Empiricism
with
Scrum

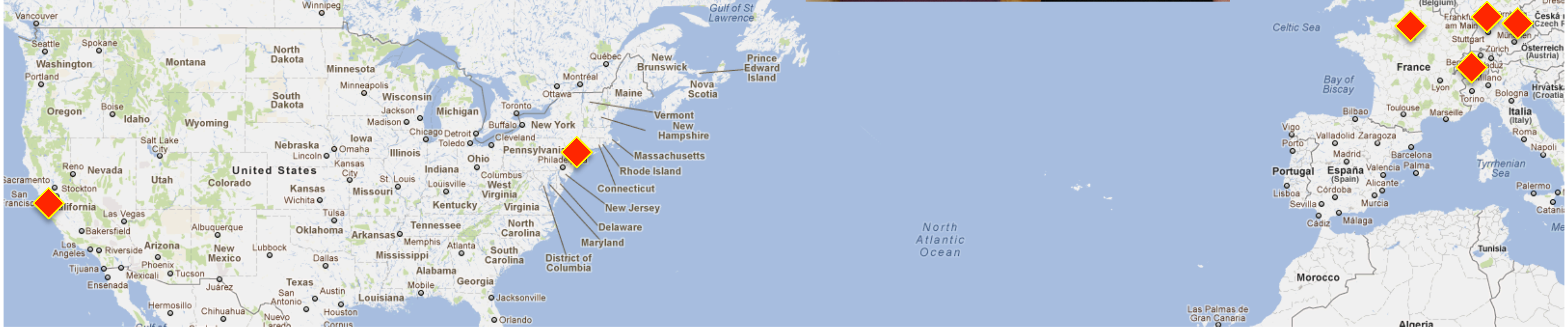
Ralph Jocham effective agile. ralph.jocham@effectiveagile.com

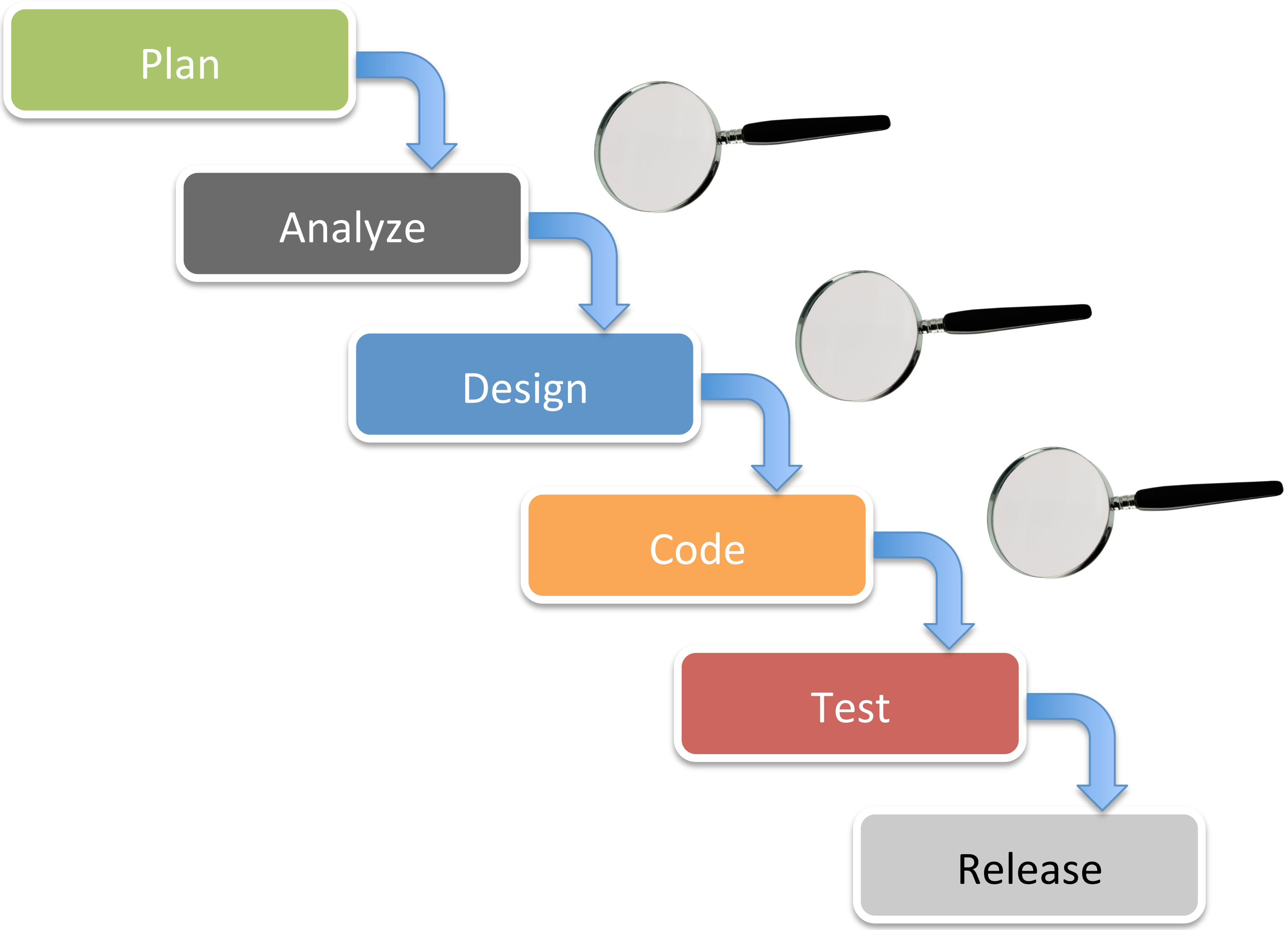
Ralph Jocham

- Started as programmer; discovered process as a problem early on
- First Unified Process with UML
- Agile since 2000 with XP
- Scrum in 2003
- Oracle, LinkedIn, Roche, Google, The Gap, Swisscom, Texas Instruments, Siemens Medical, ThoughtWorks, JPMorganChase
- Did come around, different cultures and domains
- Founder of `effective agile.`
- Trainer with Scrum.org

Professional
Scrum Trainer

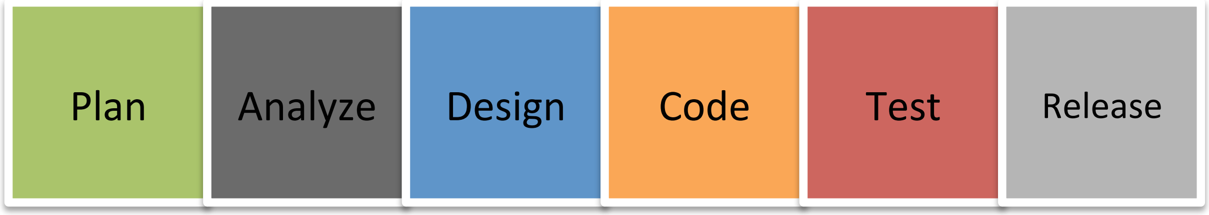
training and certification programs by Scrum.org™





Plan
Analyze
Design
Code
Test
Release

Work is organized by activity



Plan

Analyze

Design

Code

Test

Release

➔ Big Batches

Big Batches

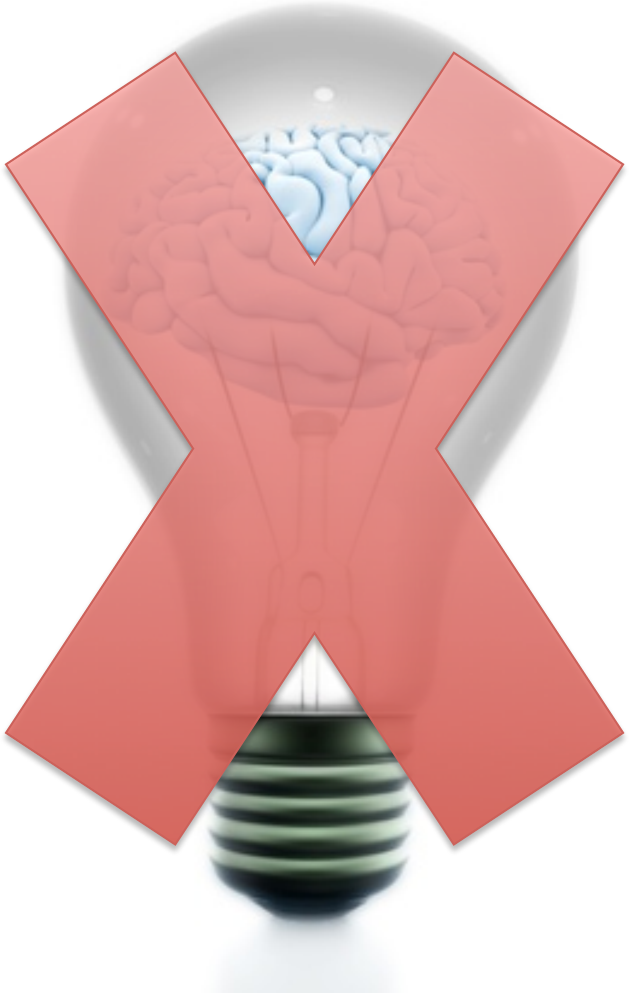
Is this the right approach for software development?

Scientific Management

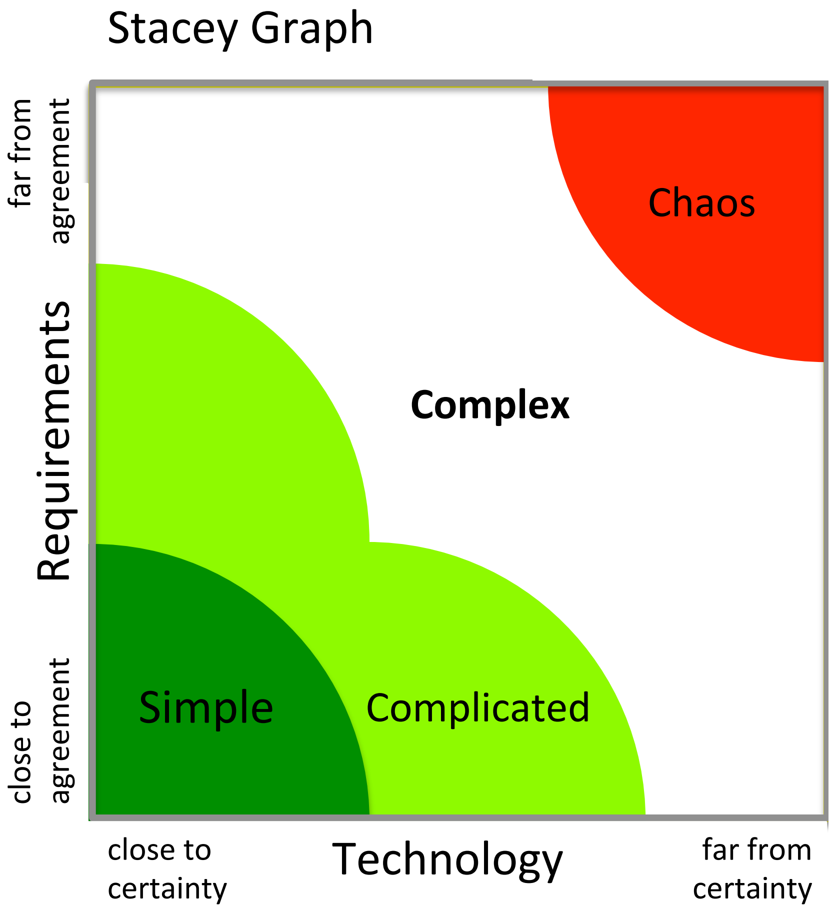
Is this the right approach for software development?

Command and Control

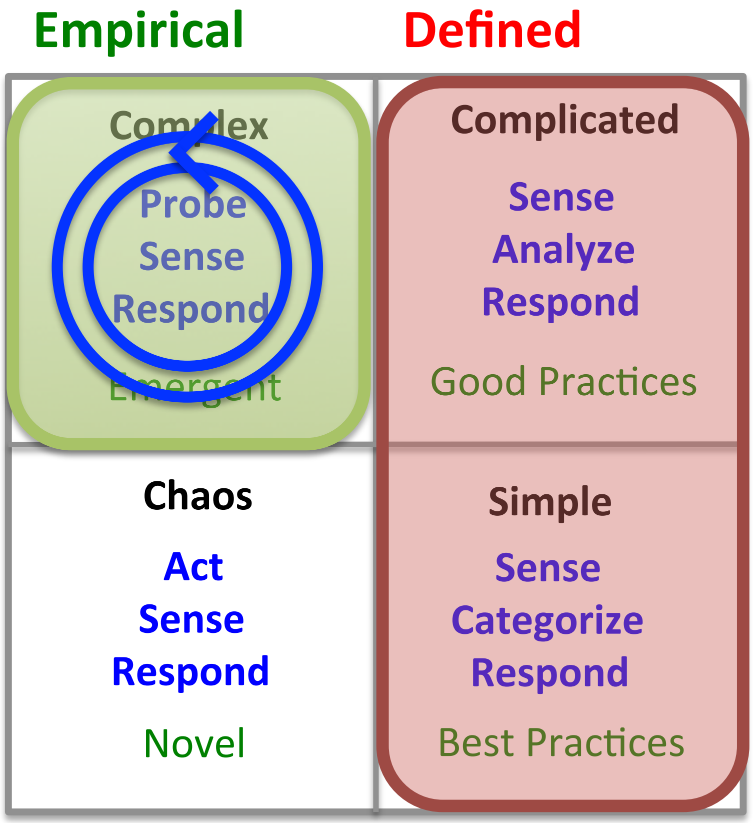
Scientific Management



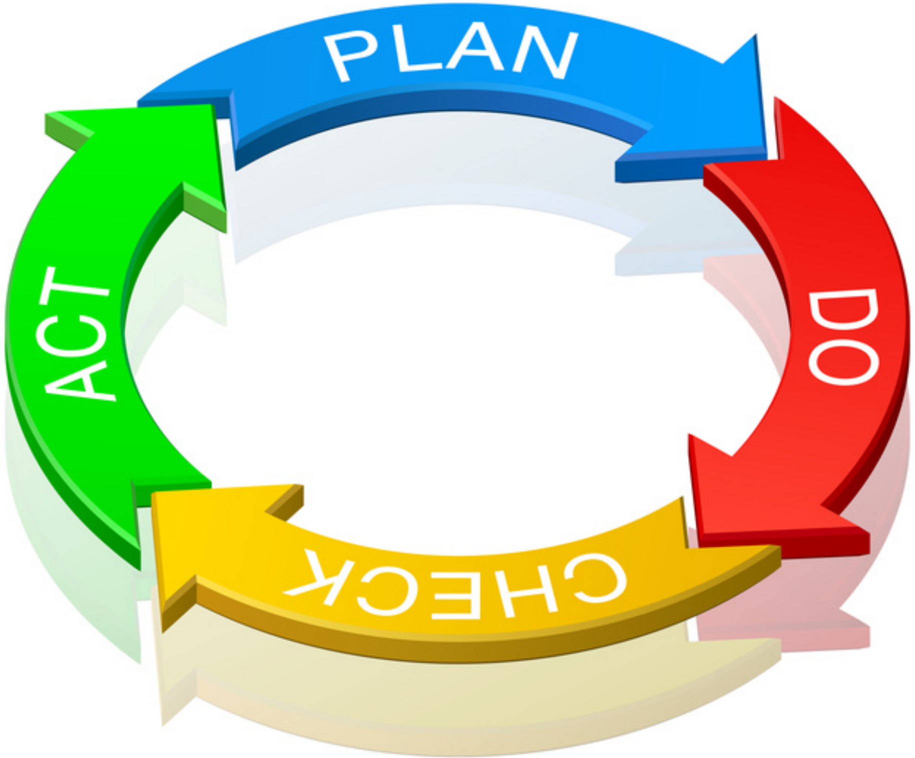
Complexity



(source: Ralph Stacey, University of Herfordshire)

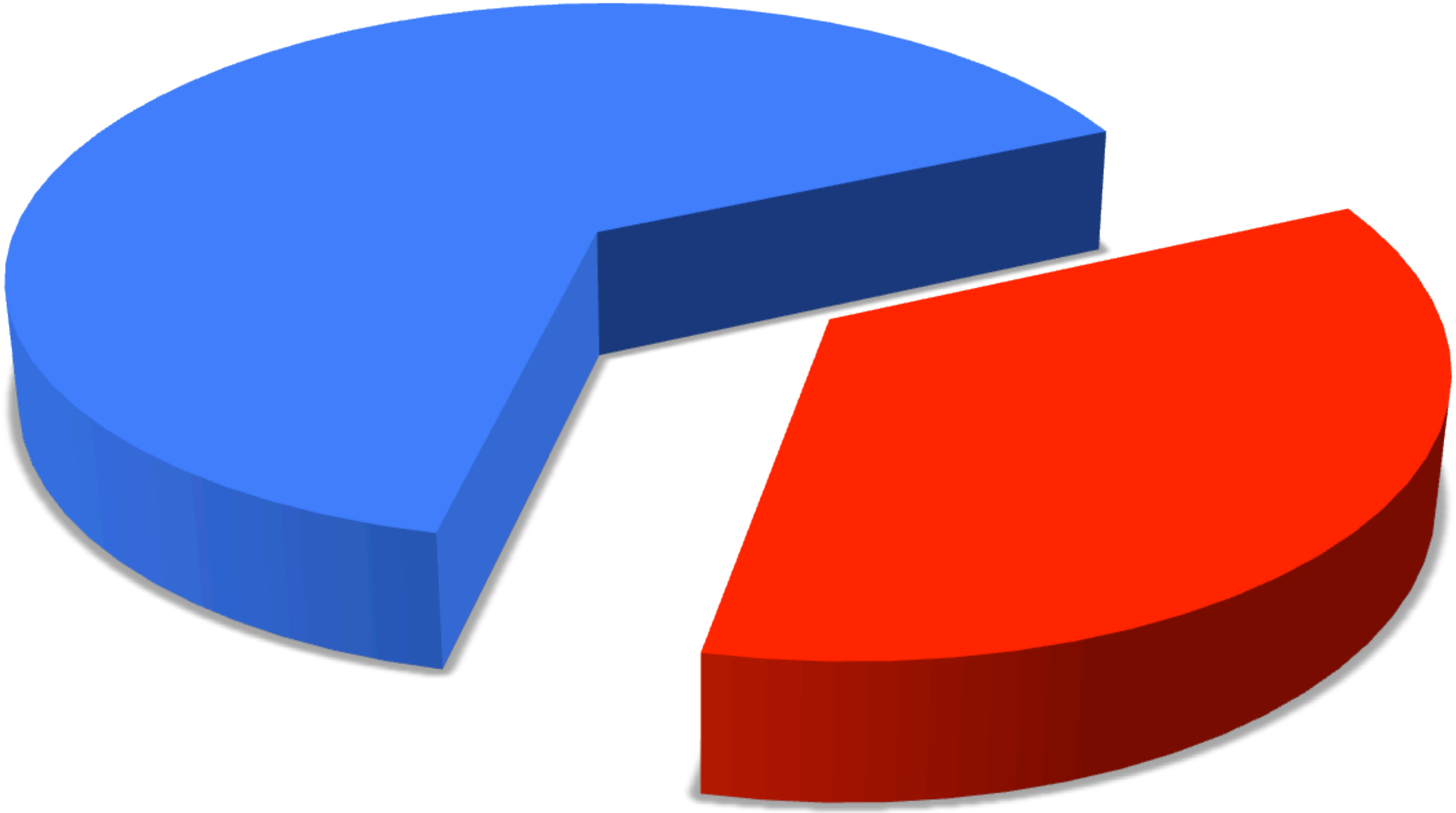


(source: Dave Snowden, IBM)

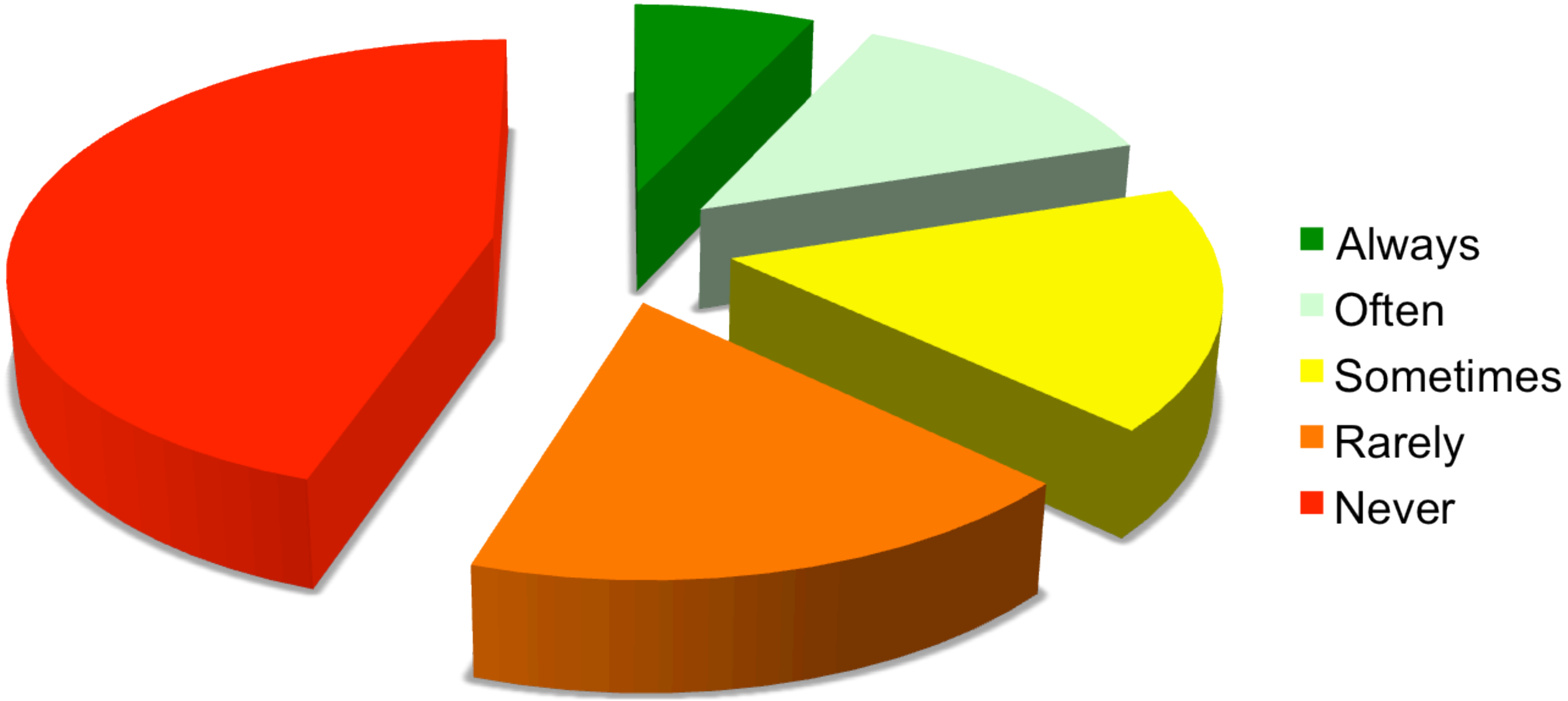


Demming Cycle

35% of Requirements Change

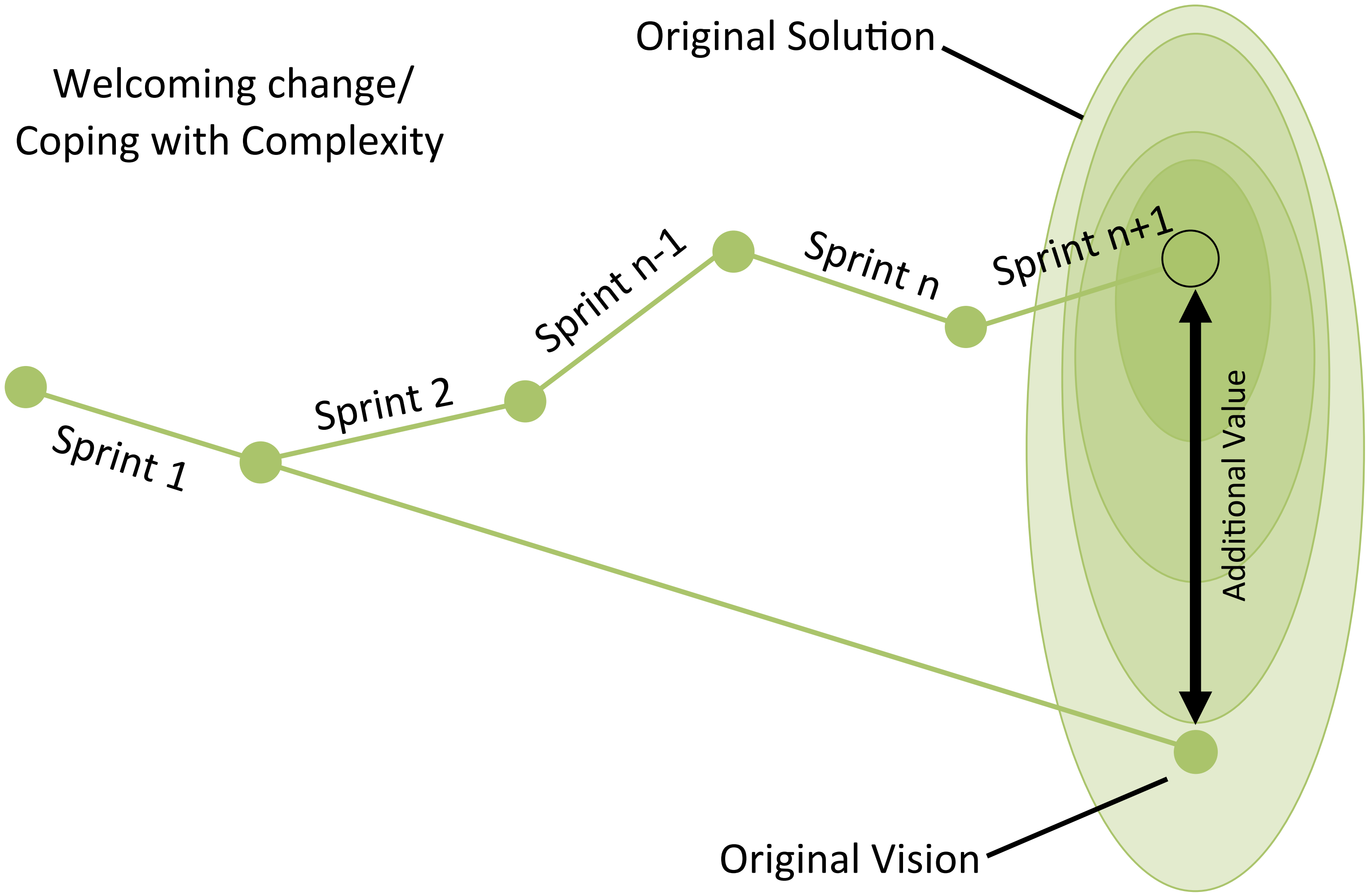


>60% of features are rarely or never used



(source: Standish Group)

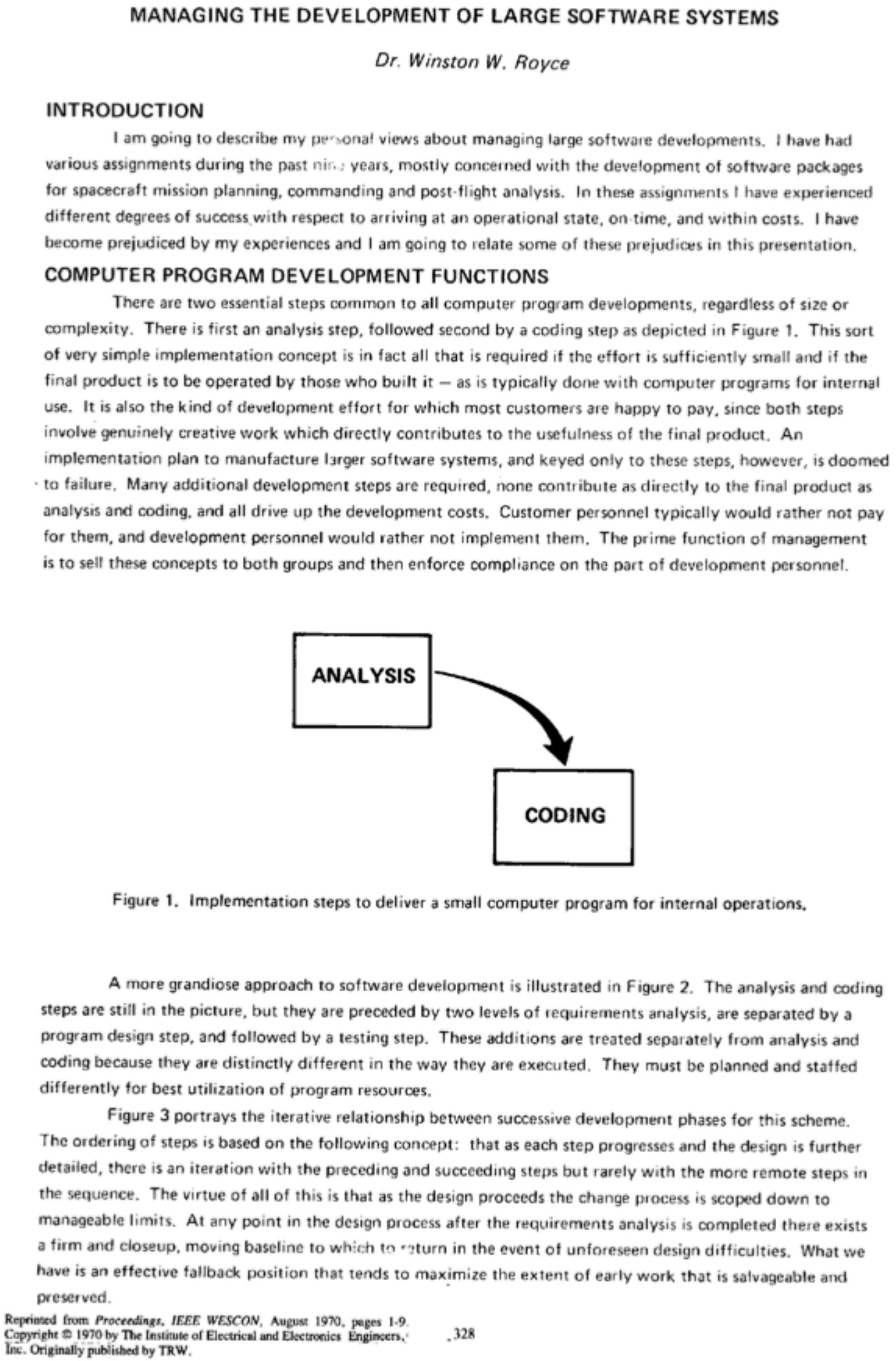
Value



Welcoming change/
Coping with Complexity

Original Solution

Original Vision



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 my experiences and I am going to relate some of these prejudices in this presentation.

COMPUTER PROGRAM DEVELOPMENT FUNCTIONS

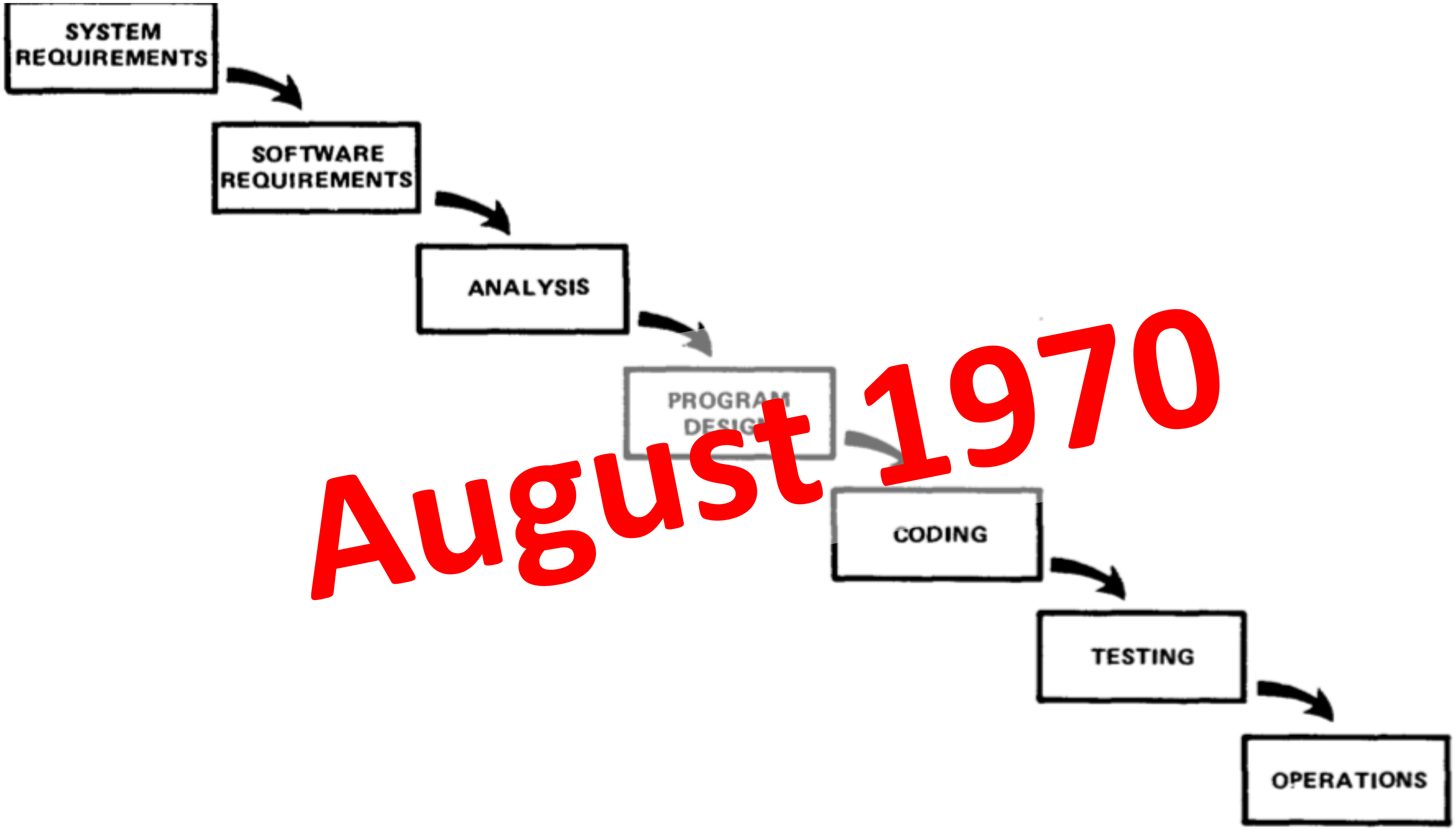
There are two essential steps common to all computer program developments, regardless of size or complexity. There is first an analysis step, followed second by a coding step as depicted in Figure 1. This sort of very simple implementation concept is in fact all that is required if the effort is sufficiently 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 as directly to the final product as analysis and coding, and all drive 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 sell these concepts to both groups and then enforce compliance on the part of development personnel.

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 additions 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 scoped 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 to return in the event of unforeseen design difficulties. What we have is an effective fallback position that tends to maximize the extent of early work that is salvageable and preserved.

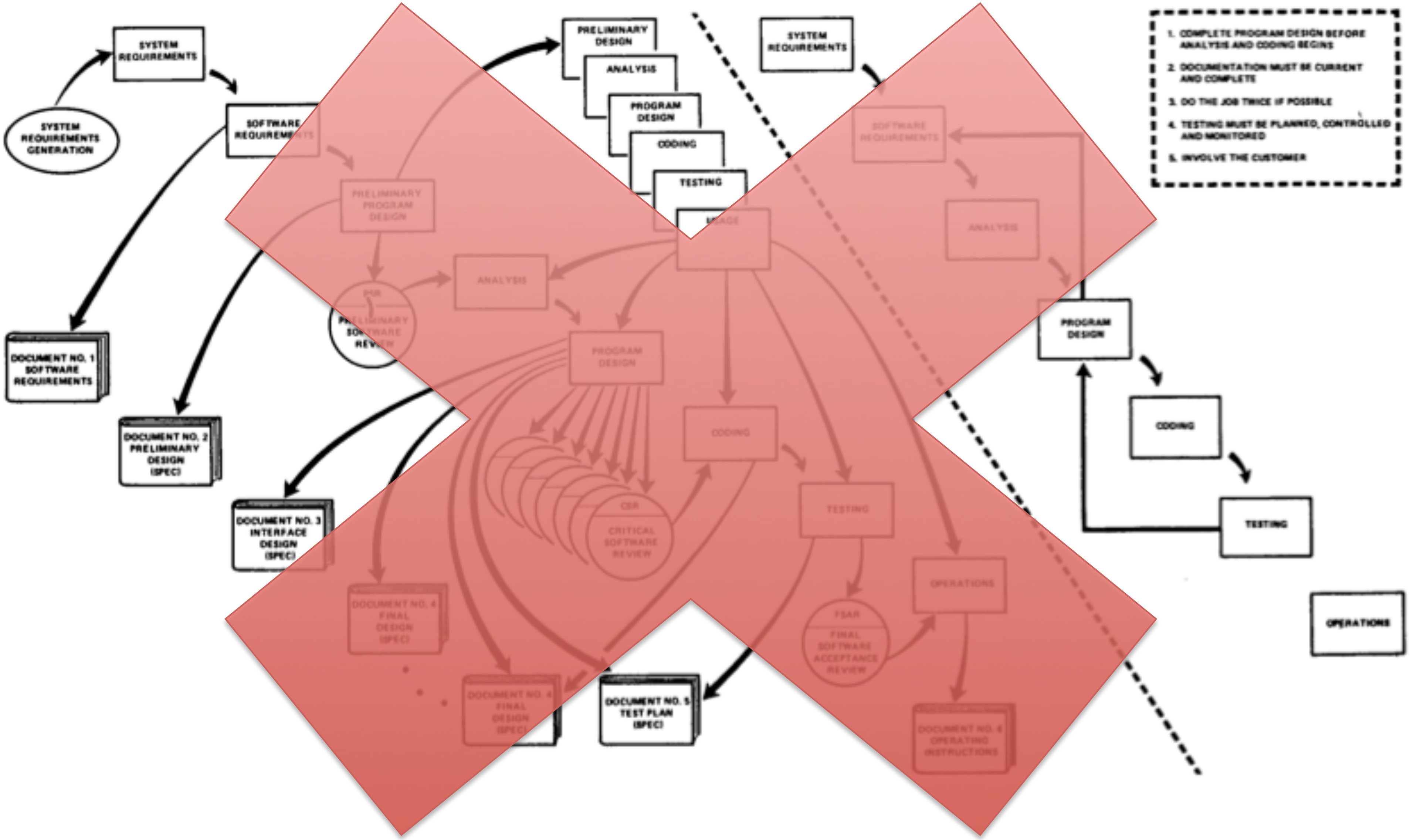
Reprinted from *Proceedings, IEEE WESCON,* August 1970, pages 1-9.
Copyright © 1970 by The Institute of Electrical and Electronics Engineers, Inc. Originally published by TRW.



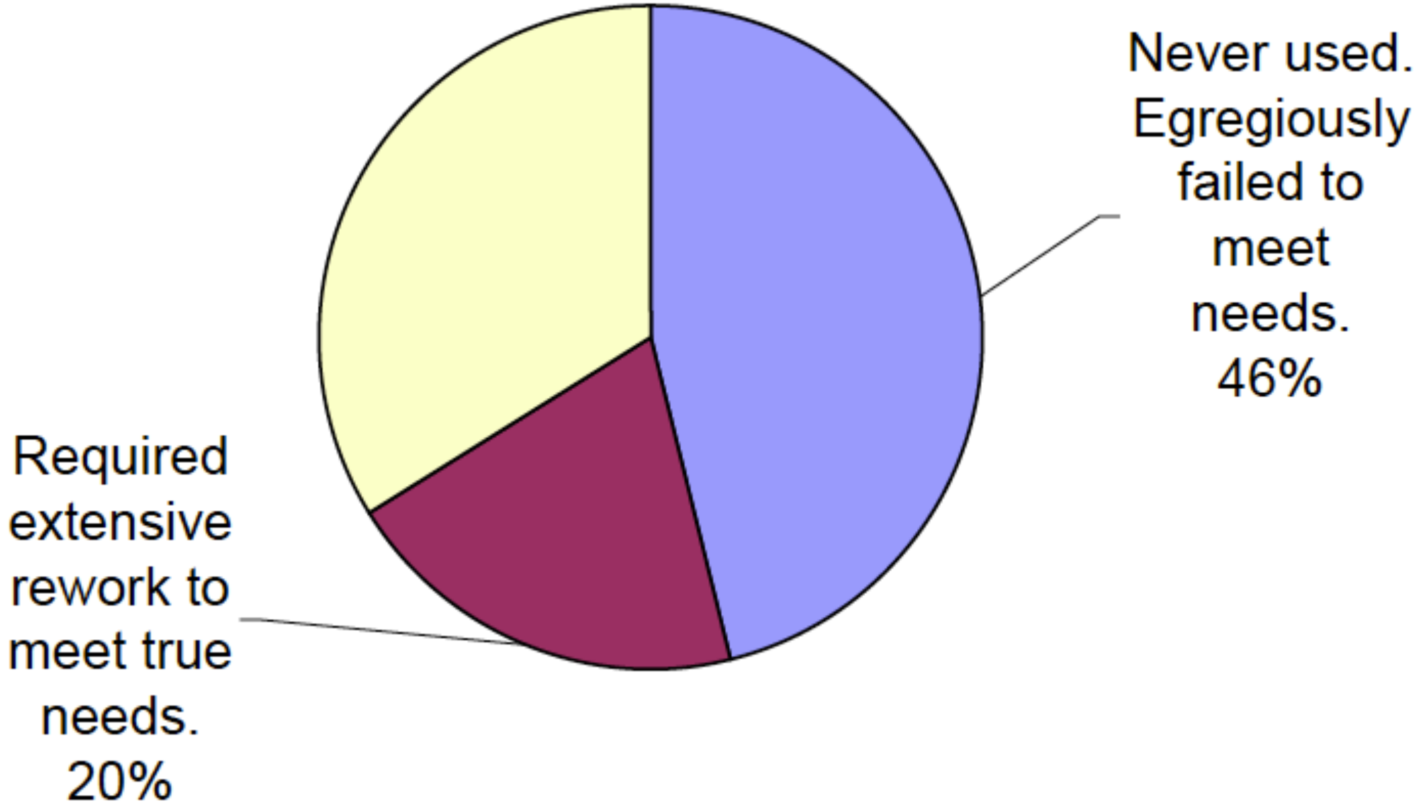
August 1970

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


SYSTEM REQUIREMENTS GENERATION
SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
DOCUMENT NO. 1 SOFTWARE REQUIREMENTS
DOCUMENT NO. 2 PRELIMINARY DESIGN (SPEC)
PRELIMINARY PROGRAM DESIGN
PSR PRELIMINARY SOFTWARE REVIEW
ANALYSIS
PROGRAM DESIGN
DOCUMENT NO. 3 INTERFACE DESIGN (SPEC)
CSR CRITICAL SOFTWARE REVIEW
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 5 TEST PLAN (SPEC)
PRELIMINARY DESIGN
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
USAGE
CODING
TESTING
FSAR FINAL SOFTWARE ACCEPTANCE REVIEW
OPERATIONS
DOCUMENT NO. 6 OPERATING INSTRUCTIONS
SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS
1. COMPLETE PROGRAM DESIGN BEFORE ANALYSIS AND CODING BEGINS
2. DOCUMENTATION MUST BE CURRENT AND COMPLETE
3. DO THE JOB TWICE IF POSSIBLE
4. TESTING MUST BE PLANNED, CONTROLLED AND MONITORED
5. INVOLVE THE CUSTOMER

$37B worth of DoD projects using 2167A

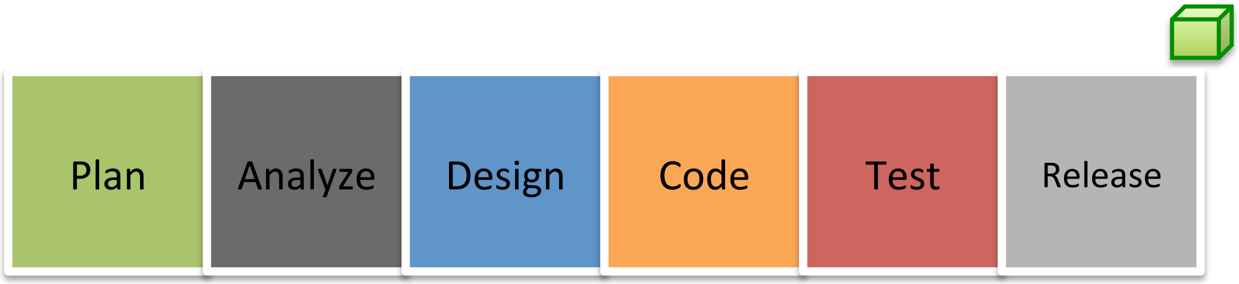


~~MIL-STD-498~~
~~167A~~

Defined vs Emperical

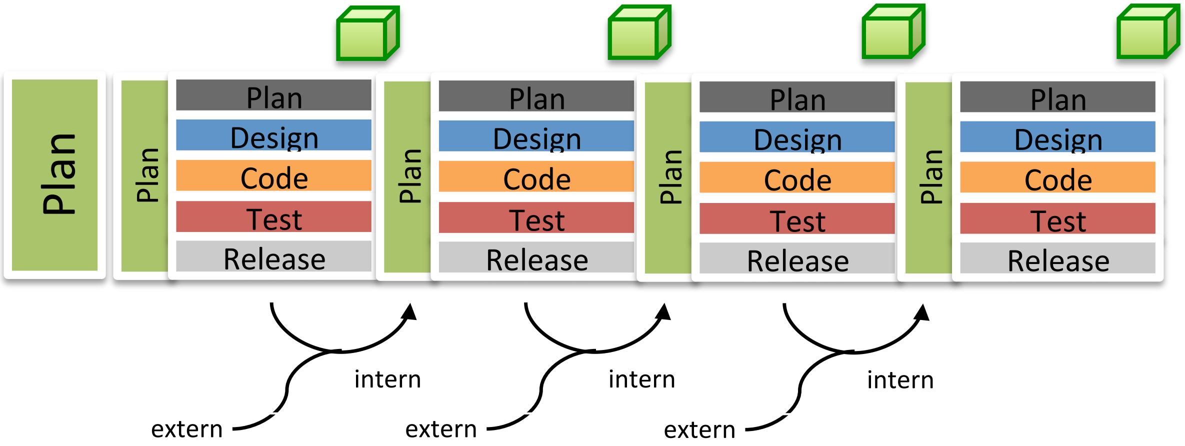
Waterfall (Defined)

Plan for the entire project up-front



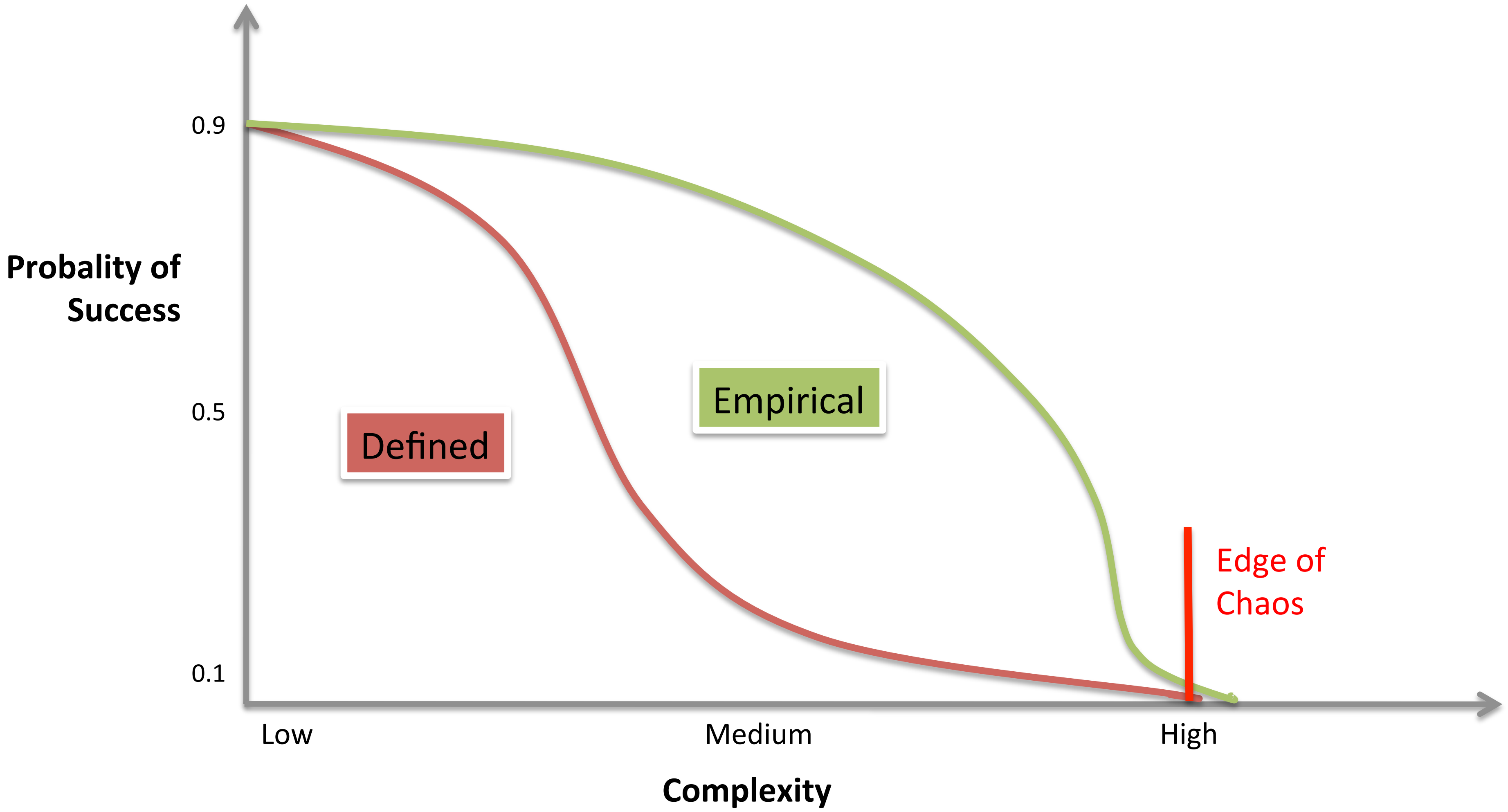
Scrum (Empirical)

Plan a little for the entire project and then a little for each Sprint



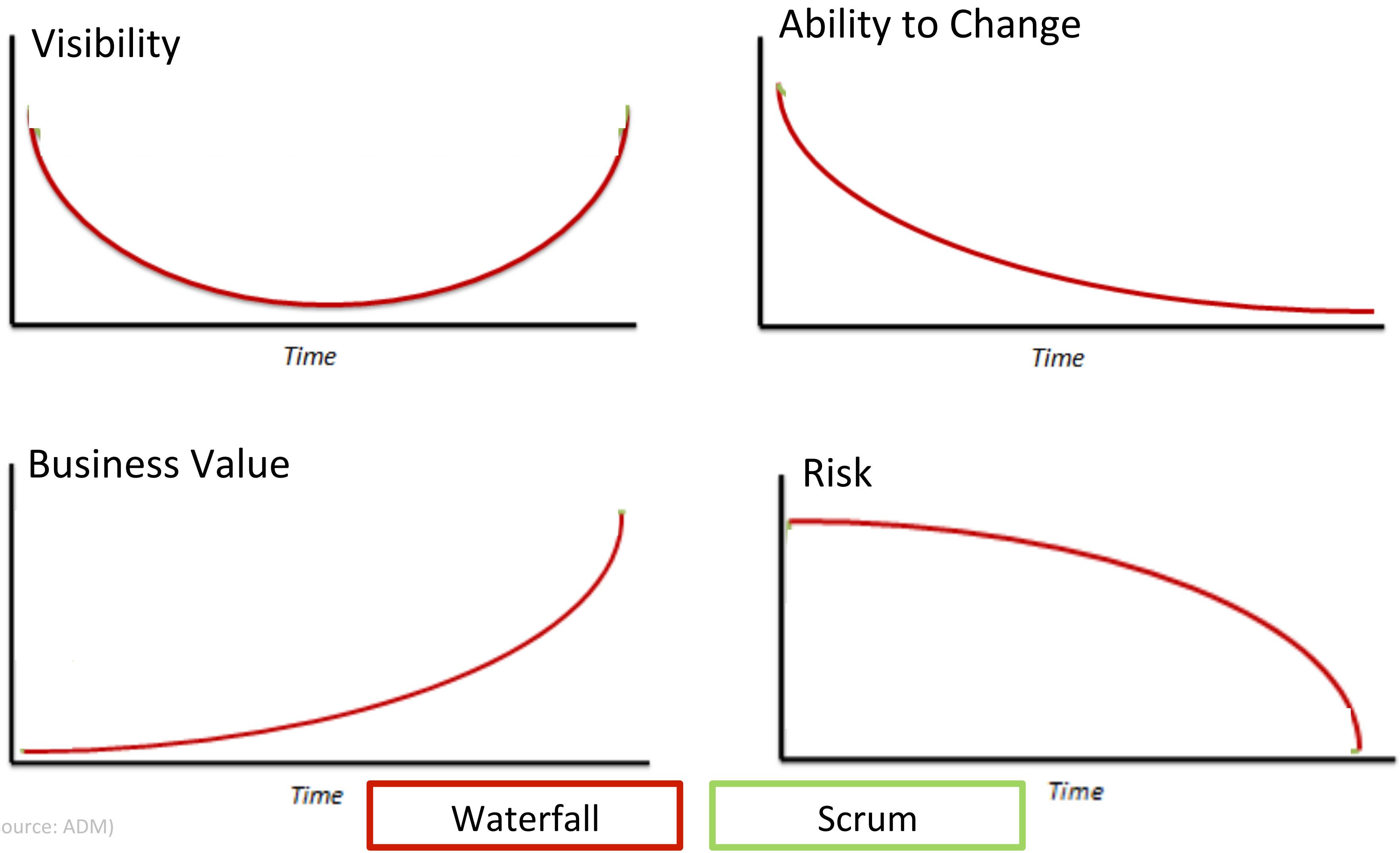
(source: ADM)

Higher Chance of Success



(source: ADM)

Why a higher Change?



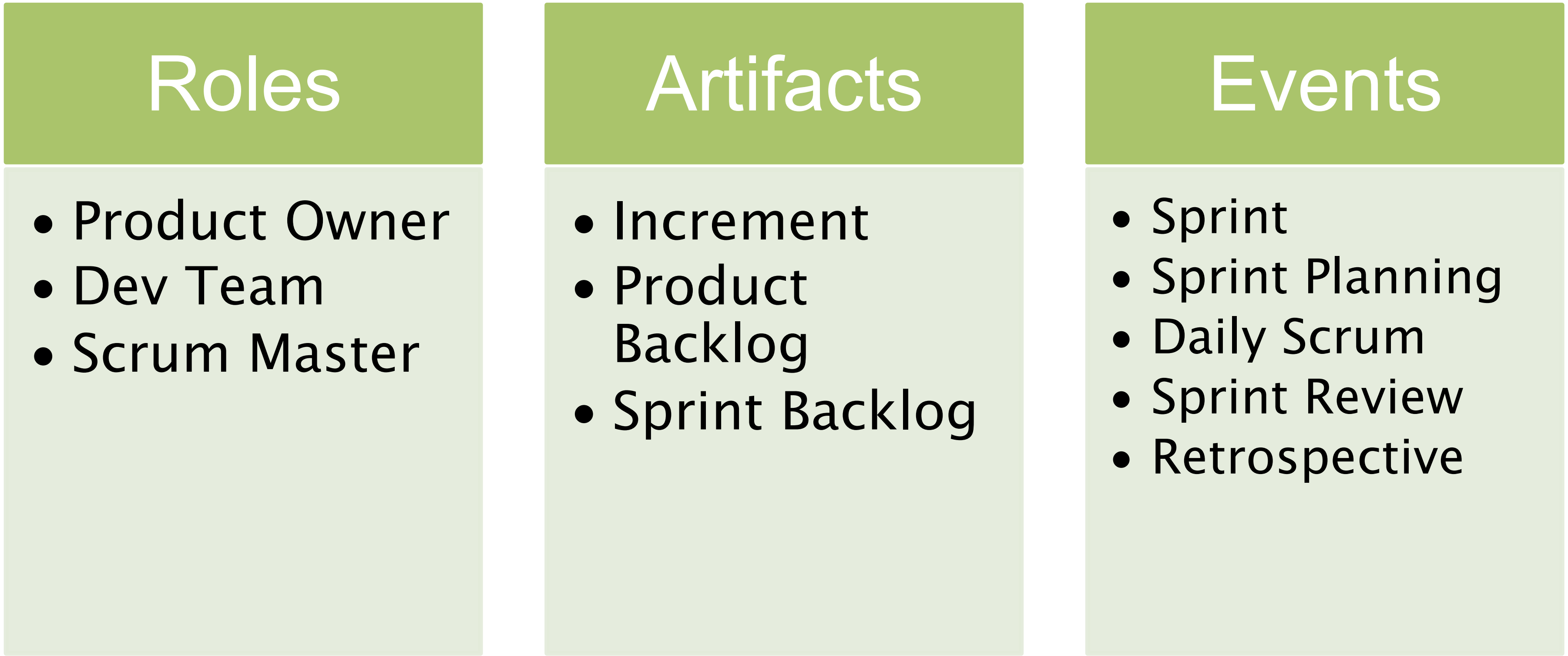
(source: ADM)

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



Roles

- Product Owner
- Dev Team
- Scrum Master

Artifacts

- Increment
- Product Backlog
- Sprint Backlog

Events

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Retrospective

(source: ADM)

Roles, Artifacts and Events in Action

Roles

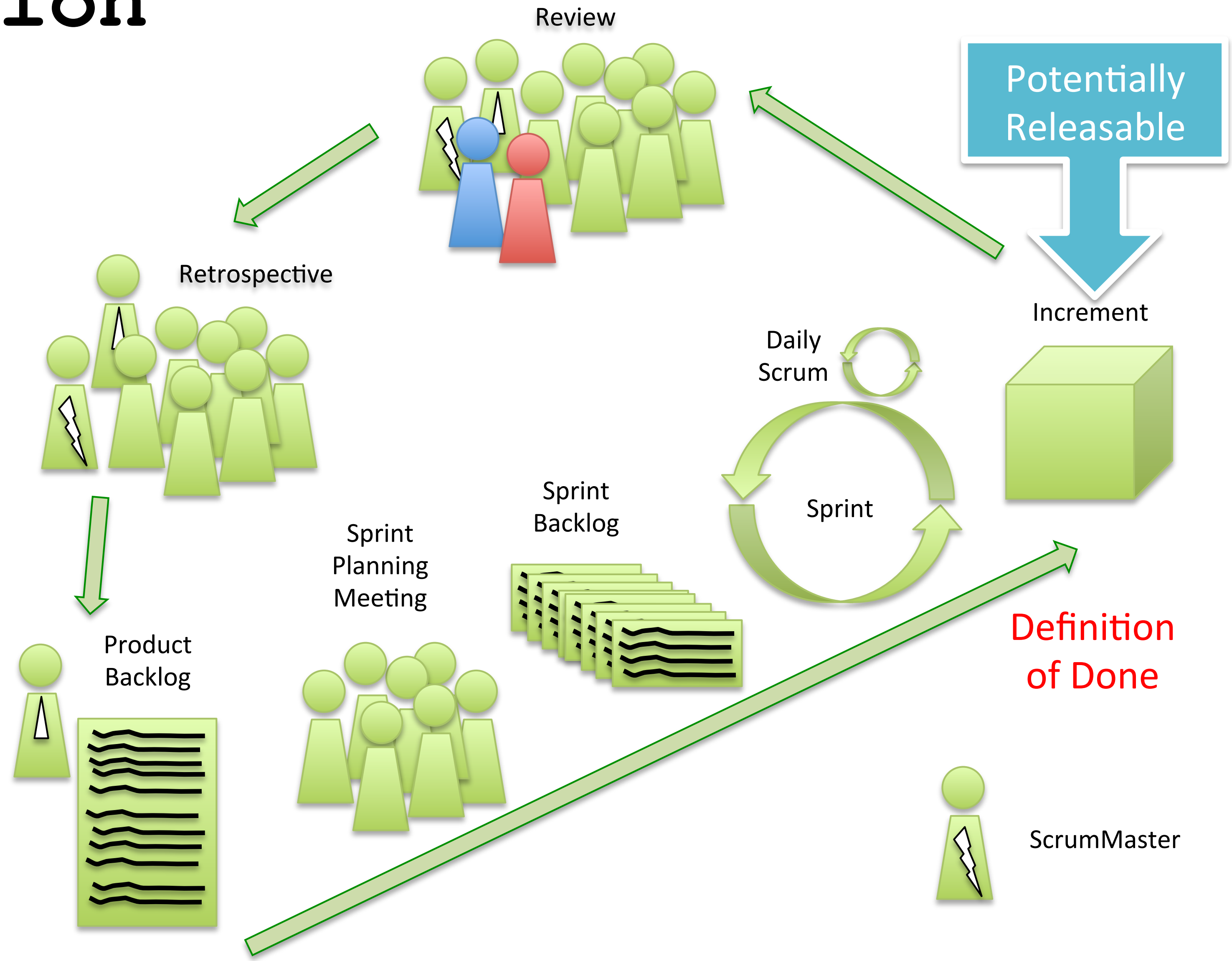
Product Owner
Development Team
Scrum Master

Artifacts

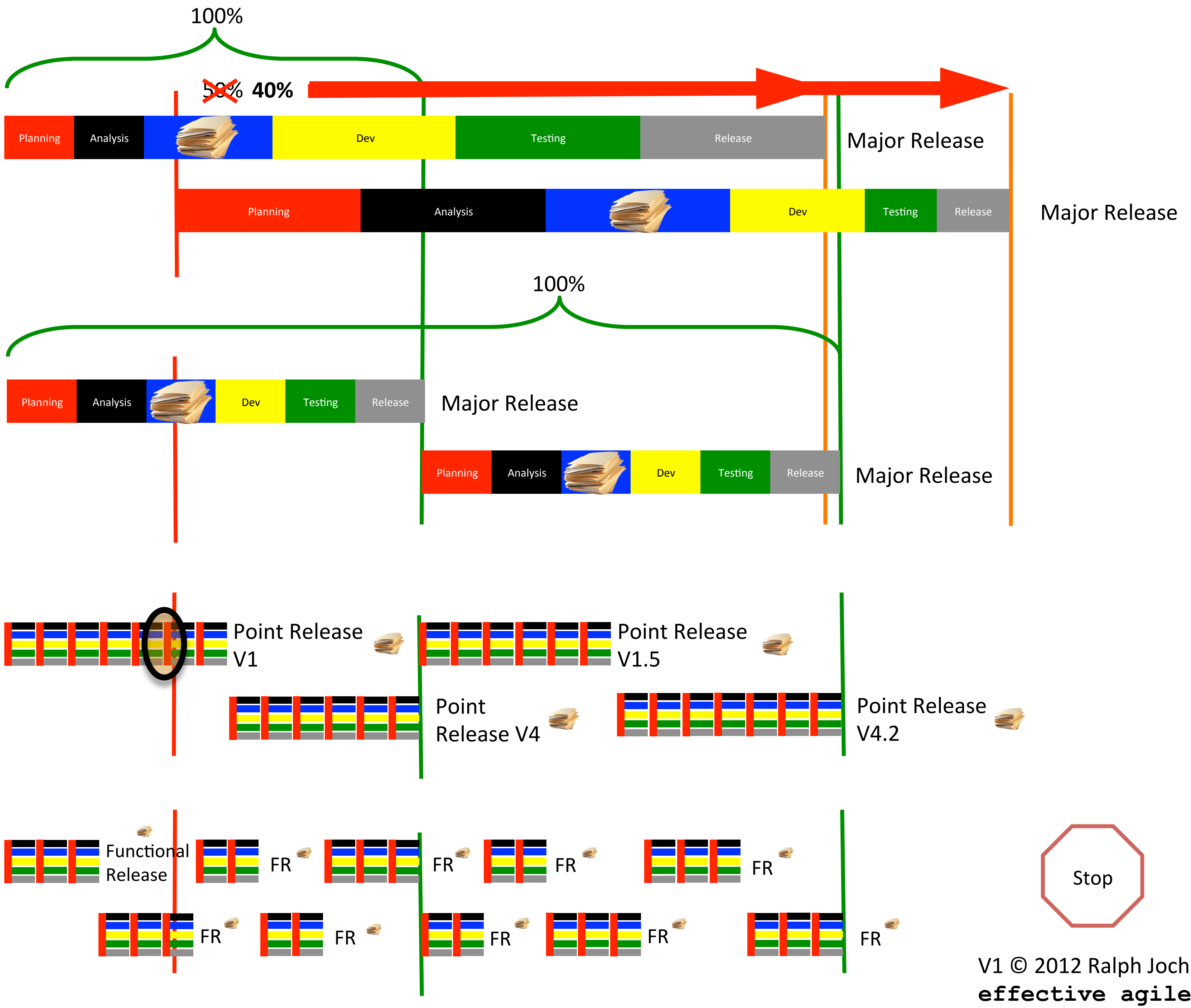
Product Backlog
Sprint Backlog
Increment

Events

Sprint Planning
Sprint
Daily Scrum
Sprint Review
Retrospective

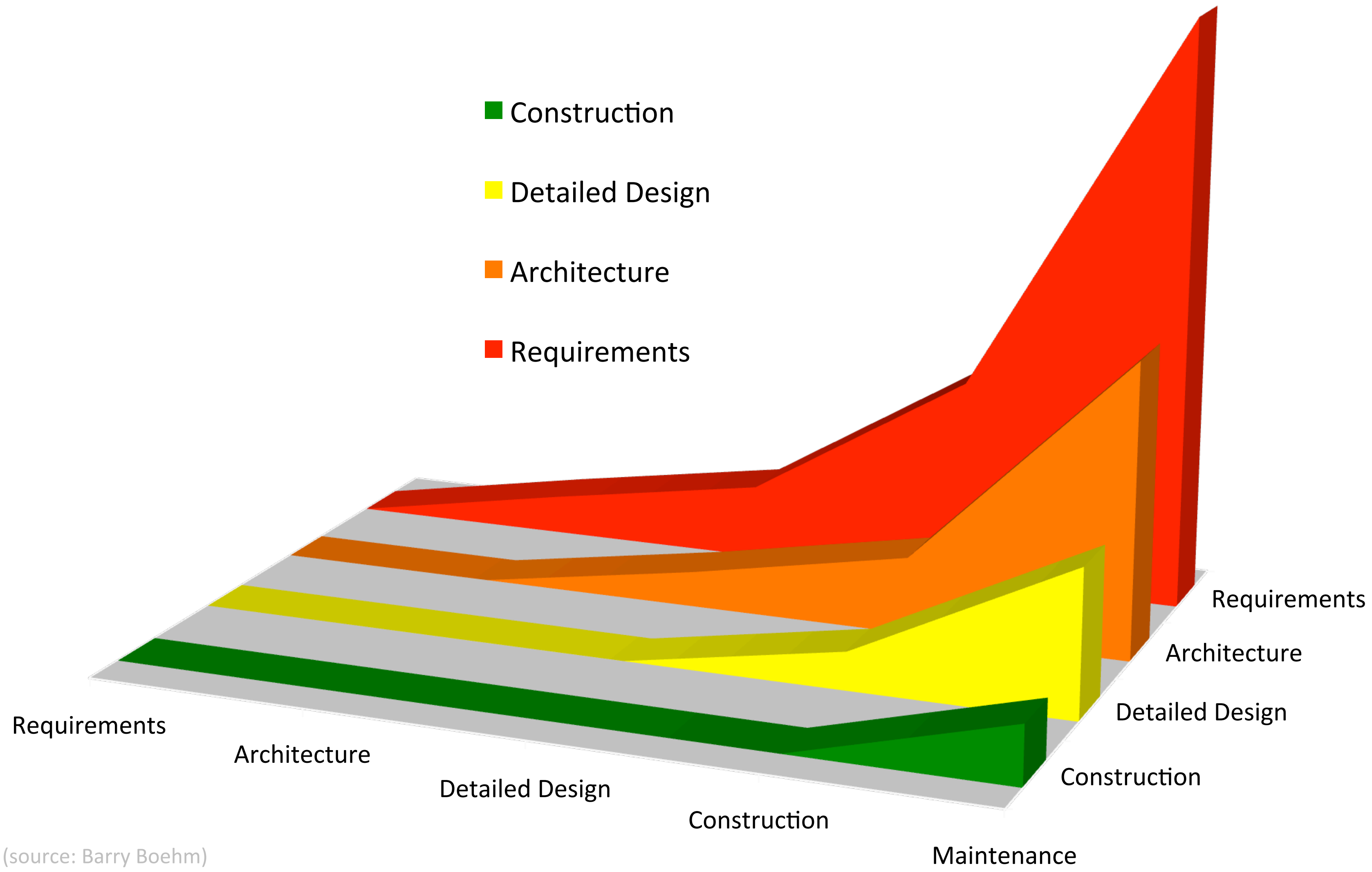


(source: ADM)



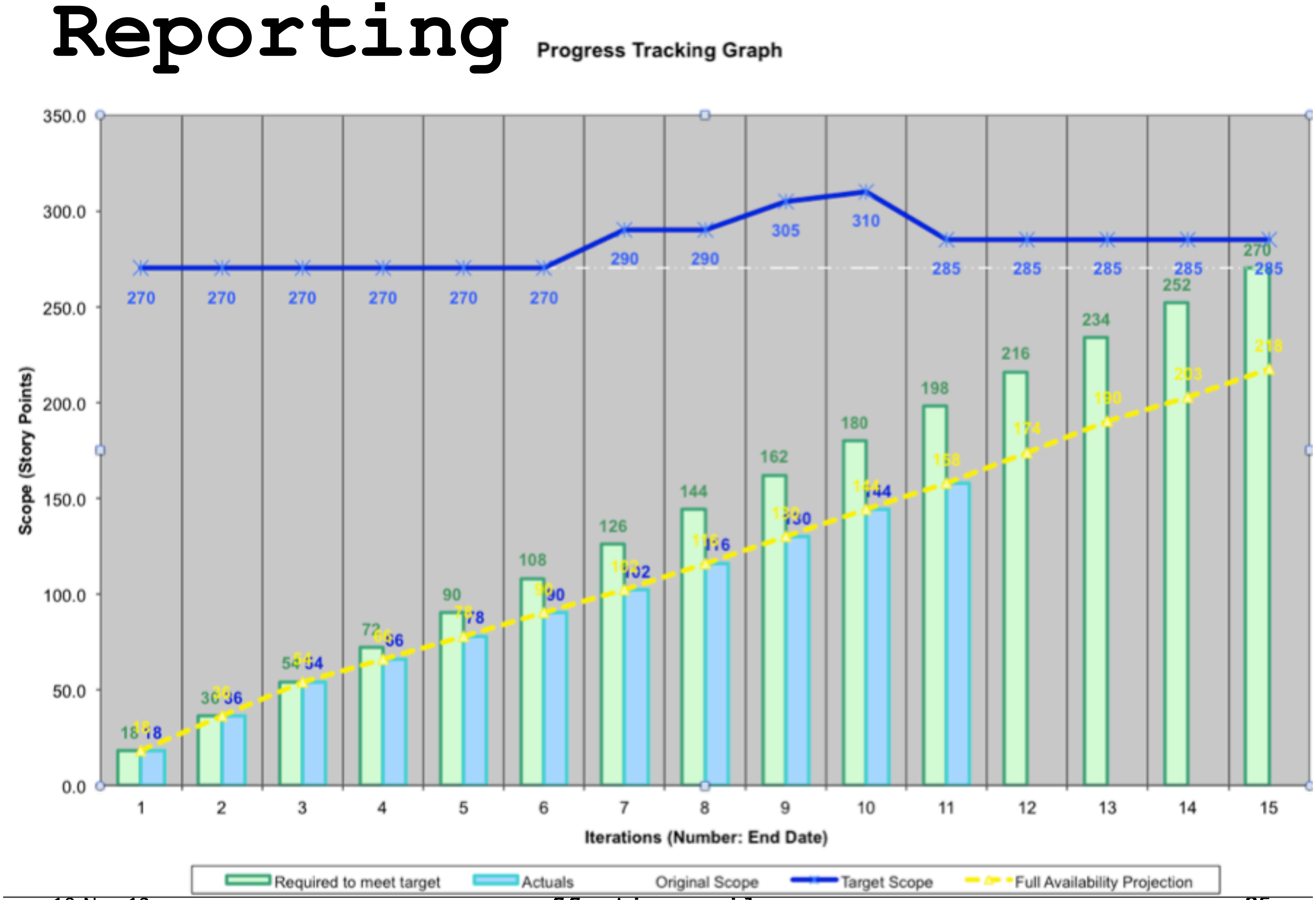
V1 © 2012 Ralph Jocham
effective agile.
www.effectiveagile.com

Cost of fixing a Bug



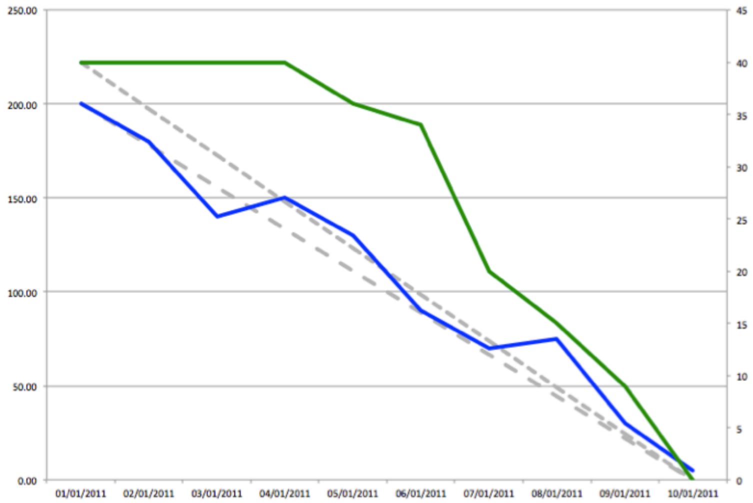
(source: Barry Boehm)

Reporting

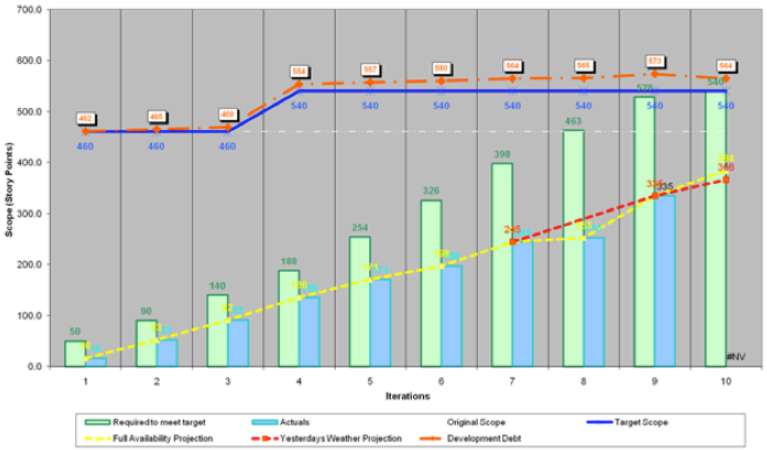

Progress Tracking Graph
Scope (Story Points)
350.0
300.0
250.0
200.0
150.0
100.0
50.0
0.0
270
270
270
270
270
270
290
290
305
310
285
285
285
285
285
18
36
54
72
90
108
126
144
162
180
198
216
234
252
270
18
36
54
66
78
90
102
116
130
144
158
18
36
54
66
78
90
102
116
130
144
158
174
190
203
218
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Iterations (Number: End Date)
Required to meet target
Actuals
Original Scope
Target Scope
Full Availability Projection

Reporting

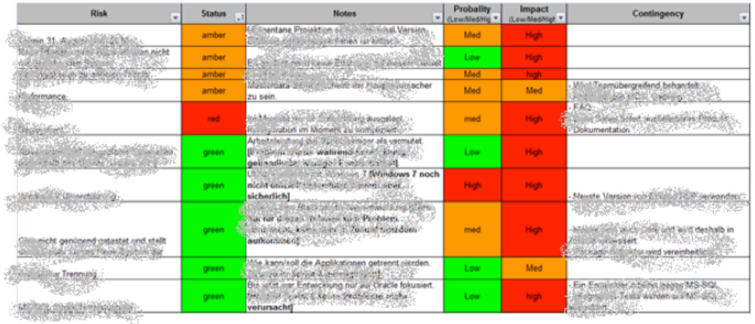
Sprint Burndown



Release Burnup

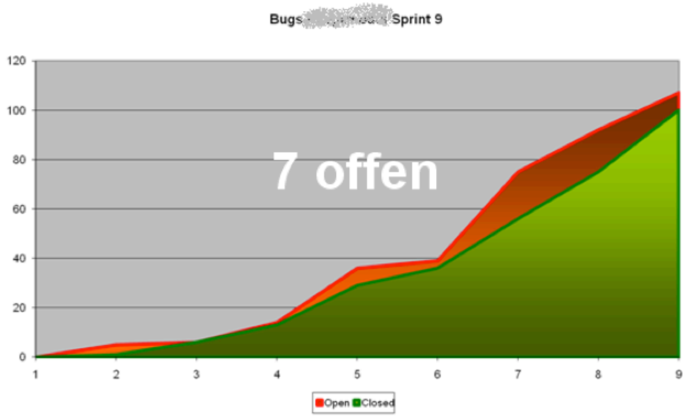


Risiken



Risk	Status	Notes	Probality (Low/Med/High)	Impact (Low/Med/High)	Contingency
[redacted]	amber	[redacted]	Med	High	
[redacted]	amber	[redacted]	Low	High	
[redacted]	amber	[redacted]	Med	high	
[redacted]	amber	[redacted]	Med	Med	[redacted]
[redacted]	red	[redacted]	med	High	[redacted]
[redacted]	green	[redacted]	Low	High	
[redacted]	green	[redacted]	High	High	[redacted]
[redacted]	green	[redacted]	med	High	[redacted]
[redacted]	green	[redacted]	Low	Med	
[redacted]	green	[redacted]	Low	high	[redacted]

Bugs



Questions?

READ

THE SCRUM GUIDE

http://www.scrum.org/Scrum-Guides

Ralph Jocham

effective agile.

www.effectiveagile.com

ralph@effectiveagile.com

@rjocham