Agile Practices in Industry

Ralph Jocham
effective agile.
ralph.jocham@effectiveagile.com
Ralph Jocham

- Started as programmer; discovered process as a problem early on
- First Unified Process with UML
- Agile since 2000, Scrum since 2003
- Did come around, different cultures and domains
- Founder of effective agile.
- Trainer with Scrum.org, PSD, PSF, PSM, PSCP, SPS, MGT 3.0

+41 79 448 6388
ralph.jocham@effectiveagile.com
Thank You for Being Late

AN OPTIMIST’S GUIDE TO THRIVING IN THE AGE OF ACCELERATIONS

THOMAS L. FRIEDMAN

AUTHOR OF THE WORLD IS FLAT
You are the technology Future
Why?
Now Every Company Is A Software Company

Techonomy
How Tech Transforms Business and Society FULL BIO
Opinions expressed by Forbes Contributors are their own.

David Kirkpatrick, Contributor

Ford sells computers-on-wheels. McKinsey hawks consulting-in-a-box. FedEx boasts a developer skunkworks. The era of separating traditional industries and technology industries is over—and those who fail to adapt right now will soon find themselves obsolete.

The future of big industry can be seen through a little technology company in Silicon Valley. Sunnyvale-based Picarro, with 90 employees, has developed a highly accurate mobile measurement device for gasses, with related software that turns the tool’s output into an easy-to-understand visualization. Any organization or individual can drive Picarro’s $70,000 instrument around and create stunningly detailed maps documenting, say, point-by-point

Why Google could become the Amazon of banking

Internet giants will not replace banks but will disrupt the sector through other services taking consumers away from traditional banks

Schweizer Banken blockieren Apple Pay


Rapid Application Development the Zappos Way

Amazon also releases software every 11.6 seconds and is the leader in public cloud infrastructure.

Source: Forrester - State Of Scaling Agile In The Age Of The Customer (2014)

Source: The Economist “Organizational agility: how businesses can survive and thrive in turbulent times” (2009)
Scrum Is A Major Management Discovery

A reader remarked about my recent article on Salesforce.com, “Frankly, this looks like a plug for Scrum.” Well, yes. If there was a Nobel Prize for management, and if there was any justice in the world, I believe that the prize would be awarded, among others, to Jeff Sutherland, Ken Schwaber and Mike Cohn for their contributions to the invention of Scrum. Until recently, like most people, I had never heard of Scrum. This is not surprising, as it is rarely mentioned in general management textbooks or discussed in business schools. I came across Scrum several years ago, almost by accident.
effective agile.
The Red Queen's race is an incident that appears in Lewis Carroll's *Through the Looking-Glass and involves the Red Queen, a representation of a Queen in chess, and Alice constantly running but remaining in the same spot.*

"Well, in our country," said Alice, still panting a little, "you'd generally get to somewhere else—if you run very fast for a long time, as we've been doing."

"A slow sort of country!" said the Queen. "Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"
Linear
Plan

Analyze

Design

Code

Test

Release
Work is organized by activity?

→ Big Batches
Scientific Management

Is this the right approach for software development?
Scientific Management
Complexity
Stacey Graph

- Simple
- Complicated
- Complex
- Chaos

- Requirements: close to agreement, far from agreement
- Technology: close to certainty, far from certainty

Empirical

- Complex: Probe, Sense, Respond
- Complicated: Sense, Analyze, Respond
- Chaos: Act, Sense, Respond
- Simple: Sense, Categorize, Respond

(Defined)

- Complicated: Good Practices
- Simple: Best Practices

(source: Dave Snowden, IBM)

(source: Ralph Stacey, University of Herfordshire)

effective agile.
Demming Cycle
35% of Requirements Change
>60% of features are rarely or never used

(source: Standish Group)
Value

Welcoming change/
Coping with Complexity

Original Vision

Additional Value

Sprint 1
Sprint 2
Sprint n-1
Sprint n
Sprint n+1

effective agile.
Iterative & Incremental
MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS
Dr. Winston W. Royce

INTRODUCTION
I am going to describe my personal views about managing large software developments. I have had various assignments during the past nine years, mostly concerned with the development of software packages for spacecraft mission planning, commanding and post-flight analysis. In these assignments I have experienced different degrees of success with respect to arriving at an operational state, on time, and within costs. I have become prejudiced by these experiences and I am going to relate some of these prejudices in this presentation.

COMPUTER PROGRAM DEVELOPMENT FUNCTIONS
There are two essential steps common to all computer program developments, regardless of size or complexity. There is first an analysis step, followed second by a coding step as depicted in Figure 1. This sort of very simple implementation concept is in fact all that is required: if the effort is sufficiently small and if the final product is to be operated by those who built it — as is typically done with computer programs for internal use. It is also the kind of development effort for which most customers are happy to pay, since both steps involve genuinely creative work which directly contributes to the usefulness of the final product. An implementation plan to manufacture larger software systems, and keyed only to these steps, however, is doomed to failure. Many additional development steps are required, none contribute directly to the final product as analysis and coding, and all driven up the development costs. Customer personnel typically would rather not pay for them, and development personnel would rather not implement them. The prime function of management is to see these concepts to both groups and then enforce compliance on the part of development personnel.

![Diagram](image)

Figure 1. Implementation steps to deliver a small computer program for internal operations.

A more grandiose approach to software development is illustrated in Figure 2. The analysis and coding steps are still in the picture, but they are preceded by two levels of requirements analysis, are separated by a program design step, and followed by a testing step. These additional steps are treated separately from analysis and coding because they are distinctly different in the way they are executed. They must be planned and staffed differently for best utilization of program resources.

Figure 3 portrays the iterative relationship between successive development phases for this scheme. The ordering of steps is based on the following concept: that as each step progresses and the design is further detailed, there is an iteration with the preceding and succeeding steps but rarely with the more remote steps in the sequence. The virtue of all of this is that as the design proceeds the change process is slowed down to manageable limits. At any point in the design process after the requirements analysis is completed there exists a firm and closeup, moving baseline to which the system in the event of unforeseen design difficulties. What we have in an effective fallback position that tends to maximize the extent of early work that is salvageable and preserved.

Copyright © 1970 by The Institute of Electrical and Electronics Engineers, Inc. Originally published by TRW.
Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

I believe in this concept, but the implementation described above is risky and invites failure. The problem is illustrated in Figure 4. The testing phase which occurs at the end of the development cycle is the effective agile.
$37B worth of DoD projects using 2167A

- Required extensive rework to meet true needs. 20%
- Never used. Egregiously failed to meet needs. 46%

MIL-STD-4987
Defined vs Emperical

**Waterfall (Defined)**
Plan for the entire project up-front

**Scrum (Empirical)**
Plan a little for the entire project and then a little for each Sprint

(source: effective agile)
Higher Chance of Success

Defined vs. Empirical

Low Complexity

Medium Complexity

High Complexity

Probablity of Success

Effective Agile

(Adapted from ADM)

4-Oct-17
Why a higher Change?

Visibility

Ability to Change

Business Value

Risk

Time

Not Scrum, but sound engineering practices like XP

Waterfall

Scrum

effective agile.
Scrum
A simple Definition of Scrum

• Scrum (n): A framework within which people can address complex problems, and productively and creatively develop products of the highest possible value.

(source: ADM)
Framework

<table>
<thead>
<tr>
<th>Roles</th>
<th>Artifacts</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Product Owner</td>
<td>• Increment</td>
<td>• Sprint</td>
</tr>
<tr>
<td>• Dev Team</td>
<td>• Product Backlog</td>
<td>• Sprint Planning</td>
</tr>
<tr>
<td>• Scrum Master</td>
<td>• Sprint Backlog</td>
<td>• Daily Scrum</td>
</tr>
</tbody>
</table>

(source: ADM)

effective agile.
Sprint Goal

Why

Ready
- Size
- Estimate
- Acceptance Criteria

How
Refinement

What

15 Minutes

8 Hours

1-4 Weeks

4 Hours

3 Hours

DONE

Plan

Analysis

Design

Implement

Test

Release

©2015 effective agile.
Scrum Values

- Commitment
- Courage
- Focus
- Respect
- Openness

effective agile.