.cfm



Contact Information

Timothy A. Chick

Sr. Member of the Technical Staff
TSP Initiative

Telephone: +1 412-268-1473

Email: tchick@sei.cmu.edu

Web

www.sei.cmu.edu/TSP

www.sei.cmu.edu/TSPSymposium

U.S. Mail

Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Customer Relations

Email: info@sei.cmu.edu

Telephone: +1 412-268-5800

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257



NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

