

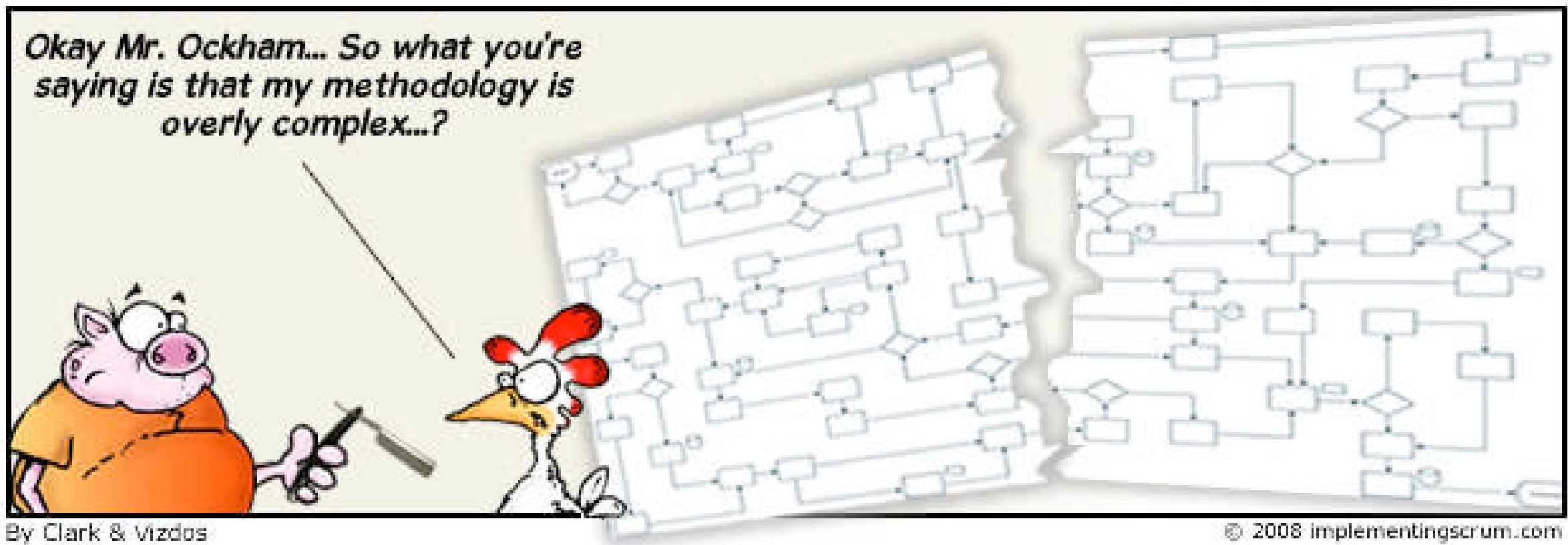
Agile Software Development with Scrum



ScrumCenter

Next level working.

Agile Basics



Introduction

Myths and Words of Wisdom

Survival and Change

“It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change.”

Charles Darwin

Simple vs. Complex Processes

“Simple, clear purpose and principles give rise to complex, intelligent behavior. Complex rules and regulations give rise to simple, stupid behavior.”

Dee Hock, founder of VISA

The Myths and Realities of Agile Software Development

▶ **Myths**

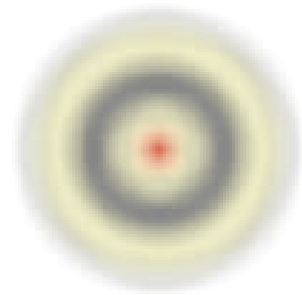
- ▶ No documentation
- ▶ No reporting
- ▶ Undisciplined & chaotic
- ▶ No architecture
- ▶ No analysis
- ▶ No planning
- ▶ Not predictable
- ▶ Not scalable
- ▶ Just a fashion
- ▶ Is a silver bullet
- ▶ Fixed price not possible

▶ **Reality**

- ▶ Documentation is right-sized
- ▶ Detailed and realistic reporting
- ▶ Structured and disciplined
- ▶ Emergent architecture
- ▶ “Just-in-time” analysis
- ▶ Long and short range planning
- ▶ Transparency!
- ▶ Works with large projects as well
- ▶ It’s mainstream now
- ▶ There are no silver bullets!
- ▶ Does fixed price really work anyway?

Traditional Development

Traditional Development

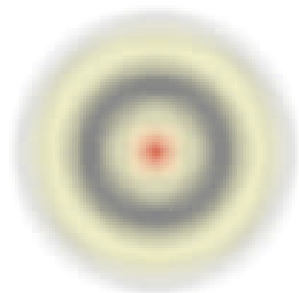


The target is not clear at the start

Traditional Development

Traditional

The target is not clear at the start

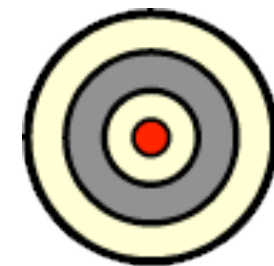


Traditional Development

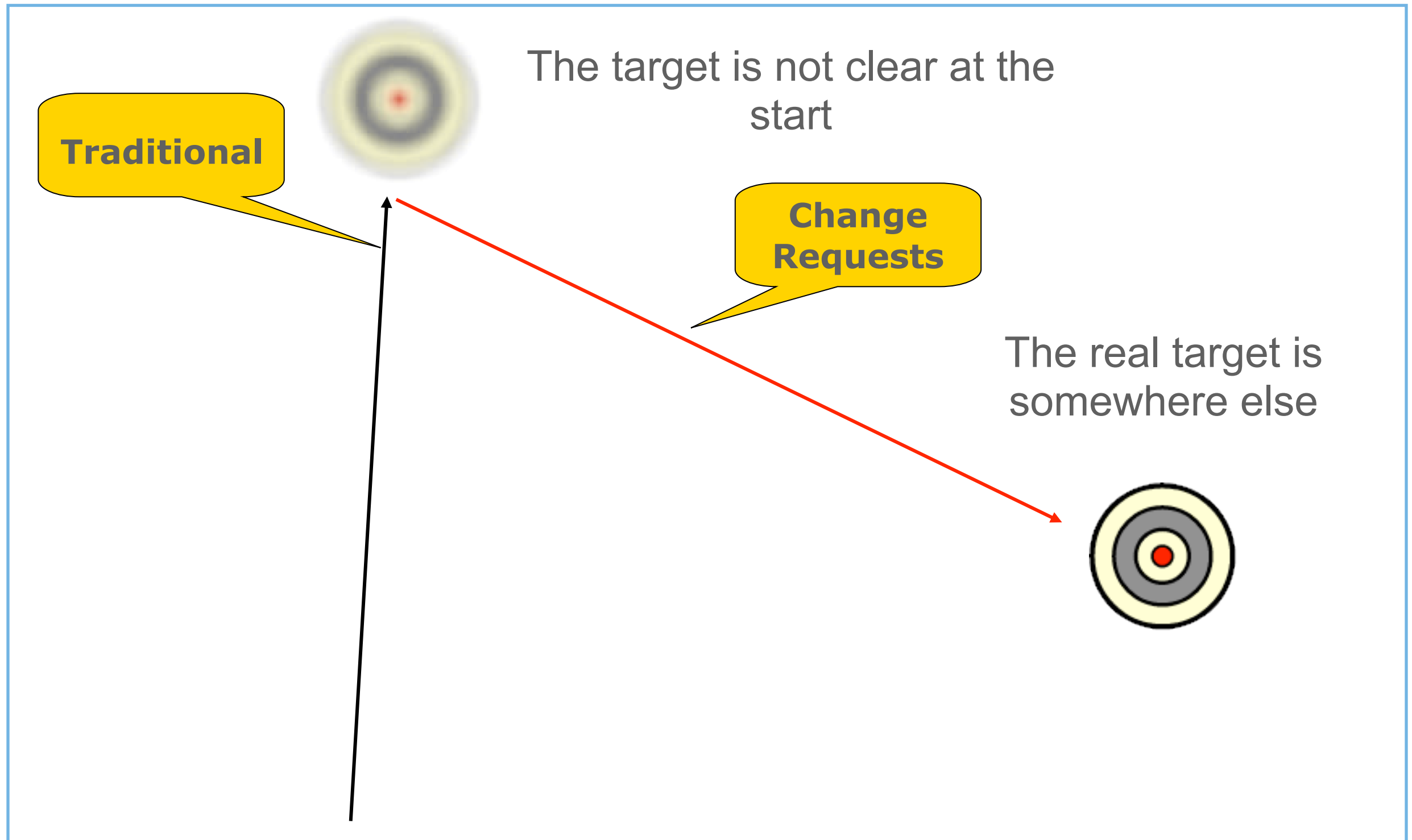
Traditional

The target is not clear at the start

The real target is somewhere else

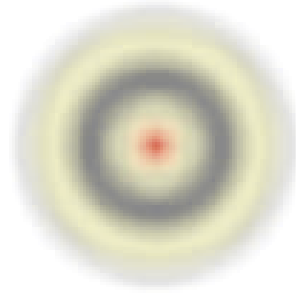


Traditional Development



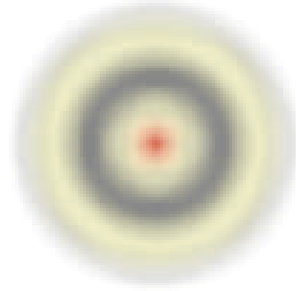
Agile Development

Agile Development



The target is not clear at the start

Agile Development



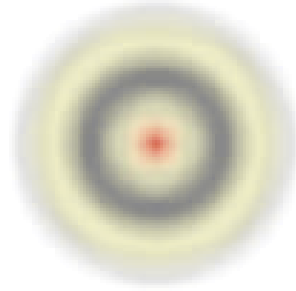
The target is not clear at the start

The real target is somewhere else



Agile Development

Traditional



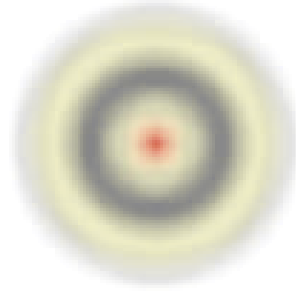
The target is not clear at the start

The real target is somewhere else



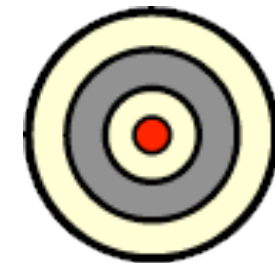
Agile Development

Traditional

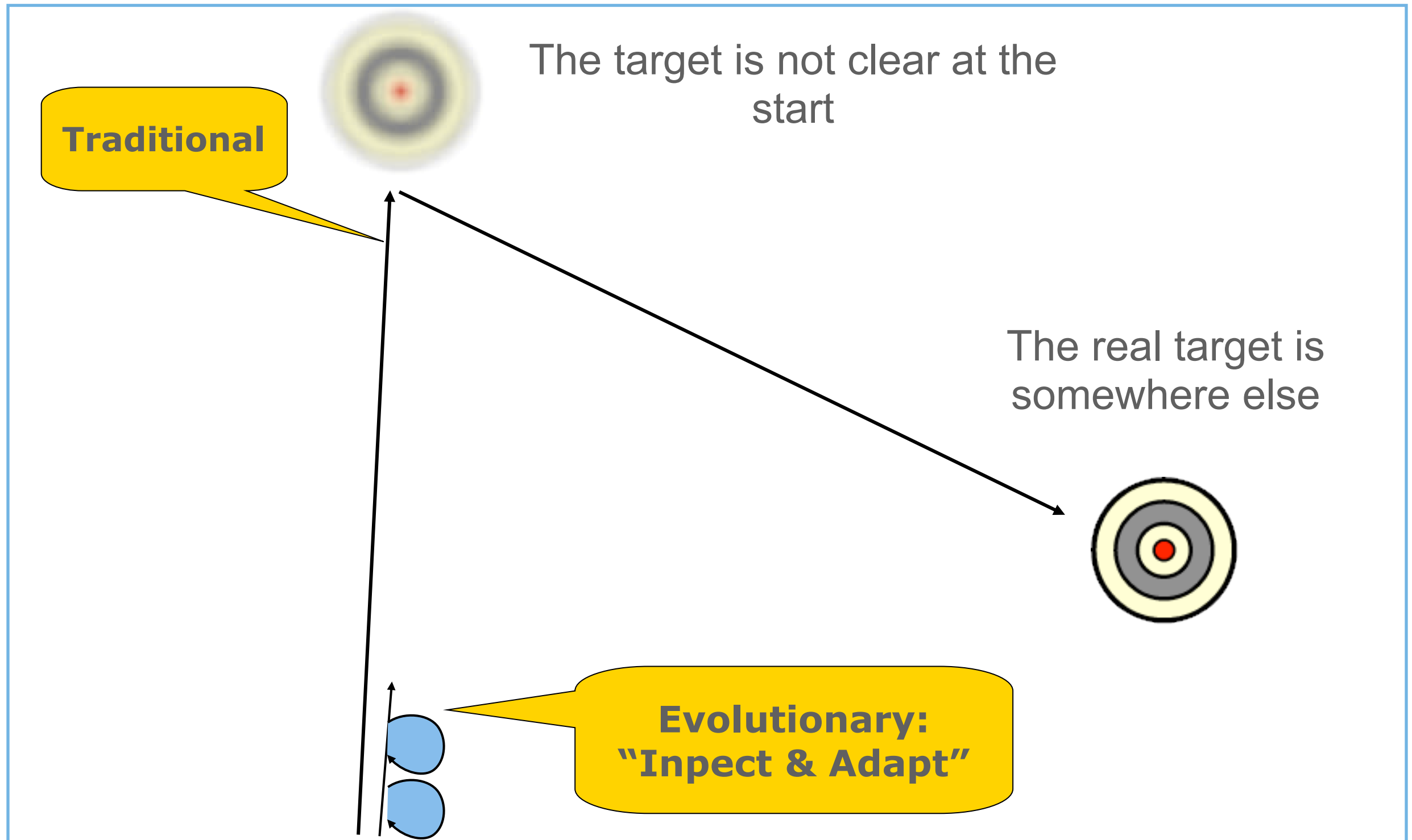


The target is not clear at the start

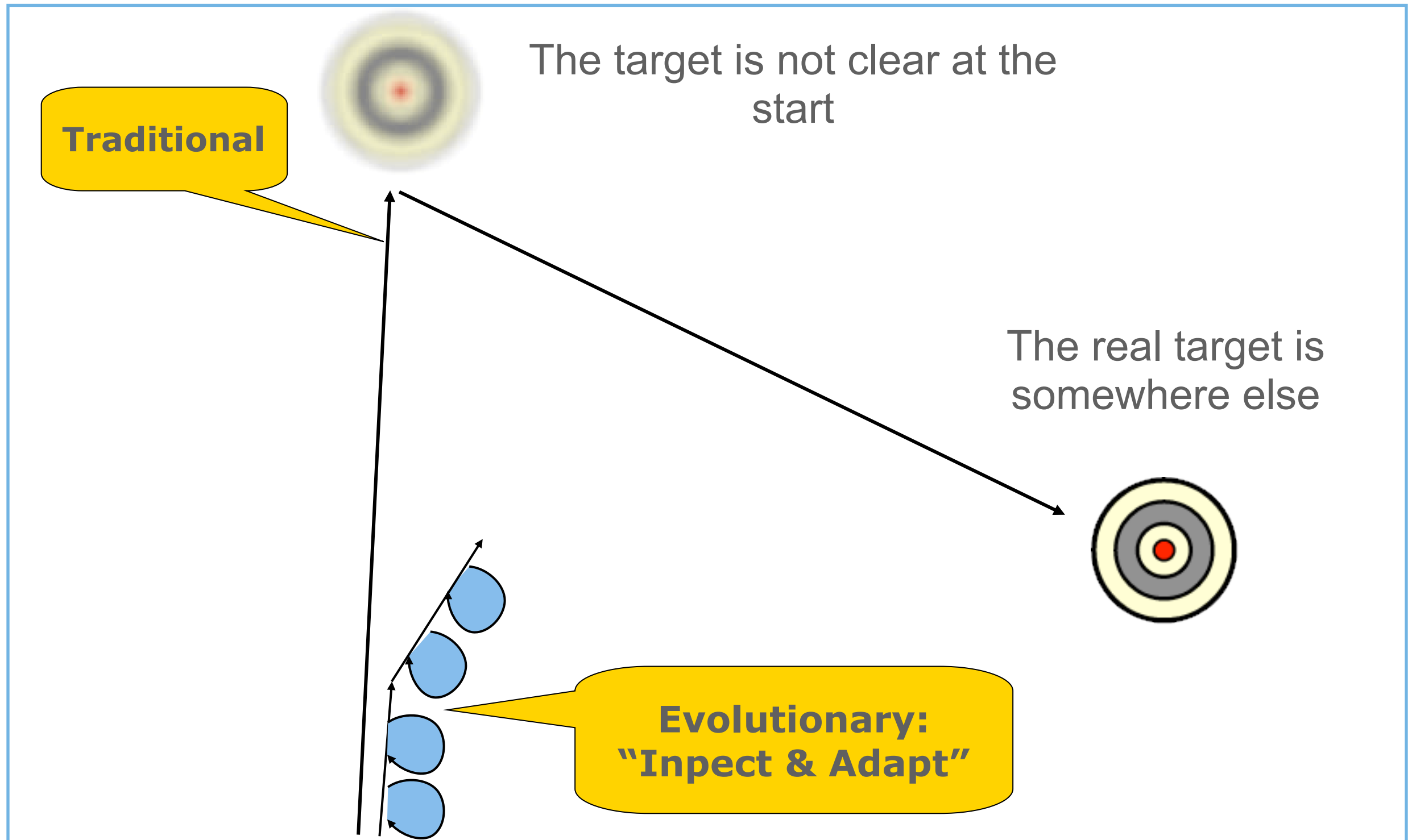
The real target is somewhere else



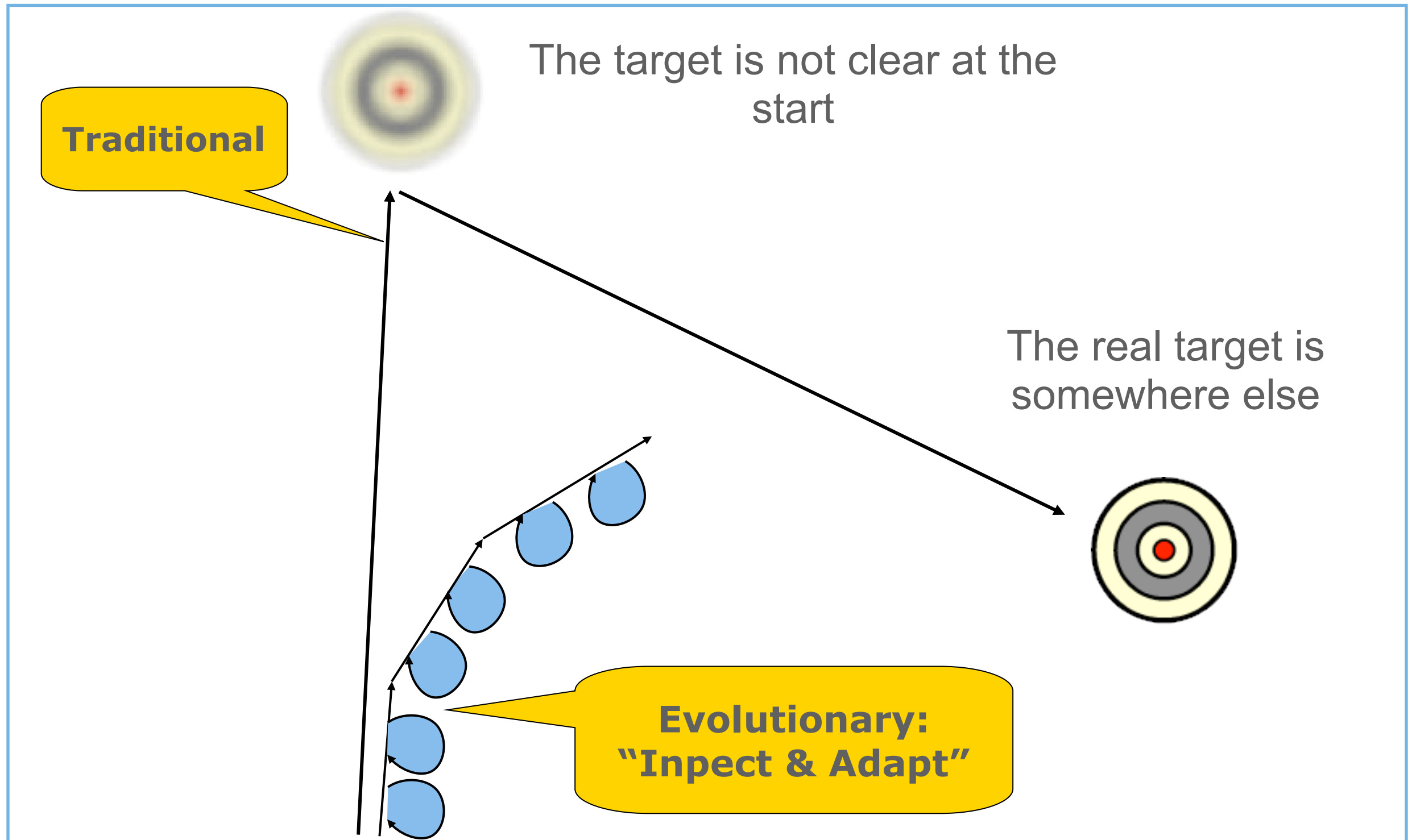
Agile Development



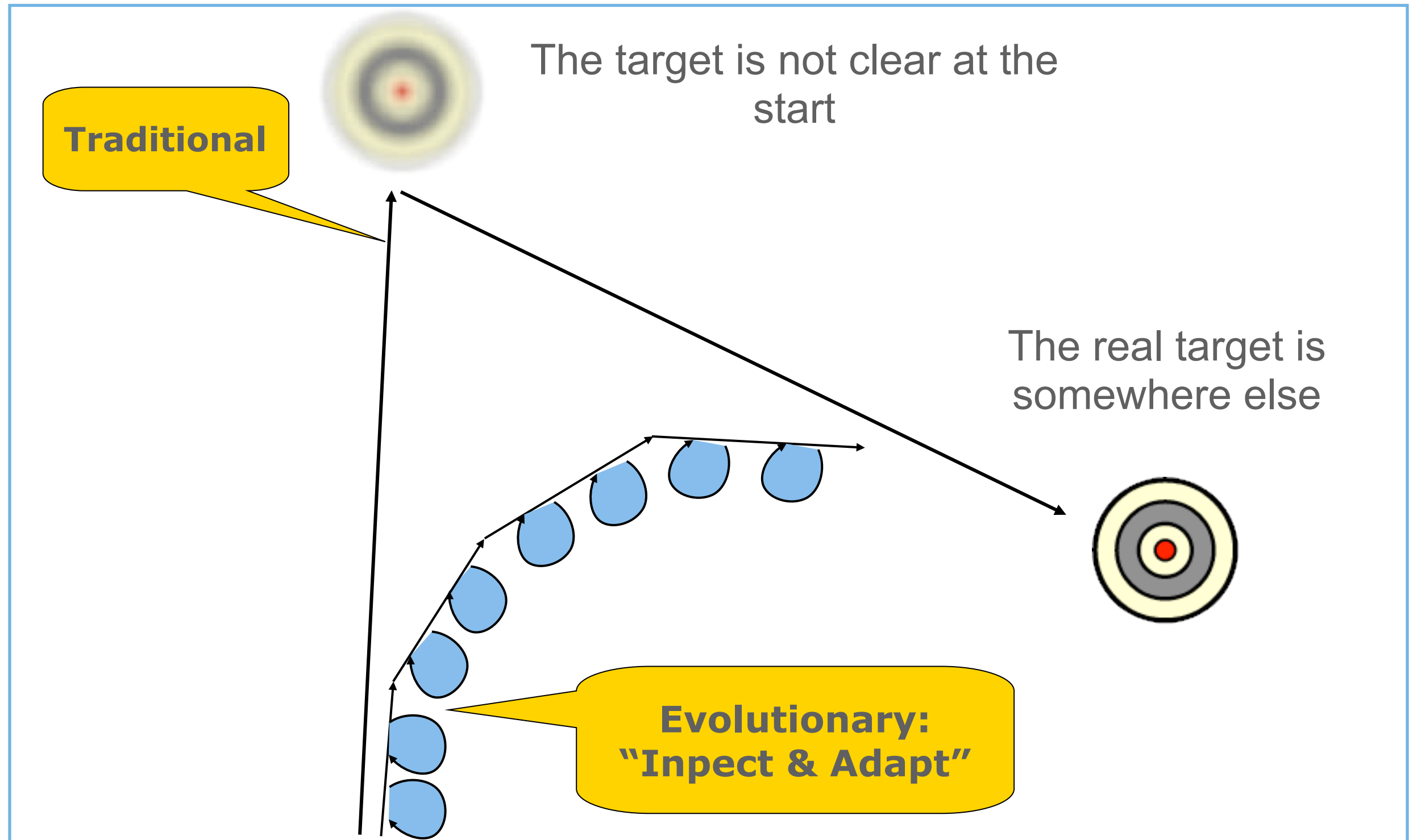
Agile Development



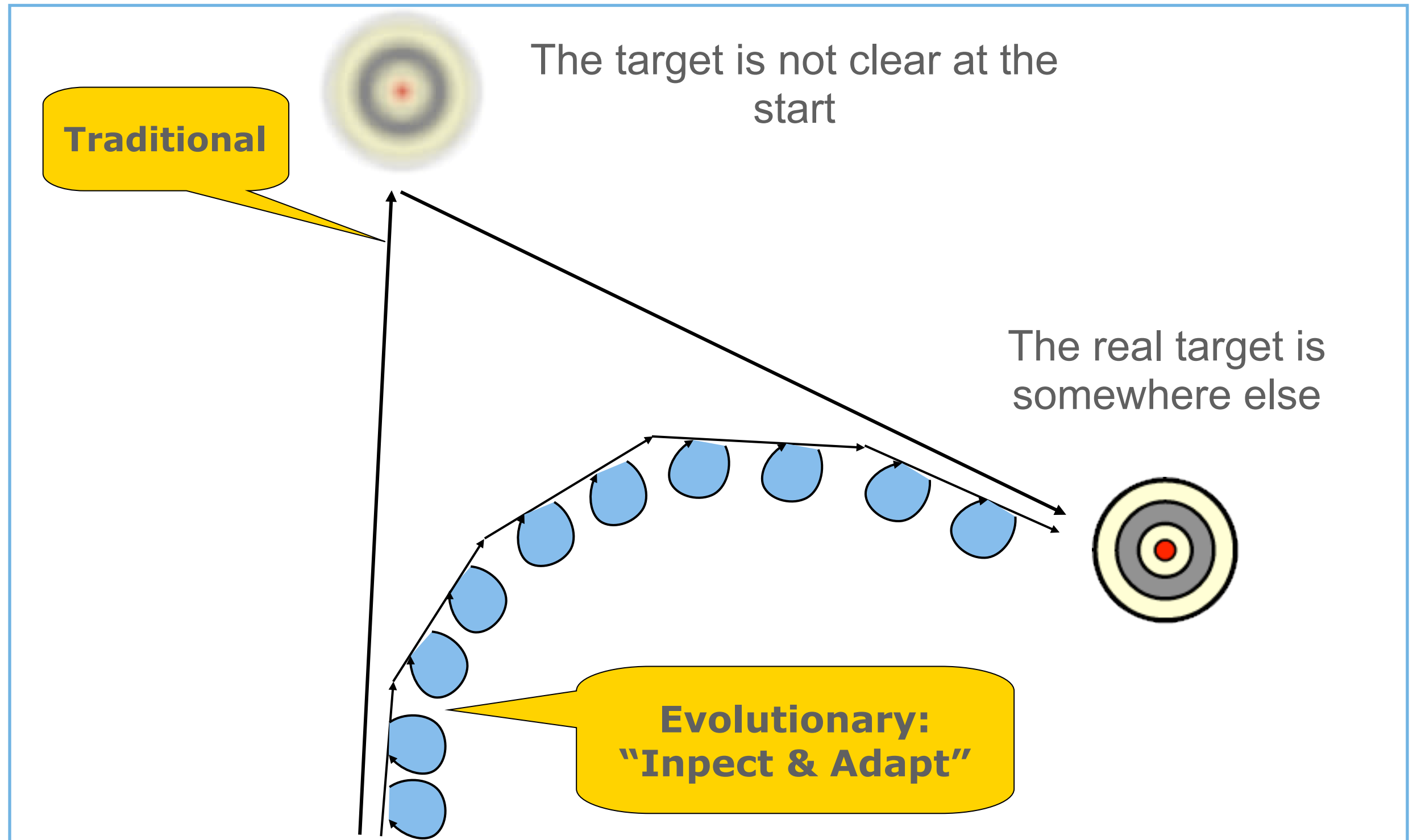
Agile Development



Agile Development



Agile Development



The Agile Manifesto

Individuals and interactions

over

Processes and tools

Running software

over

Comprehensive
documentation

Customer collaboration

over

Contract negotiation

Responding to change

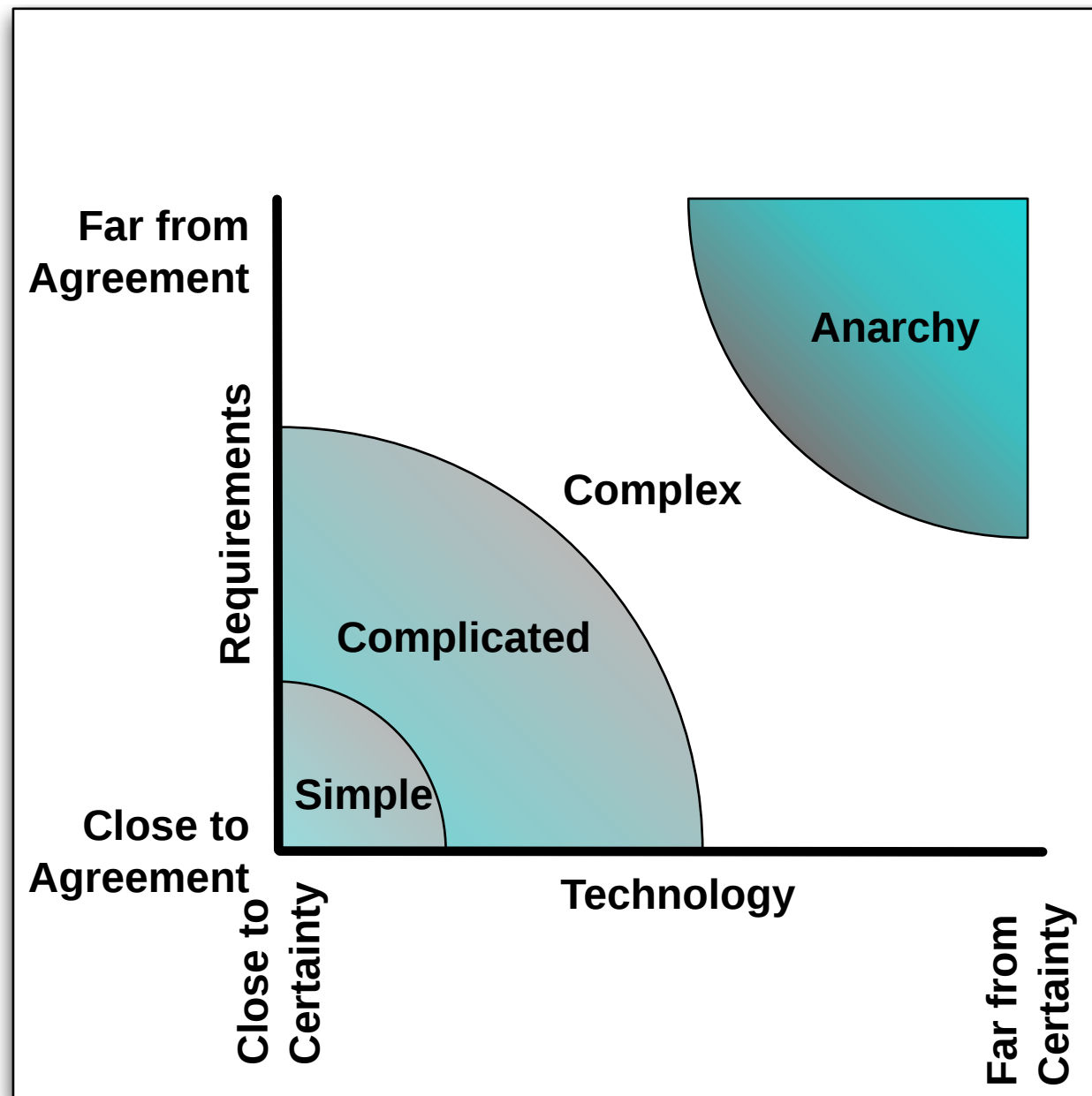
over

Following a plan

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

The Agile Manifesto Principles

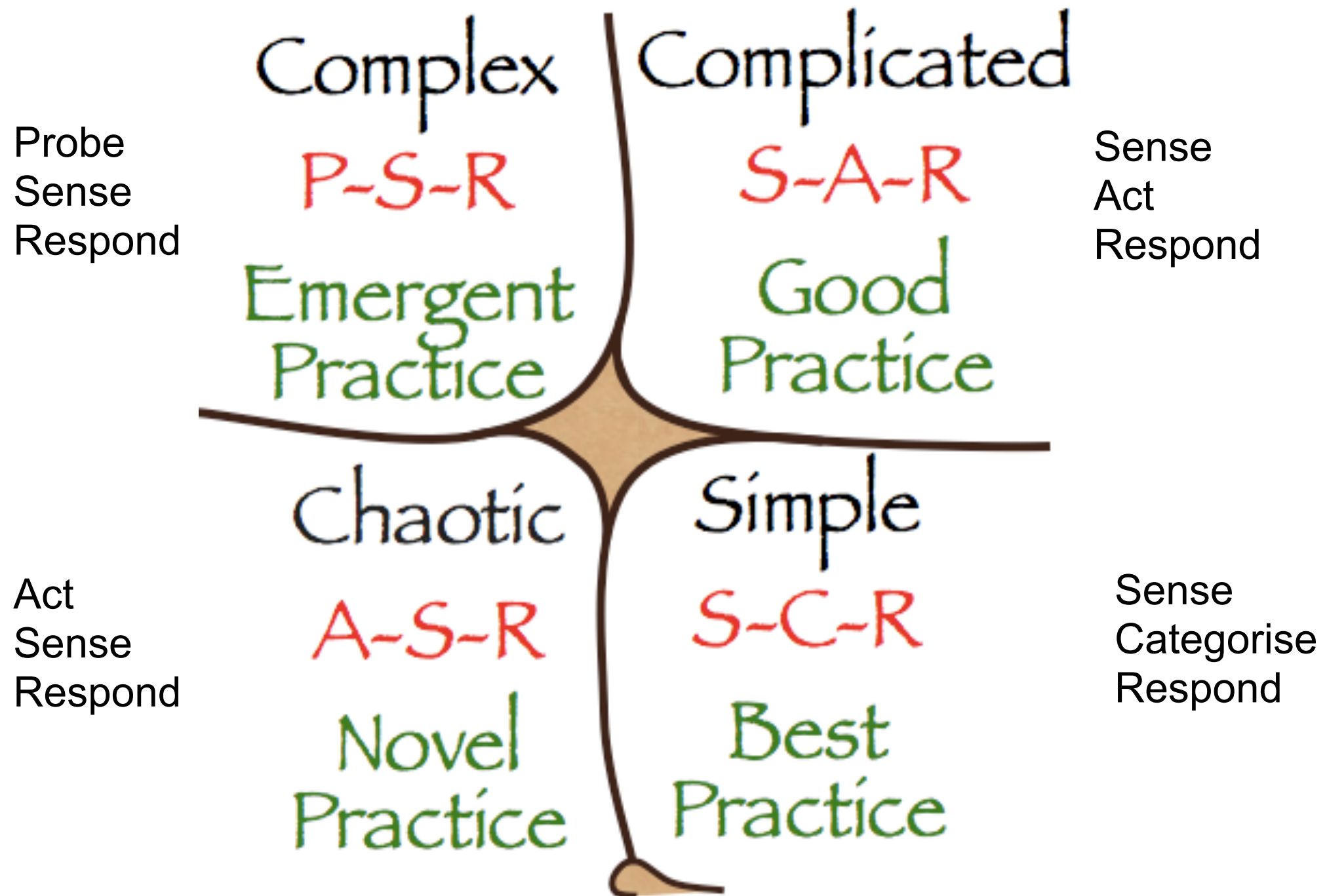
- ▶ Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- ▶ Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- ▶ Deliver working software frequently, from a couple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



Source: *Strategic Management and Organizational Dynamics* by Ralph Stacey in *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

In Which Region is Almost All Software Development?

The Cynefin Framework



Quelle: Wikipedia

Scrum in 5 Minutes

- ▶ The **Product Owner** is responsible for creating and prioritizing a list of open features. This is the **Product Backlog**, a complete but dynamic to-do list for the project.
- ▶ Before each **Sprint**, the team decides in a **Sprint Planning** meeting, how many of the highest prioritized features they can deliver during the Sprint. The team decides what tasks are necessary and enters them into the **Sprint Backlog**.
- ▶ During the Sprint, the team synchronizes in a **Daily Scrum** meeting and follow progress in the **burn down chart**.
- ▶ The **ScrumMaster** coaches the team, removes impediments and ensures that the team is able to work effectively.
- ▶ During the Sprint, valuable functionality is developed and a **potentially shippable product increment** created, which is demonstrated by the team during a **Sprint Review** meeting.
- ▶ At the end of every Sprint, the **Scrum Team** holds a **Retrospective** and identifies how they can work together more effectively during the next Sprint.

The Essentials

- ▶ Sprint

- ▶ An iteration with a fixed duration

- ▶ 3 Meetings

- ▶ Sprint Planning
 - ▶ Sprint Review
 - ▶ Daily Scrum

- ▶ 3 Documents

- ▶ Product Backlog
 - ▶ Sprint Backlog
 - ▶ Sprint Burn Down Chart

- ▶ 3 Roles in the Scrum Team

- ▶ Product Owner
 - ▶ ScrumMaster
 - ▶ Team

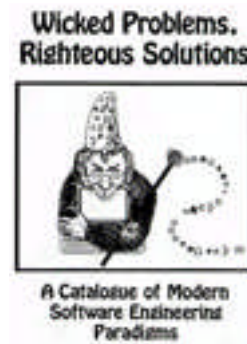


Manifesto for Agile Software Development

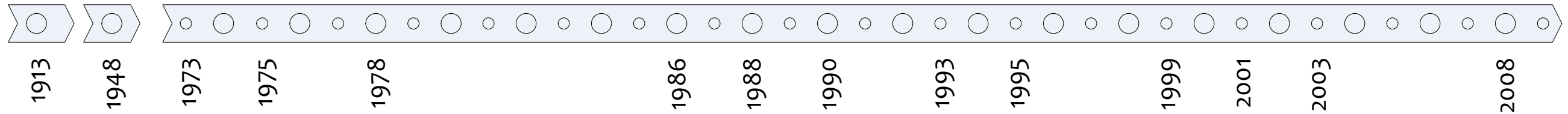
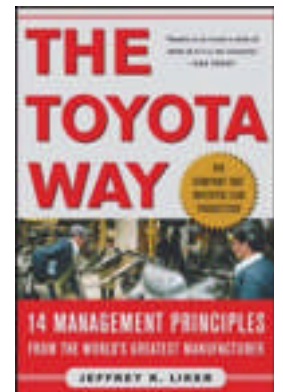
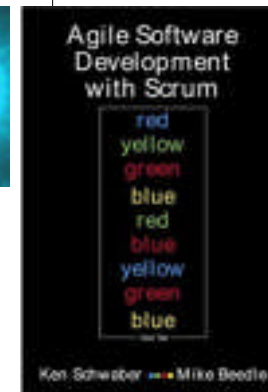
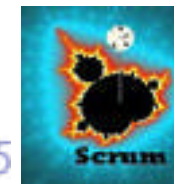
Wir enthüllen bessere Wege zur Softwareentwicklung, indem wir Software entwickeln und Anderen helfen, dies zu tun. Durch diese Arbeit haben wir Folgendes schätzen gelernt:

Individuen und Interaktionen mehr als Prozesse und Tools
Funktionierende Software mehr als umfassende Dokumentation
Zusammenarbeit mit Kunden mehr als Vertragsverhandlungen
Reaktion auf Änderungen mehr als einen Plan zu befolgen

Obwohl auch die Dinge auf der Rechten ihren Wert haben, schätzen wir die auf Linken höher ein.

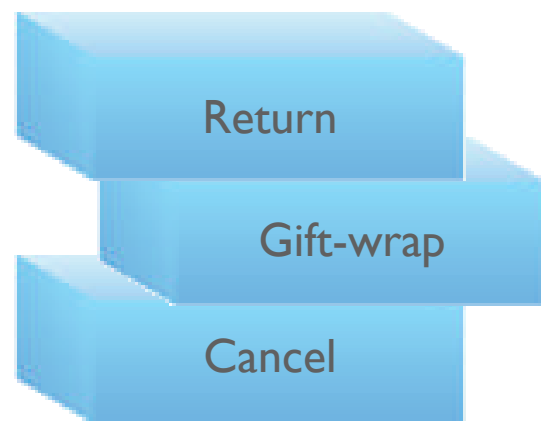


The First Scrum at Easel



The Roots of Scrum. Jens Korte, Syndato

Scrum Flow

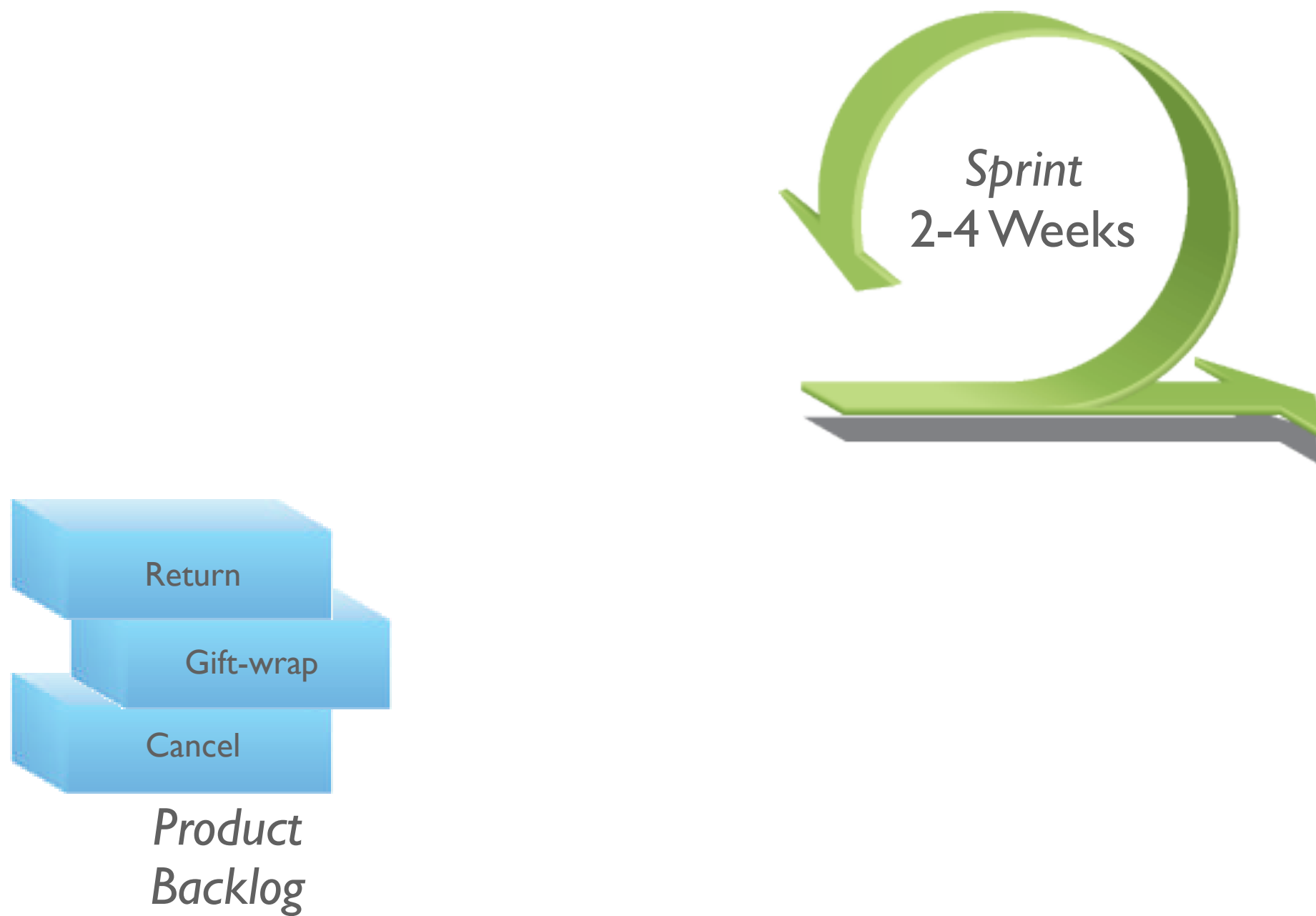


*Product
Backlog*



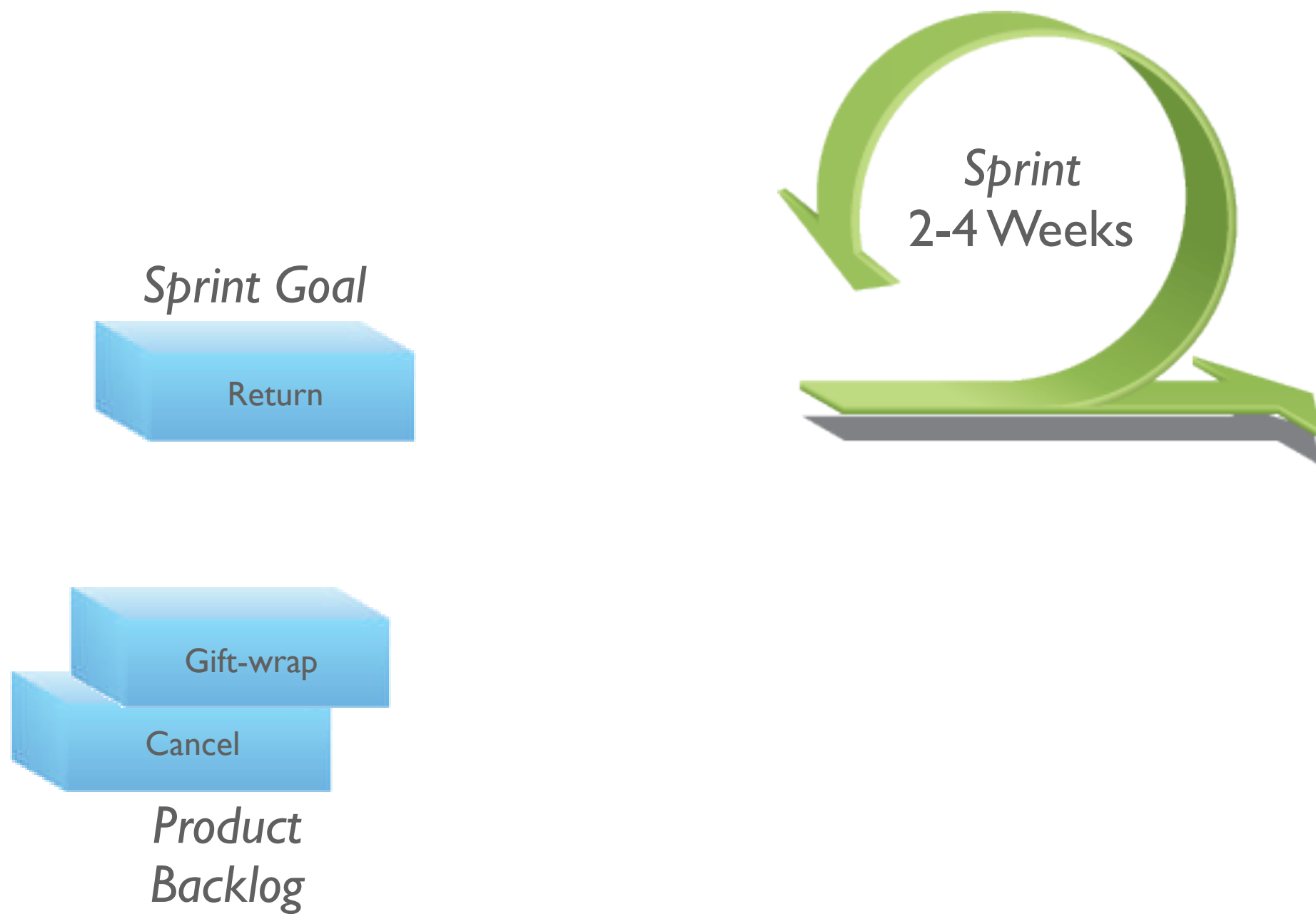
source:
Mountain Goat Software, LLC

Scrum Flow



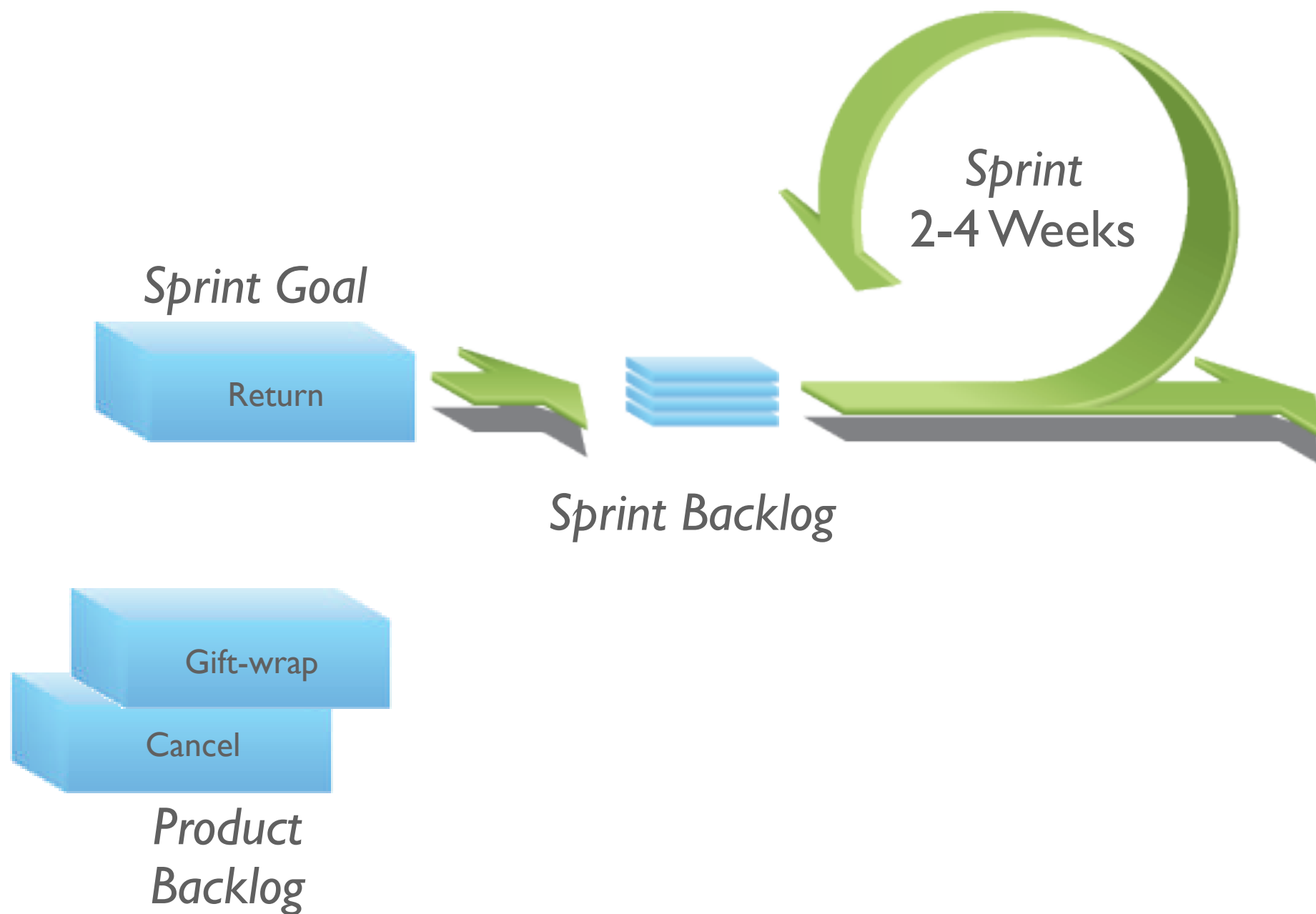
source:
Mountain Goat Software, LLC

Scrum Flow



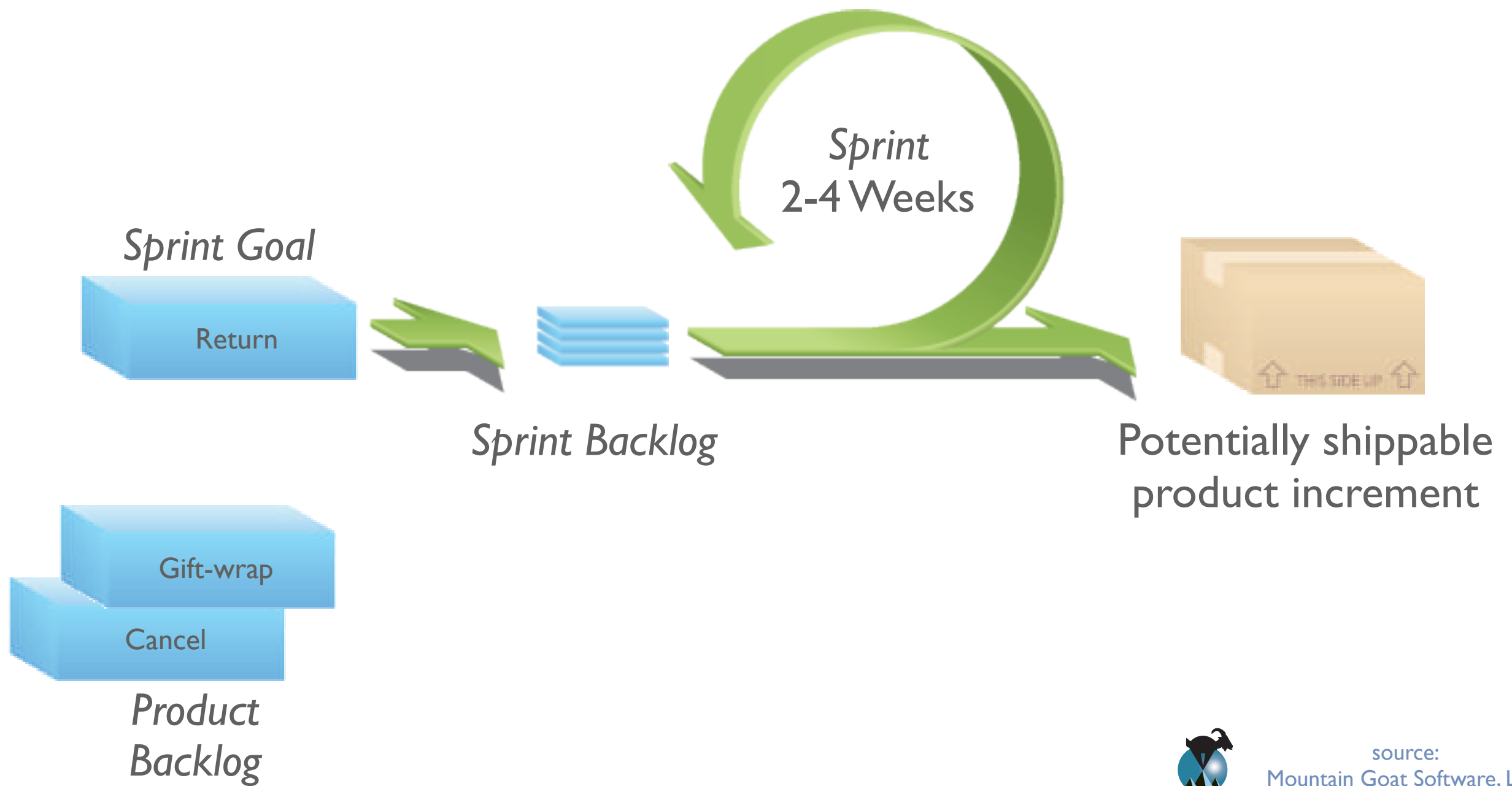
source:
Mountain Goat Software, LLC

Scrum Flow



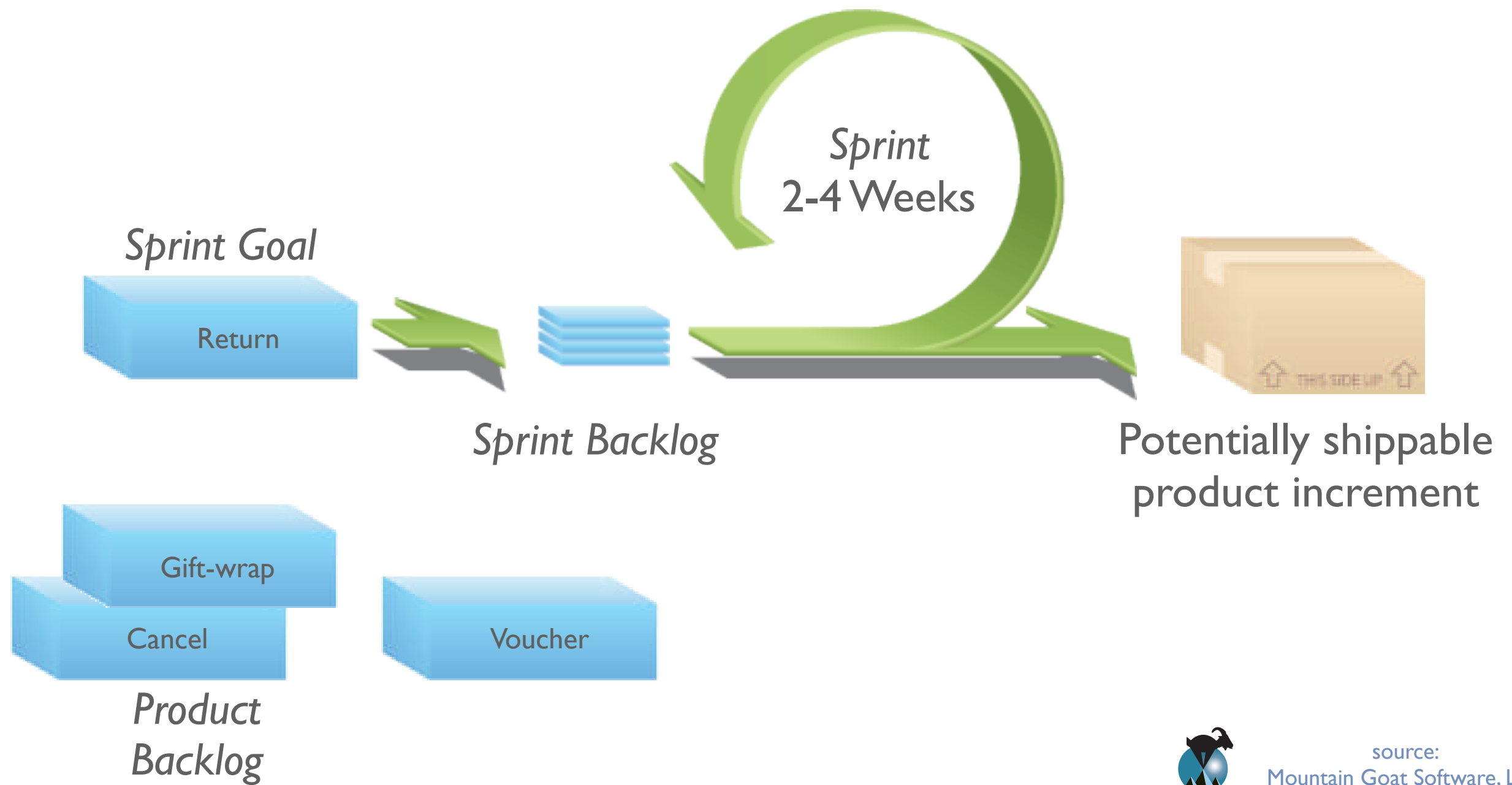
source:
Mountain Goat Software, LLC

Scrum Flow



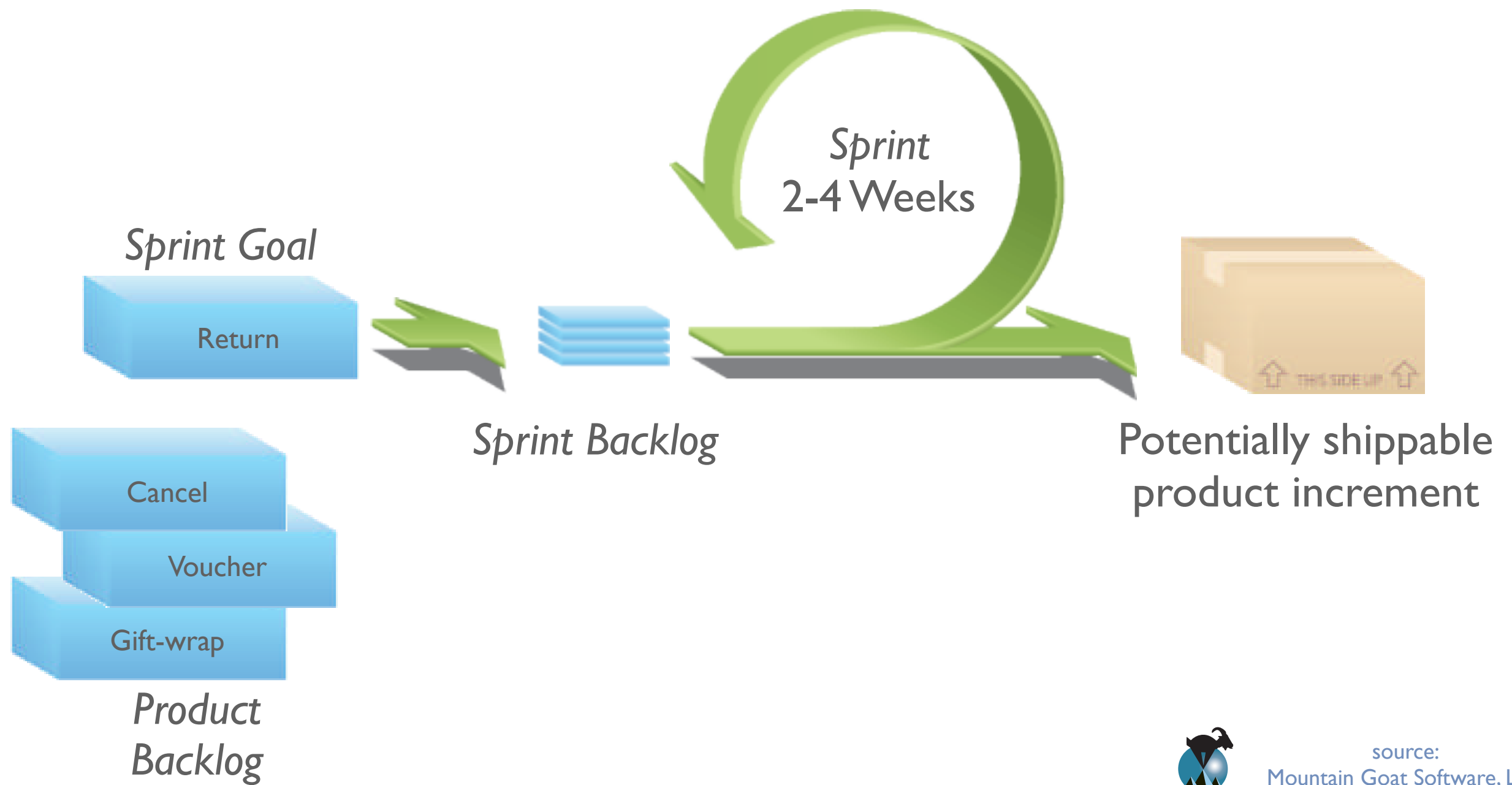
source:
Mountain Goat Software, LLC

Scrum Flow



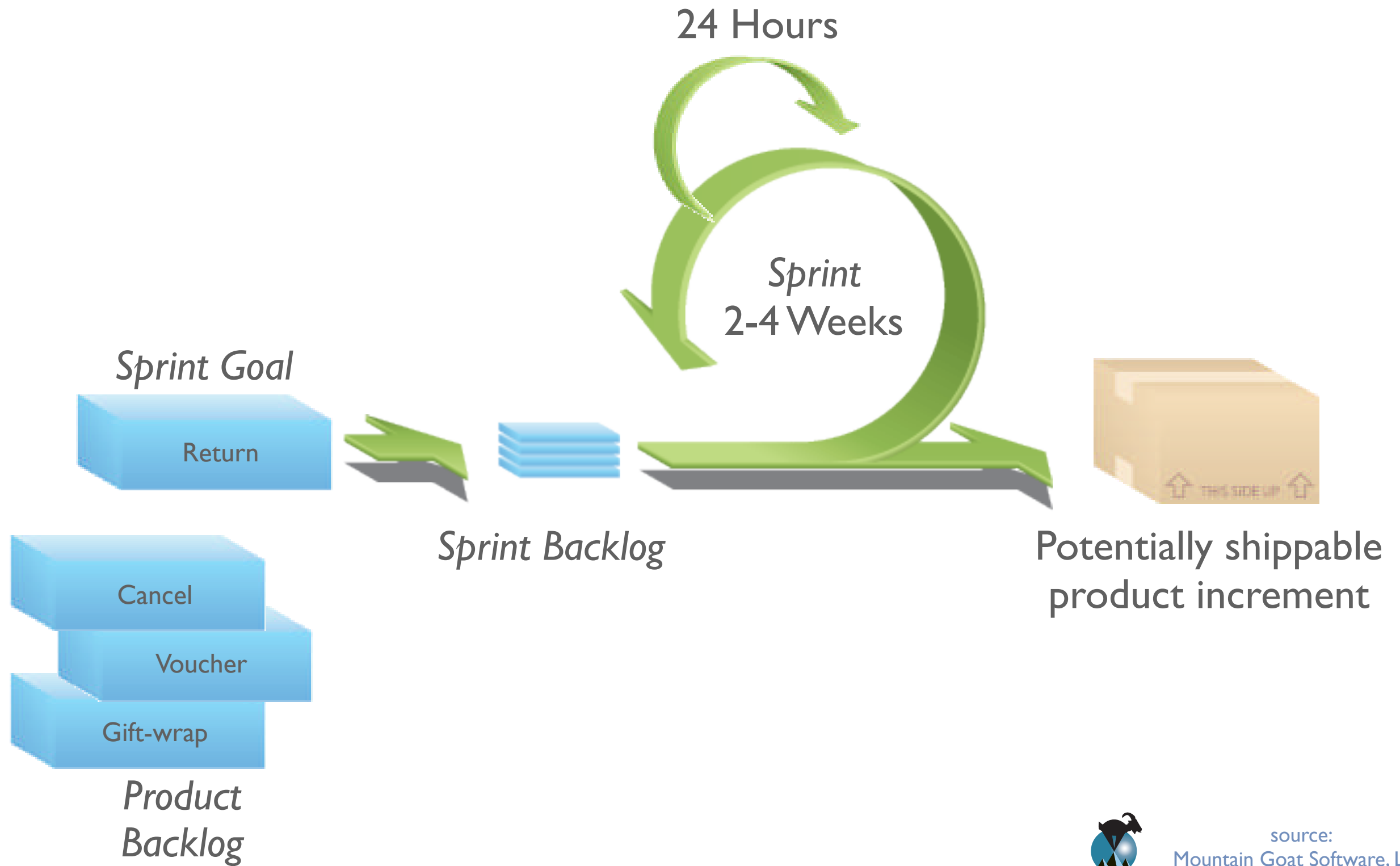
source:
Mountain Goat Software, LLC

Scrum Flow



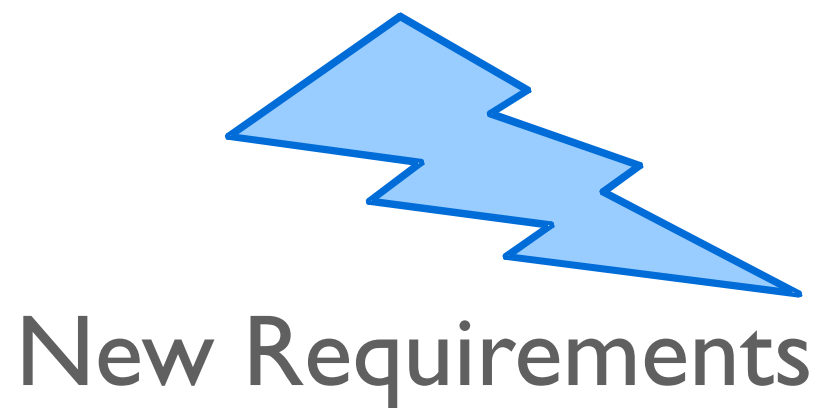
source:
Mountain Goat Software, LLC

Scrum Flow

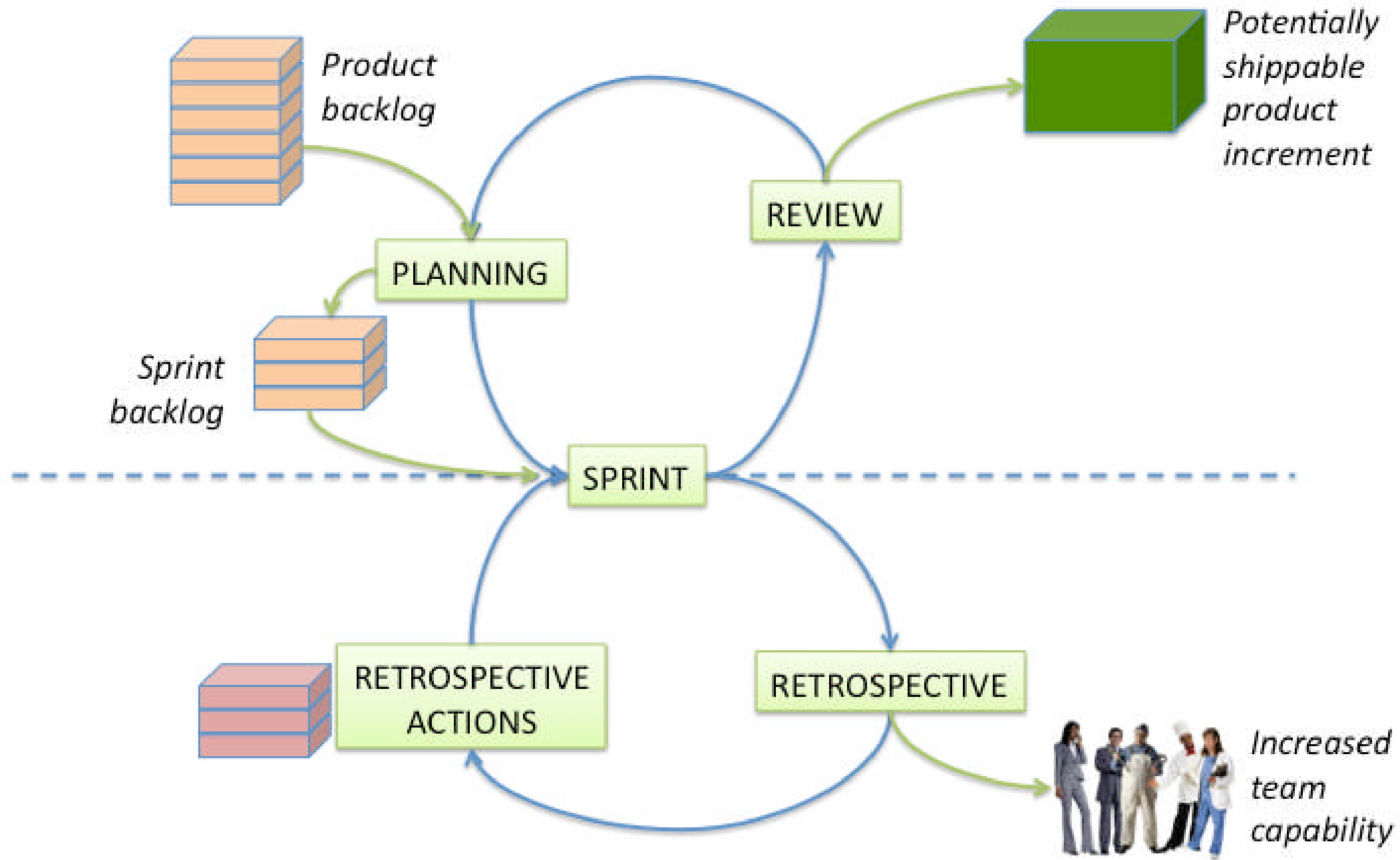


source:
Mountain Goat Software, LLC

No Changes during a Sprint



Product deliverable



Quelle: David Harvey

Automated Unit Testing and Test Driven Development

Automated Unit Testing

Unit Testing - Principles

- ▶ Test everything that could break
- ▶ Test everything that does break
- ▶ When a new bug is discovered, we first of all write a test that reproduces the bug
 - ▶ Ensures that the bug will not return
- ▶ Run the tests before every check in

What Should We Test?

- ▶ Are all of the results correct?
- ▶ Are the boundary values correct?
- ▶ Can we check the results through alternative means (independent verification of the algorithm)?
- ▶ Can we force error conditions?

Good Tests - “A TRIP”

(Hunt & Davis)

- ▶ Automatic
- ▶ Thorough
- ▶ Repeatable
- ▶ Independent
- ▶ Professional

Test Driven Development

TDD = Documentation and Design

“The act of writing a unit test is more an act of design than of verification. It is also more an act of documentation than of verification. The act of writing a unit test closes a remarkable number of feedback loops, the least of which is the one pertaining to verification of function”

Robert C. Martin

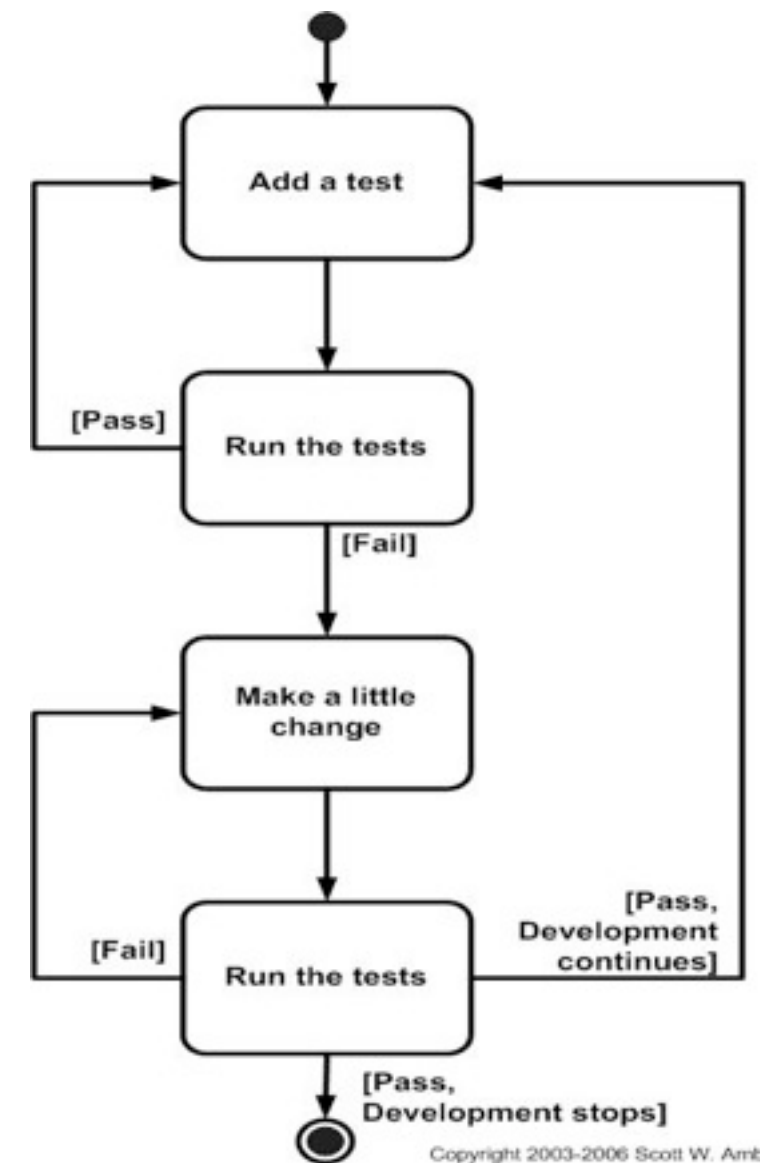
The Three Rules of TDD

- 1. You are not allowed to write any production code unless it is to make a failing unit test pass.*
- 2. You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.*
- 3. You are not allowed to write any more production code than is sufficient to pass the one failing unit test.*

Robert C. Martin

Test Driven Development

- ▶ Test Driven Development =
 - ▶ Test First Development + Refactoring



Advice

(Frank Westphal)

- ▶ Write test before you write the code, to make sure that the code is easy to test.
- ▶ The same quality criteria should apply to tests and production code: self-documenting, no duplicated code and as simple as possible.
- ▶ Don't test too much in each test method. Just one function/method in combination with one boundary condition at a time .
- ▶ Capture ideas straight-away, don't lose them (write a test!)
- ▶ Organize test classes effectively (suites, fixtures etc.)
- ▶ To test effectively, test cases must be executable in isolation from one another. Don't make assumptions about execution order. If there are some tests that are dependent on one another, combine them into a test case.
- ▶ Try to execute all unit tests after every compile.
- ▶ Keep at eye on return on investment when you are writing tests. Write only tests for which the effort is worthwhile.
- ▶ Refactor your test code as often and carefully as all other code.

Reading Recommendations

- ▶ Pragmatic Unit Testing in Java with JUnit
 - ▶ Andrew Hunt, David Thomas, The Pragmatic Programmers
- ▶ Softwaretests mit JUnit
 - ▶ Johannes Link, Dpunkt Verlag
- ▶ Testgetriebene Entwicklung mit JUnit und FIT
 - ▶ Frank Westphal, Dpunkt Verlag

SOLID

- ▶ Single Responsibility Principle
 - ▶ An object should have only a single responsibility
- ▶ Open Closed Principle
 - ▶ Software should be open for extension, but closed for modification
- ▶ Liskov Substitution Principle
 - ▶ Objects should be replaceable with instances of their subtypes without altering the correctness of that program
- ▶ Interface Segregation Principle
 - ▶ Many client specific interfaces are better than one general purpose interface
- ▶ Dependency Inversion Principle
 - ▶ Depend upon Abstractions. Do not depend upon concretions

Source: wikipedia

Code Craftsmanship

craftsmanship

crafts·man n. (pl. -men) a person who is skilled in a particular craft. an artist. crafts·man·ship n.



The Boy Scout Rule

**Leave the
campground
cleaner than you
found it.**



Robert C. Martin Series

Clean Code

A Handbook of Agile Software Craftsmanship

Foreword by James O. Coplien

Robert C. Martin

Clean Code and Software Craftsmanship

Craftsmanship over Execution

**Most software development teams
execute, but they don't take care.
We value execution, but we value
craftsmanship more.**

Robert Martin

Continuous Self Improvement



**there is just one deadly
sin:
standing still**

Clean Code Developer



The screenshot shows a web browser window with the address bar displaying `http://clean-code-developer.de/`. The browser's address bar also shows the text "clean-code-developer" and a search bar with "Google". The browser's tabs include "Community | scr...", "Alliance - transfo...", "Anmelden | Faceb...", "http://apps.faceb...", and "clean-code-devel...".

The website's header features a logo consisting of three interlocking rings (black, orange, and yellow) with the text "clean-code-developer.de" on them. To the right of the logo, the text "clean-code-developer.de" is displayed in a large, bold, black font, followed by the tagline "Prinzipien, Regeln und Praktiken für bessere Software" in a smaller, black font.

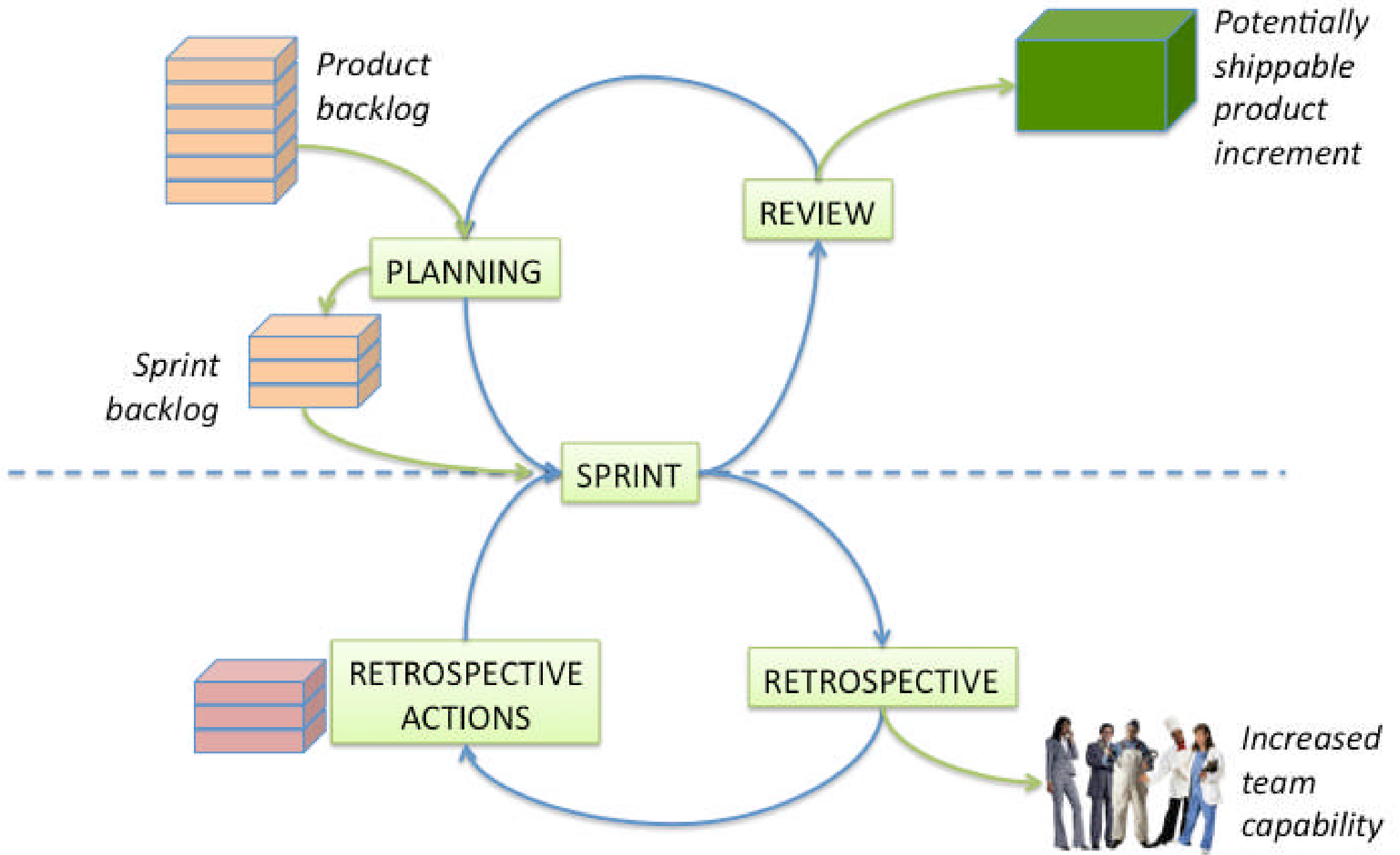
Below the header, there are navigation links: "Login", "Help/Guide", "About Trac", and "Preferences". A horizontal bar contains two tabs: "Wiki" (selected) and "Timeline". Below this bar, there are more navigation links: "Start Page", "Index", "History", and "Last Change".

The main content area is titled "Clean Code Developer (CCD)" in a bold, black font. Below the title, the text "Professionalität = Bewusstheit + Prinzipien" is displayed. The main text describes the concept of professional software development, stating that it is not about money but about a certain mindset and principles. It mentions the "CcdTeam" and lists the "CCD-Grade" (0. Schwarz, 1. Rot, 2. Orange, 3. Gelb, 4. Grün, 5. Blau, 6. Weiß) and the "Armband" (Der CCD).

On the right side of the main content area, there is a yellow sidebar with a list of links: "Homepage", "Das Wertesystem", "Die Umfragen", "Die CCD-Grade", "0. Schwarz", "1. Rot", "2. Orange", "3. Gelb", "4. Grün", "5. Blau", "6. Weiß", "Das Armband", and "Der CCD".

Agile Skills

Product deliverable



Capability deliverable

Quelle: David Harvey

Keep the Rhythm

Retrospective

- For you and your team
- Identify impediments to improvement

Sprint Review

- Report constructively and transparently

Daily Scrum

- Keep synchronized with your team

What is Your Contribution?

Stand up and say something

Be courageous

**Recognize when something isn't right
with your team**

Practice!

Take part in the agile community

Appendix

Bibliography, Copyright, Attribution,
Community and Contact Information

Bibliography

- ▶ Coens, T. & Jenkins, M., 2002. *Abolishing Performance Appraisals: Why They Backfire and What to Do Instead*, Berrett-Koehler Publishers.
- ▶ Cohn, M., 2004. *User Stories Applied: For Agile Software Development*, Addison Wesley.
- ▶ Cohn, M., 2005. *Agile Estimating and Planning*, Prentice Hall.
- ▶ Cohn, M., 2009. *Succeeding with Agile - Software Development using Scrum*, Addison Wesley.
- ▶ Derby, E. & Larsen, D., 2006. *Agile Retrospectives: Making Good Teams Great*, The Pragmatic Bookshelf.
- ▶ Feathers, M., 2007. *Working Effectively with Legacy Code*, Prentice Hall PTR.
- ▶ Freeman, S. & Pryce, N., 2009. *Growing Object-Oriented Software, Guided By Tests*, Addison Wesley.
- ▶ Gloger, B., 2008. *Scrum. Produkte zuverlässig und schnell entwickeln*, Hanser.
- ▶ Grenning, J.W., 2010. *Test Driven Development for Embedded C*, The Pragmatic Bookshelf.
- ▶ Martin, R.C., 2008. *Clean Code: A Handbook of Agile Software Craftsmanship*, Prentice Hall PTR.
- ▶ Pichler, R., 2007. *Scrum - Agiles Projektmanagement erfolgreich einsetzen*, dpunkt.verlag.
- ▶ Takeuchi, H. & Nonaka, I., 1986. The new new product development game, *Harvard Business Review*, January-February 1986.
- ▶ Westphal, F., 2005. *Testgetriebene Entwicklung mit JUnit & FIT*, dpunkt.verlag.

The Scrum Alliance

- ▶ <http://scrumalliance.org>
- ▶ A not-for-profit corporation that encourages the adoption of Scrum and licenses trainers and registered education providers (such as ScrumCenter GmbH and its founders) to deliver various training programmes, including:
 - ▶ Certified ScrumMaster (CSM)
 - ▶ Certified Scrum Product Owner (CSPO)
 - ▶ Certified Scrum Developer (CSD)
- ▶ Membership is open to CSMs, CSPOs and CSDs
- ▶ Regular “Scrum Gathering” conferences worldwide

Selected Scrum User Groups

- ▶ Agile Saxony - Scrum User Group Saxony
 - ▶ <http://agilesaxony.org>
 - ▶ Regular meetings in Dresden and Leipzig
- ▶ Agile Tuesday - Scrum User Group Munich
 - ▶ <http://agiletuesday.org>
 - ▶ Meets regularly (normally on the second Tuesday of each month)
- ▶ Scrum User Group Germany
 - ▶ <http://scrumaufdeutsch.pbworks.com/>
 - ▶ German speaking user group covering D-A-CH
 - ▶ Annual 1-day Open-Space conference

Copyright and Attribution

Except where otherwise noted:

© 2010 Simon Roberts and Christoph Mathis, all rights reserved.

The Scrum flow animation is taken from
Mike Cohn: A redistributable introduction to Scrum
<http://www.mountaingoatsoftware.com/scrum-a-presentation>

Contact Information

▶ Simon Roberts

- ▶ simon.roberts@scrumcenter.com
- ▶ mobile +49 160 8082012, twitter @srob

▶ ScrumCenter GmbH

- ▶ <http://scrumcenter.com>
- ▶ tel. +49 89 5003520



ScrumCenter
Next level working.