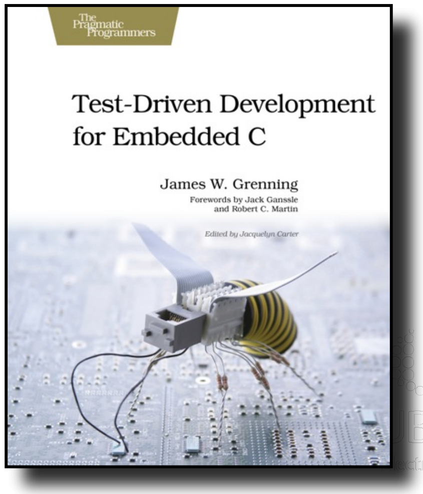
designwest
center of the engineering universe

Agile Embedded Software Development

James Grenning
@jwgrenning
james@wingman-sw.com


The Pragmatic Programmers
Test-Driven Development for Embedded C
James W. Grenning
Forewords by Jack Ganssle and Robert C. Martin
Edited by Jacquelyn Carter

Software Development is Easy!

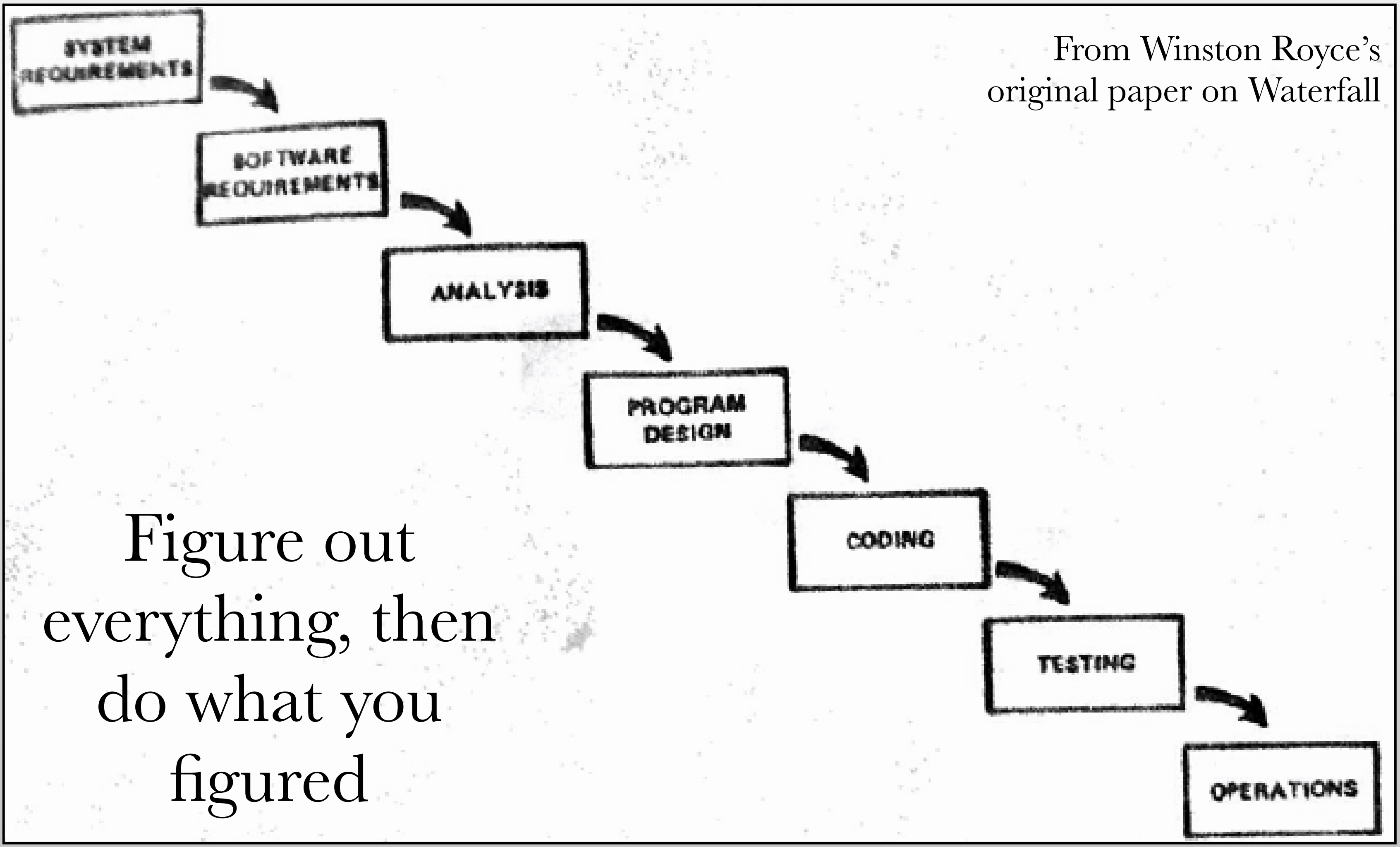
- Just like this *Black Diamond*

We never have any problems like

- Late Delivery
- Poor Quality
- Burnout
- Missed Customer Expectations

Software Development is Easy!

Can Projects be Managed Better?



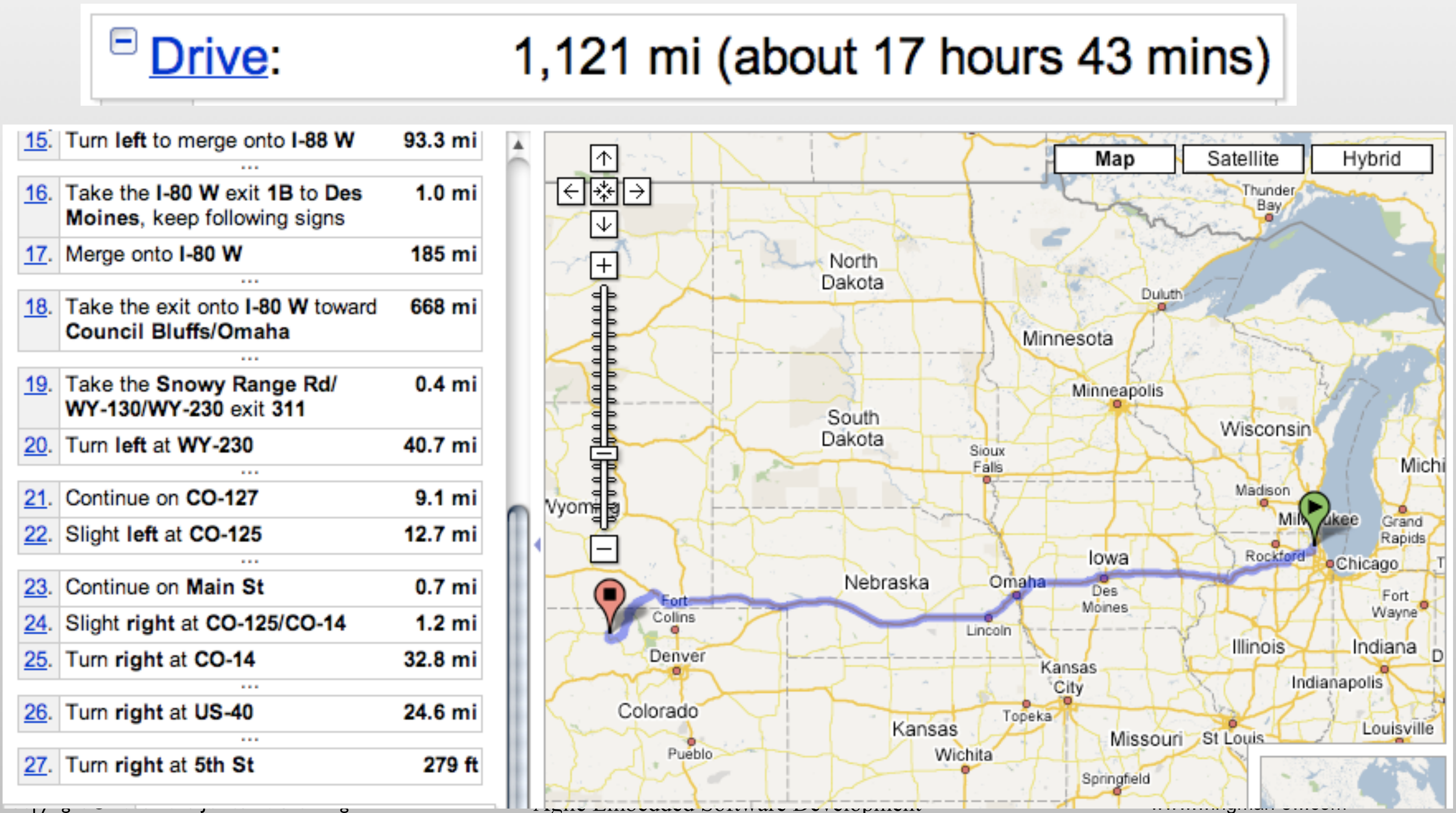
From Winston Royce's original paper on Waterfall

Figure out everything, then do what you figured

Thank you: Hubert Stoffels, from Pittsburgh, USA '''Title:''' Jonathan's Run Falls

'''Description:''' http://commons.wikimedia.org/wiki/File:Jonathan%27s_Run_Falls.jpg

Figure it All Out, Then Do It



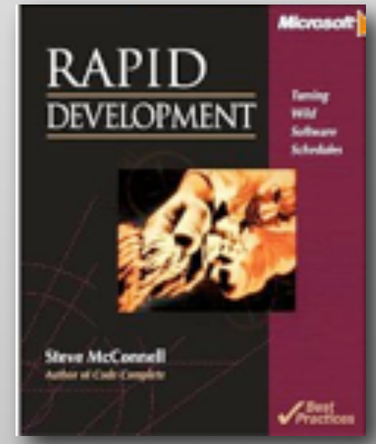
Drive: 1,121 mi (about 17 hours 43 mins)

15.	Turn **left** to merge onto **I-88 W**	93.3 mi
16.	Take the **I-80 W** exit **1B** to **Des Moines**, keep following signs	1.0 mi
17.	Merge onto **I-80 W**	185 mi
18.	Take the exit onto **I-80 W** toward **Council Bluffs/Omaha**	668 mi
19.	Take the **Snowy Range Rd/ WY-130/WY-230** exit **311**	0.4 mi
20.	Turn **left** at **WY-230**	40.7 mi
21.	Continue on **CO-127**	9.1 mi
22.	Slight **left** at **CO-125**	12.7 mi
23.	Continue on **Main St**	0.7 mi
24.	Slight **right** at **CO-125/CO-14**	1.2 mi
25.	Turn **right** at **CO-14**	32.8 mi
26.	Turn **right** at **US-40**	24.6 mi
27.	Turn **right** at **5th St**	279 ft

Thanks to Yorian, Picture of a waterfall nearby Flam, Norway http://

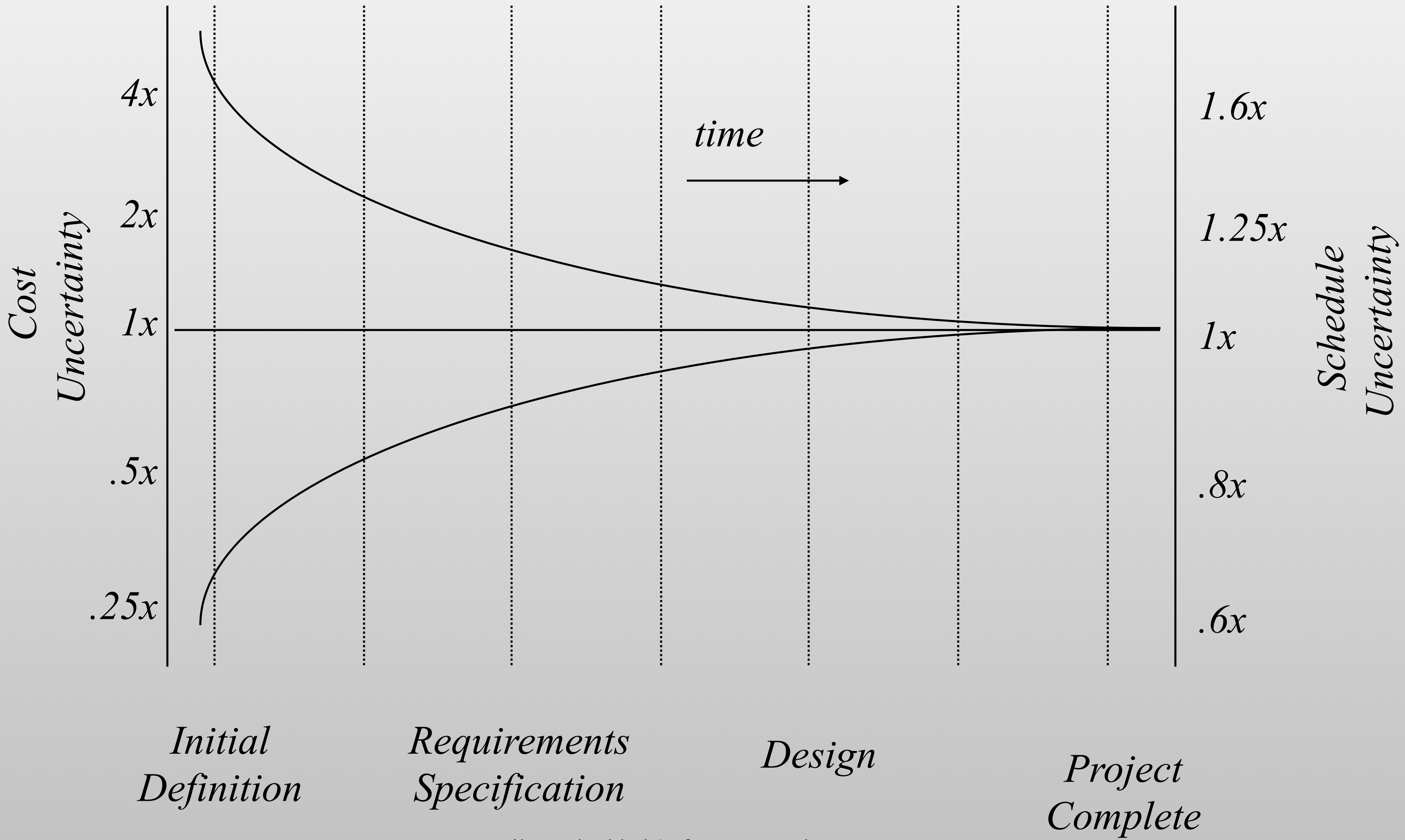
Steve McConnell
from Rapid Development

"Software projects contain too many variables to be able to set schedules with 100-percent accuracy. Far from having one particular date when a project would finish, for any given project there is a range of completion dates, of which some are more likely and some are less."



Project Cost and Schedule Uncertainty

Barry Boehm, 1995



www.wingman-sw.com
james@wingman-sw.com

What is Agile?

Can we get features and functionality to flow?

What is Agile?

- *Agile software development is a conceptual framework for undertaking software engineering projects.*

  -- wikipedia

- a.k.a. Extreme Programming, Scrum, Feature Driven Development, DSDM, Crystal Clear, Agile Unified Process

Agile methods are Designed to...

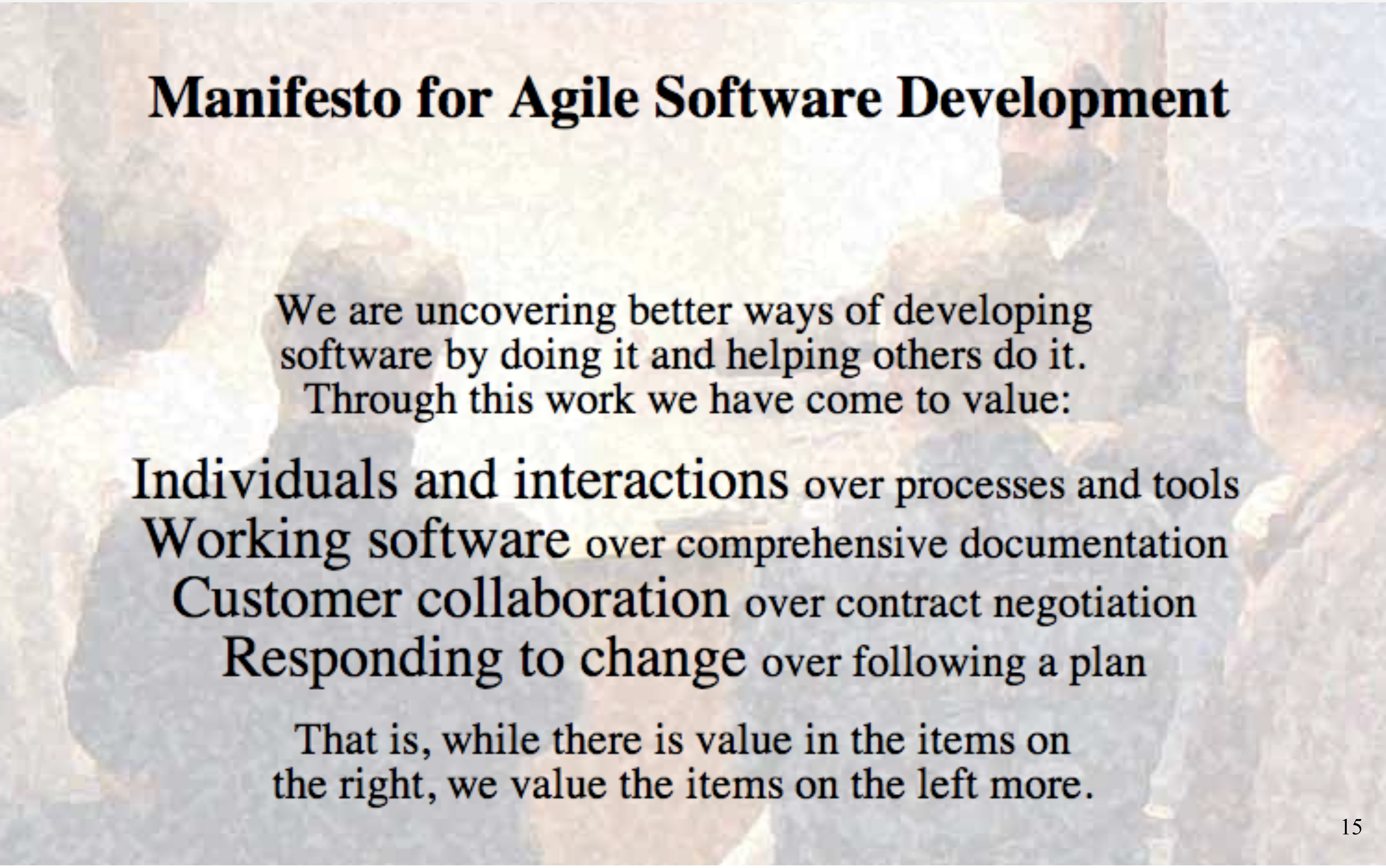
- Manage with Data
- Improve Visibility
- Improve Predictability
- Improve Quality
- Improve Productivity
- Reduce Waste

www.wingman-sw.com
james@wingman-sw.com

Agile Principles

- Communications
- Simplicity
- Feedback
- Courage
- Respect
- Visibility
- Honesty
- Realistic
- High Quality

www.agilemanifesto.org



Manifesto for Agile Software Development

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.

Individuals and Interactions
over Processes and Tools

Skilled Self-Organizing Teams

- Developers work together to organize the work
- Customer or Product Owner works with the teams to define work and establish priorities
- Managers usually take an outward focus, removing roadblock, rather than managing day-to-day tasks and schedules.

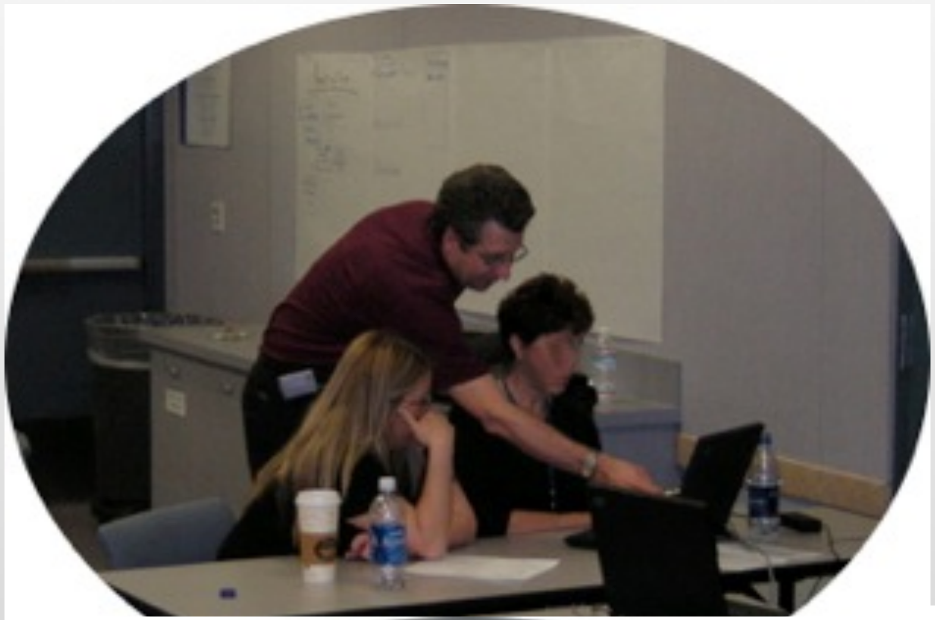
Collaboration

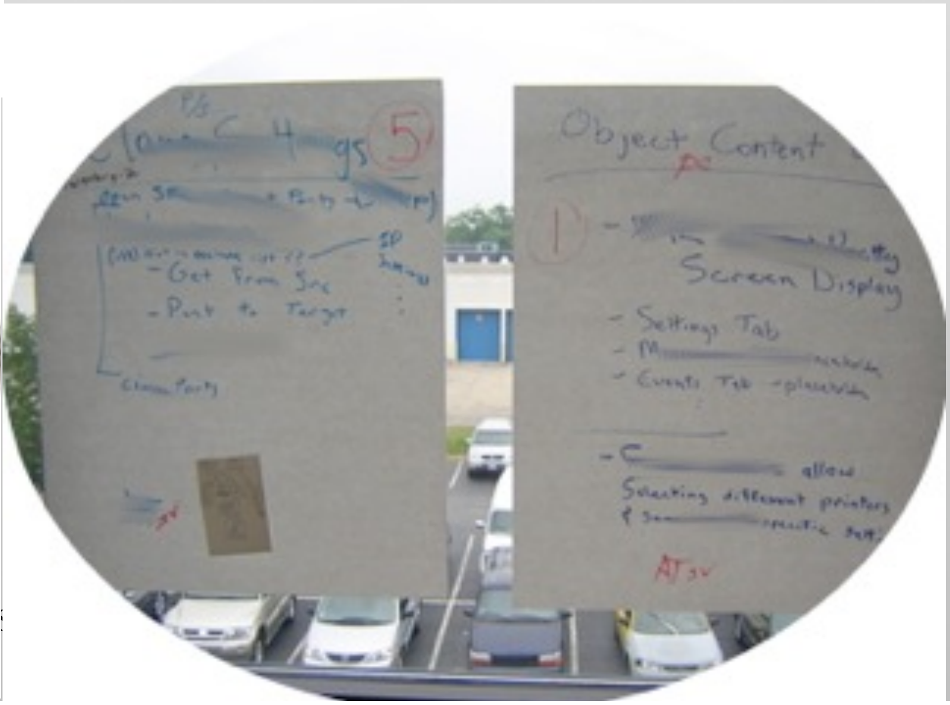
- Daily standup meeting
- Pair programming or Daily reviews
- Shared code ownership
- Team room

www.wingman-sw.com
james@wingman-sw.com

Working Software over Comprehensive Documentation

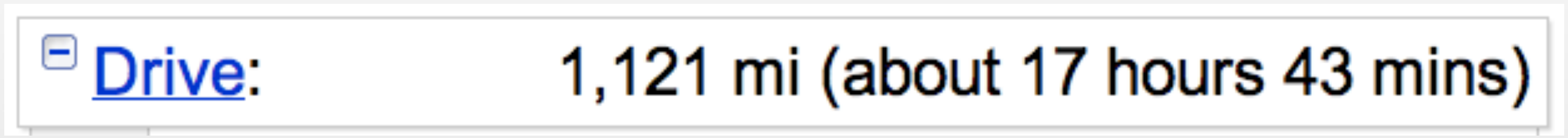
- Each team has different needs
- Less formal documentation might work.
- Prefer executable Documentation



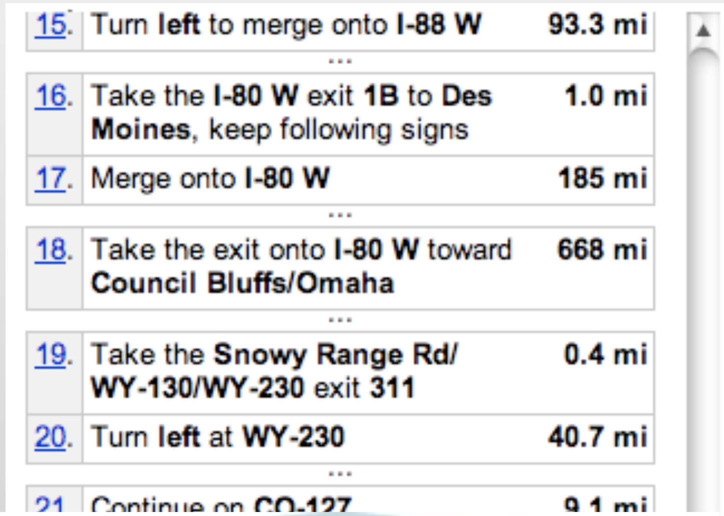


Customer Collaboration
over Contract Negotiation

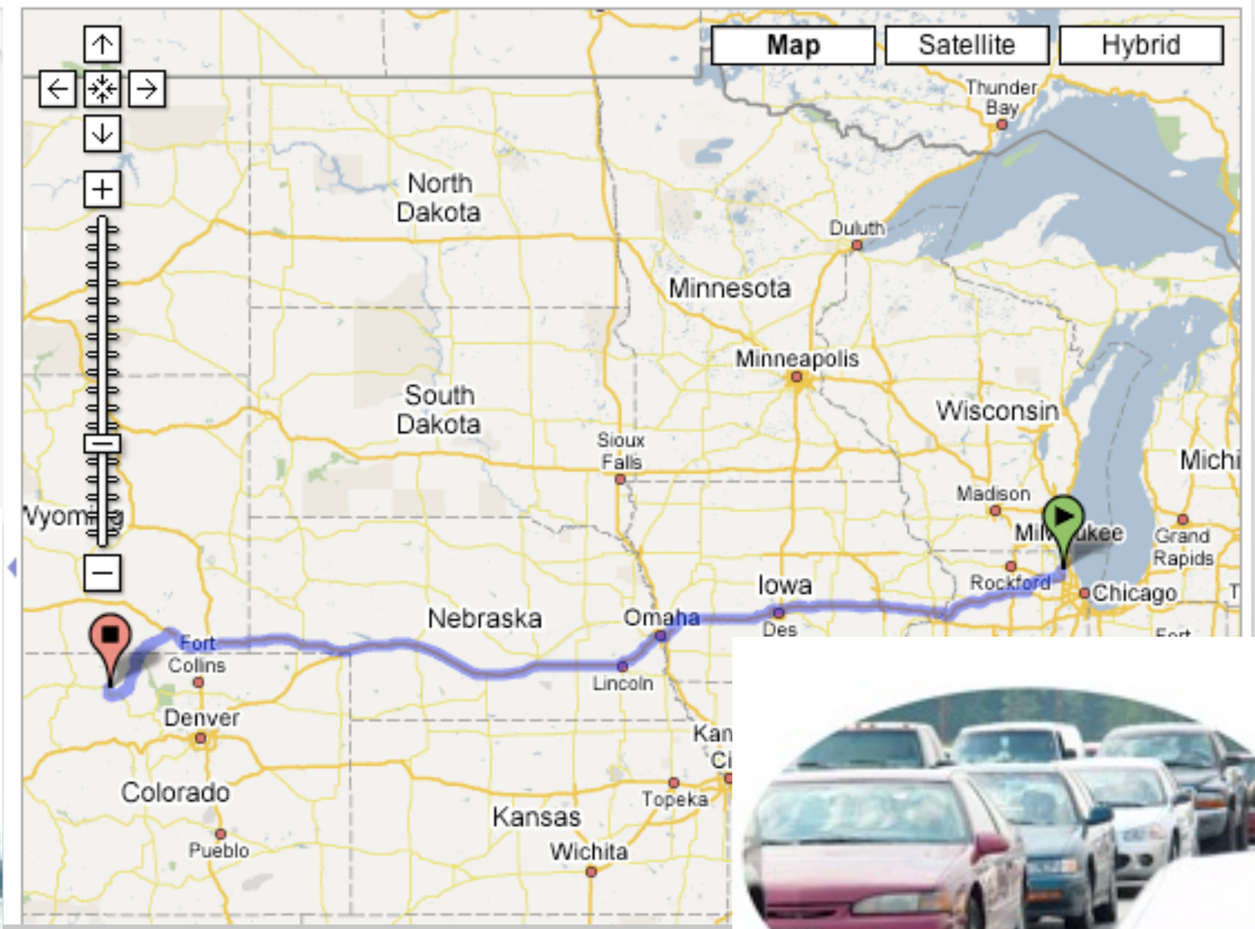
Responding to Change
over following a plan

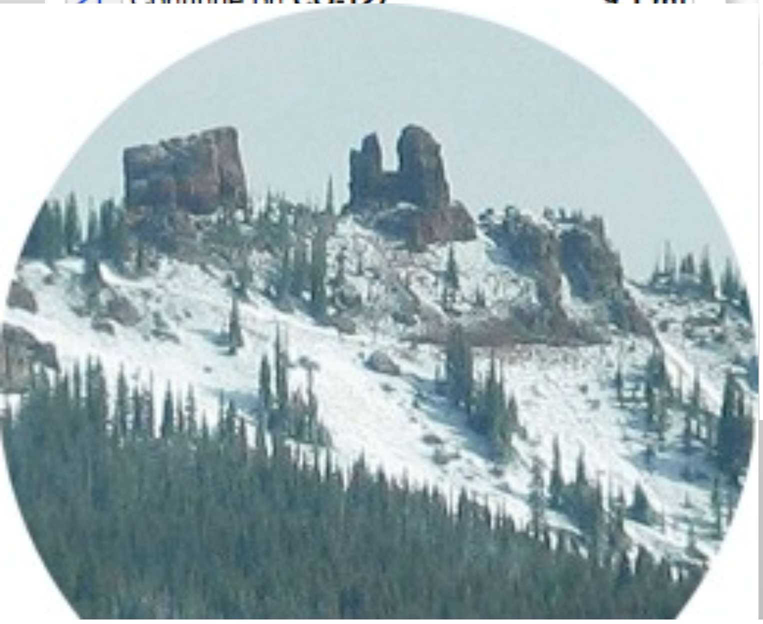


Drive: 1,121 mi (about 17 hours 43 mins)



15.	Turn **left** to merge onto **I-88 W**	93.3 mi
16.	Take the **I-80 W** exit **1B** to **Des Moines**, keep following signs	1.0 mi
17.	Merge onto **I-80 W**	185 mi
18.	Take the exit onto **I-80 W** toward **Council Bluffs/Omaha**	668 mi
19.	Take the **Snowy Range Rd/ WY-130/WY-230** exit **311**	0.4 mi
20.	Turn **left** at **WY-230**	40.7 mi
21.	Continue on **CO-127**	9.1 mi





Agile Approach is more...

- Visible
- Predictable
- Productive

- With a focus on
  - High Quality Work
  - Reduced Waste

Challenges for Embedded

- Stories and incremental scope control
- Breaking dependencies on hardware
- Applying outside of software
  - Mechanics, hardware, ASIC development

- Not unique to embedded, though prevalent
  - Your own preconceived notions
  - Organizational resistance

Iterative and Incremental Development

Projects end, products don't (hopefully)

Requirements analysis is never done

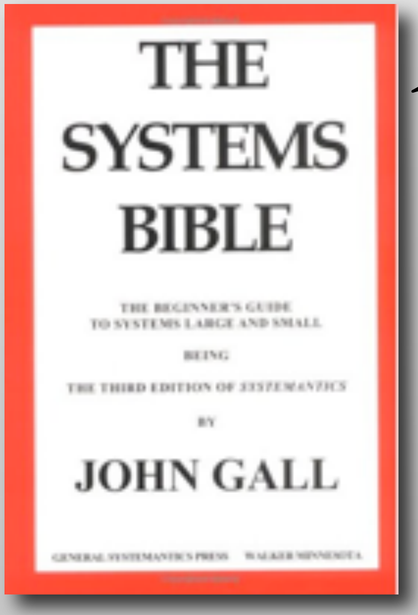
Design is never done

Why Iterative?

- A system's users seldom know exactly what they what and cannot articulate all they know
- … There are many details we can only discover once we are well into implementation
- … as humans we can only master only so much complexity
- … external forces lead to changes in requirements…

[LARMEN]

"A complex system designed from scratch never works, and cannot be made to work. You have to start over, beginning with a simple system."


THE SYSTEMS BIBLE

THE BEGINNER'S GUIDE TO SYSTEMS LARGE AND SMALL

BEING

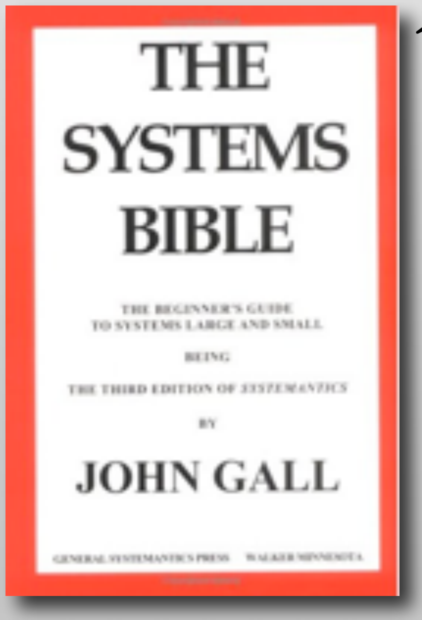
THE THIRD EDITION OF SYSTEMANTICS

BY

JOHN GALL

GENERAL SYSTEMANTICS PRESS WALKER MINNESOTA

A Complex system that works is invariably found to have evolved from a simple system that worked



THE SYSTEMS BIBLE

THE BEGINNER'S GUIDE TO SYSTEMS LARGE AND SMALL

BEING

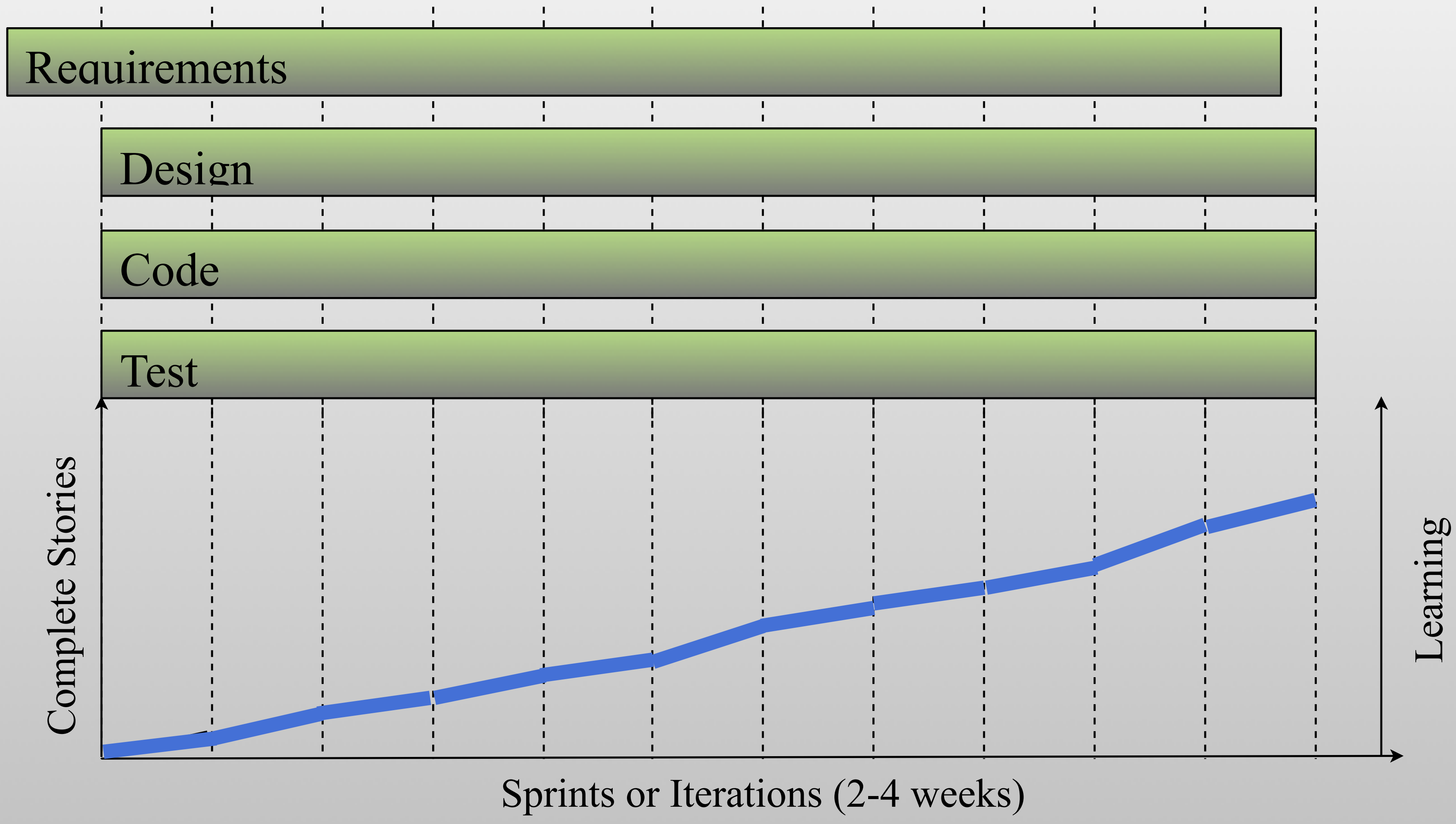
THE THIRD EDITION OF *SYSTEMANTICS*

BY

JOHN GALL

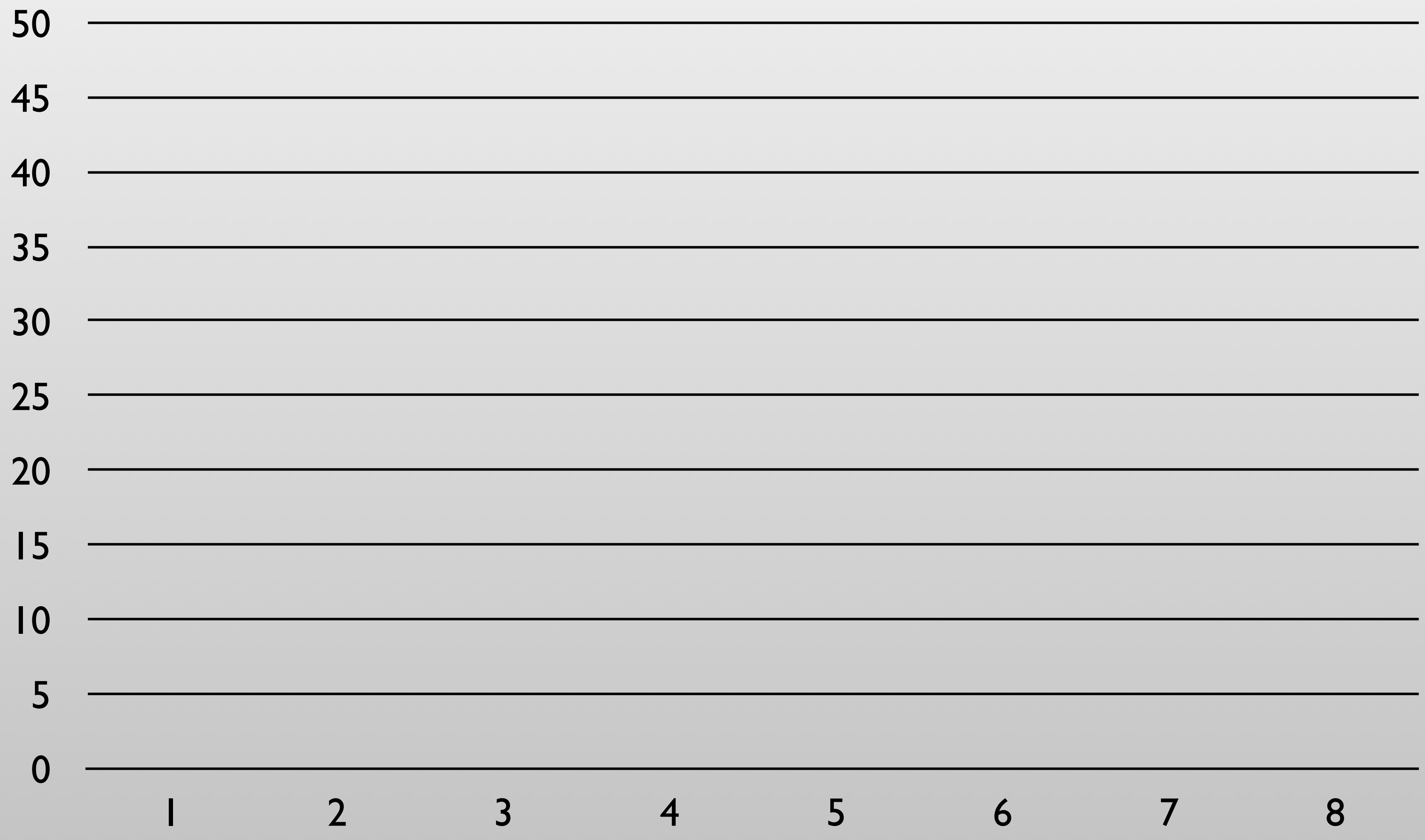
GENERAL SYSTEMANTICS PRESS WALKER MINNESOTA

Project Progress is Measurable
Functionality Built and Tested

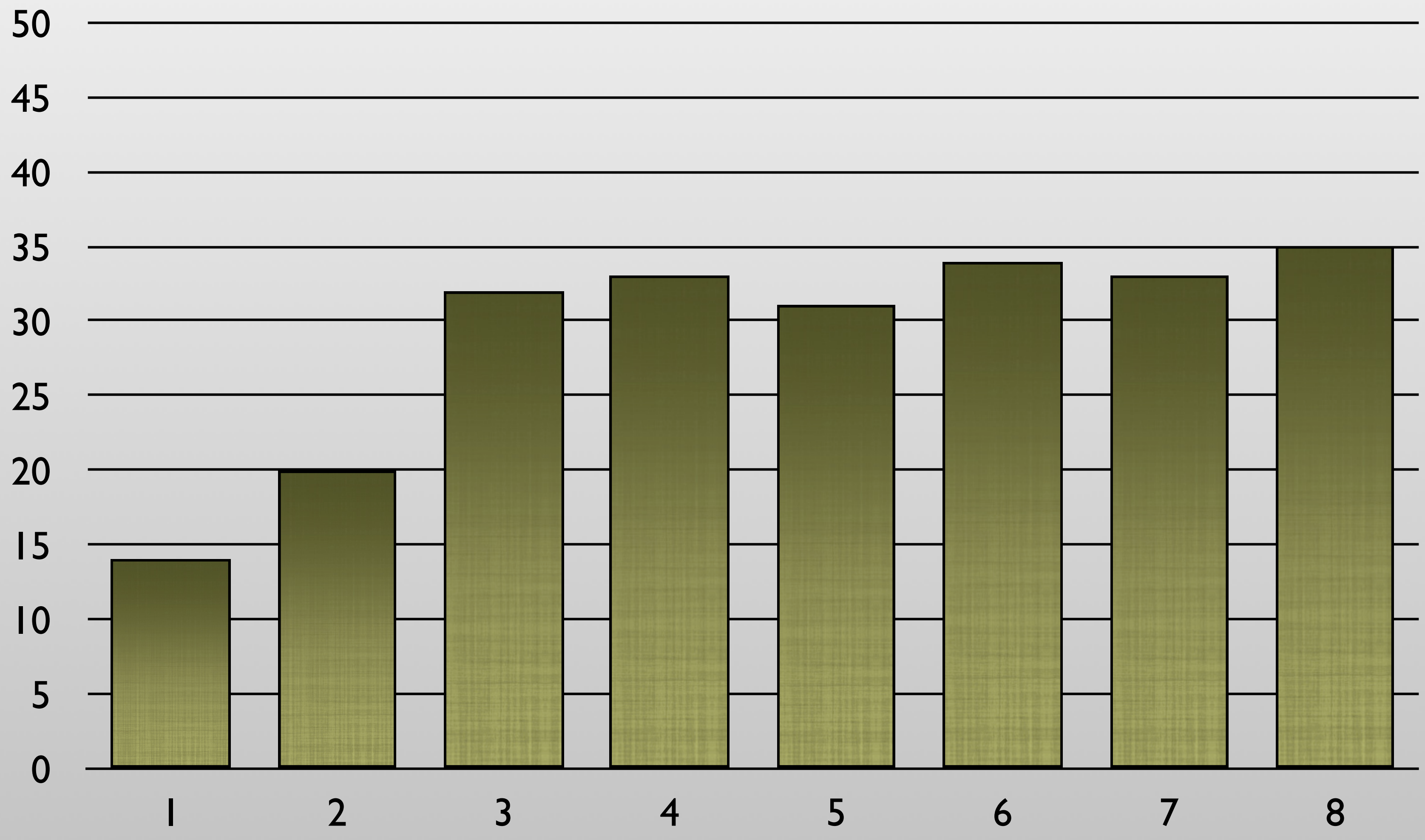


www.wingman-sw.com
james@wingman-sw.com

Measure Development Velocity
Estimated work per Iteration

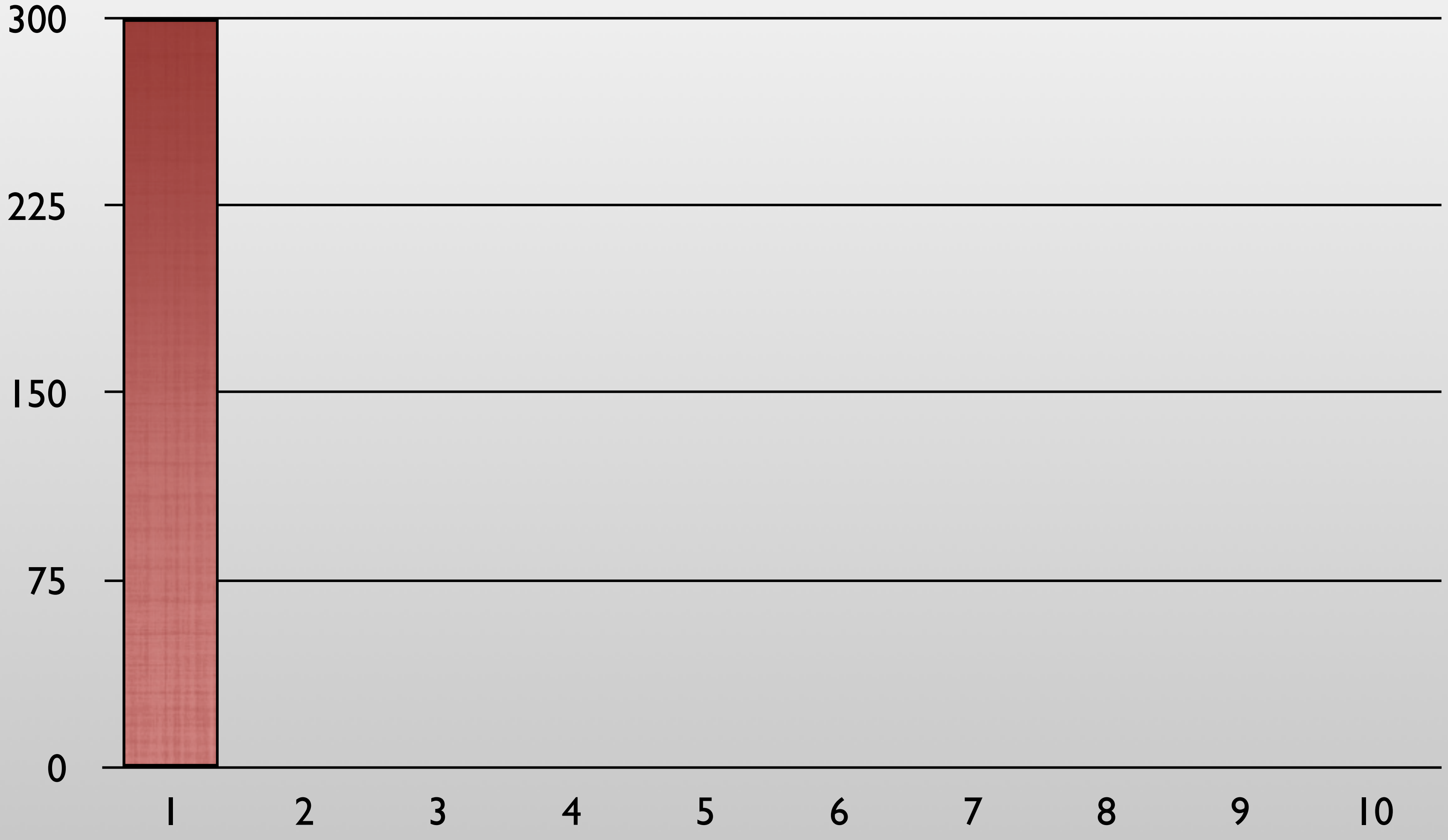


Measure Development Velocity
Estimated work per Iteration



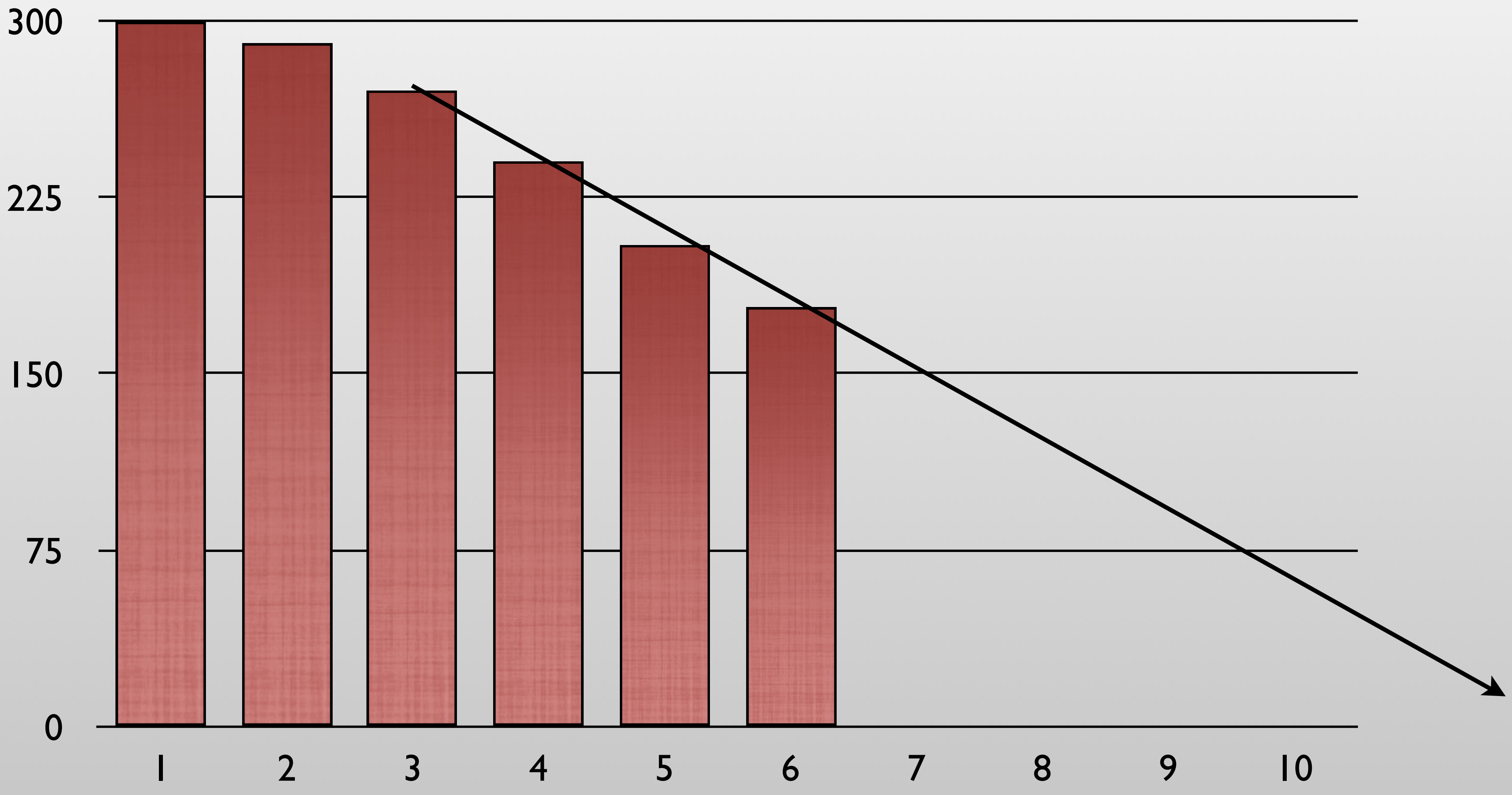
www.wingman-sw.com
james@wingman-sw.com

Product Burn Down Chart
Work to be Completed



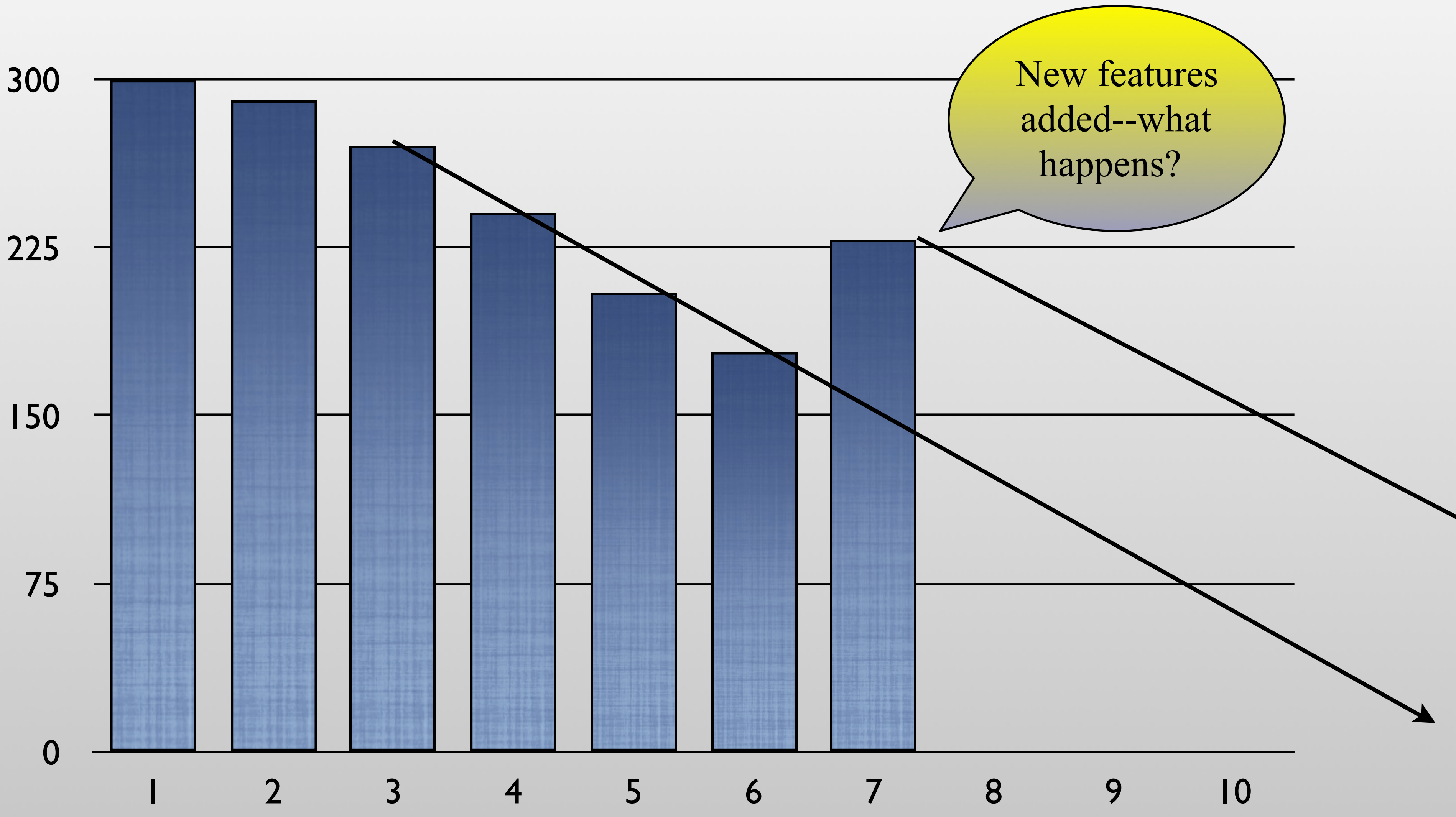
www.wingman-sw.com
james@wingman-sw.com

Product Burn Down Chart
Work to be Completed



www.wingman-sw.com
james@wingman-sw.com

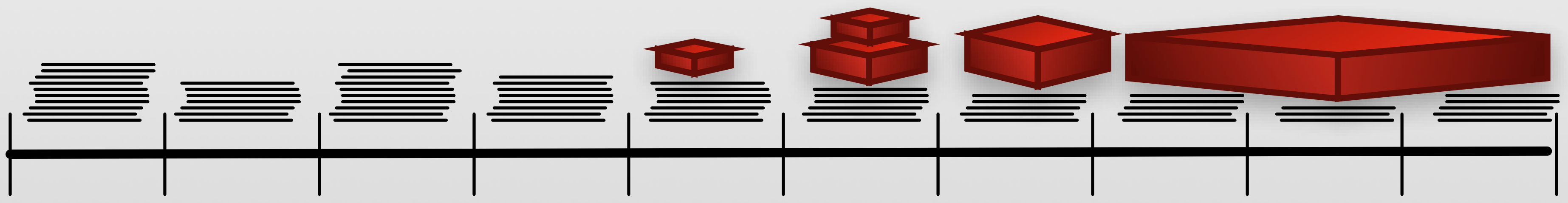
Change Becomes Visible



www.wingman-sw.com
james@wingman-sw.com

The Backlog is Made up of Stories

- The short term plan is more detailed.
- Work on it, buying time to refine longer term plan.



- Generally stories are in the order set by *customer.*
- Engineers can ask to move up stories to reduce risk.
- Stories are tested in the iteration they are implemented; story tests are automated.
- A story is done when it passes its tests.

www.wingman-sw.com
james@wingman-sw.com

Introducing the User Story

- The name of a feature.
- A promise for a conversation. (Ron Jeffries)
- Like the name of a use case, or extension.
  - Acceptance tests provide the details.
- Fine grains help make visible progress and avoid gold plating.
- I call them Product Stories

Weekend Light Schedule

Specific day Light Schedule

Weekday Light Schedule

US Holiday Light Schedule

Chinese Holiday Light Schedule

Everyday Light Schedule

Everyday-but Light Schedule

Stories and Acceptance Tests

- Stories lack detail
- Details are provided in automated acceptance tests
- The test are like executable use cases
- Test either pass or fail

Fine Grained Scope Control with Product Stories

Come to my next session ESC 227 for Details

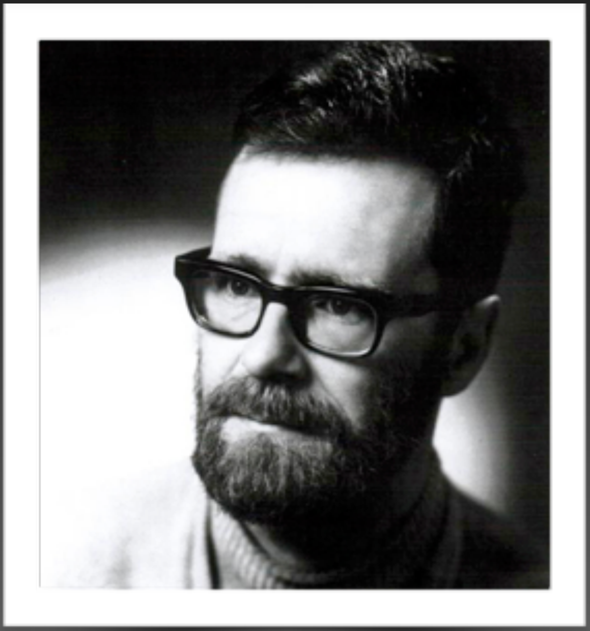
Technical Practices

High Quality - Visible Progress

- Concurrent requirements and design
- Automated test, continuously, at many levels
- Test Driven Development
- Continuous Refactoring
- Architectural Vision
- Evolutionary Design
- Continuous Integration
- Coding Standard
- Team code ownership
- Pair Programming
- Team workspace

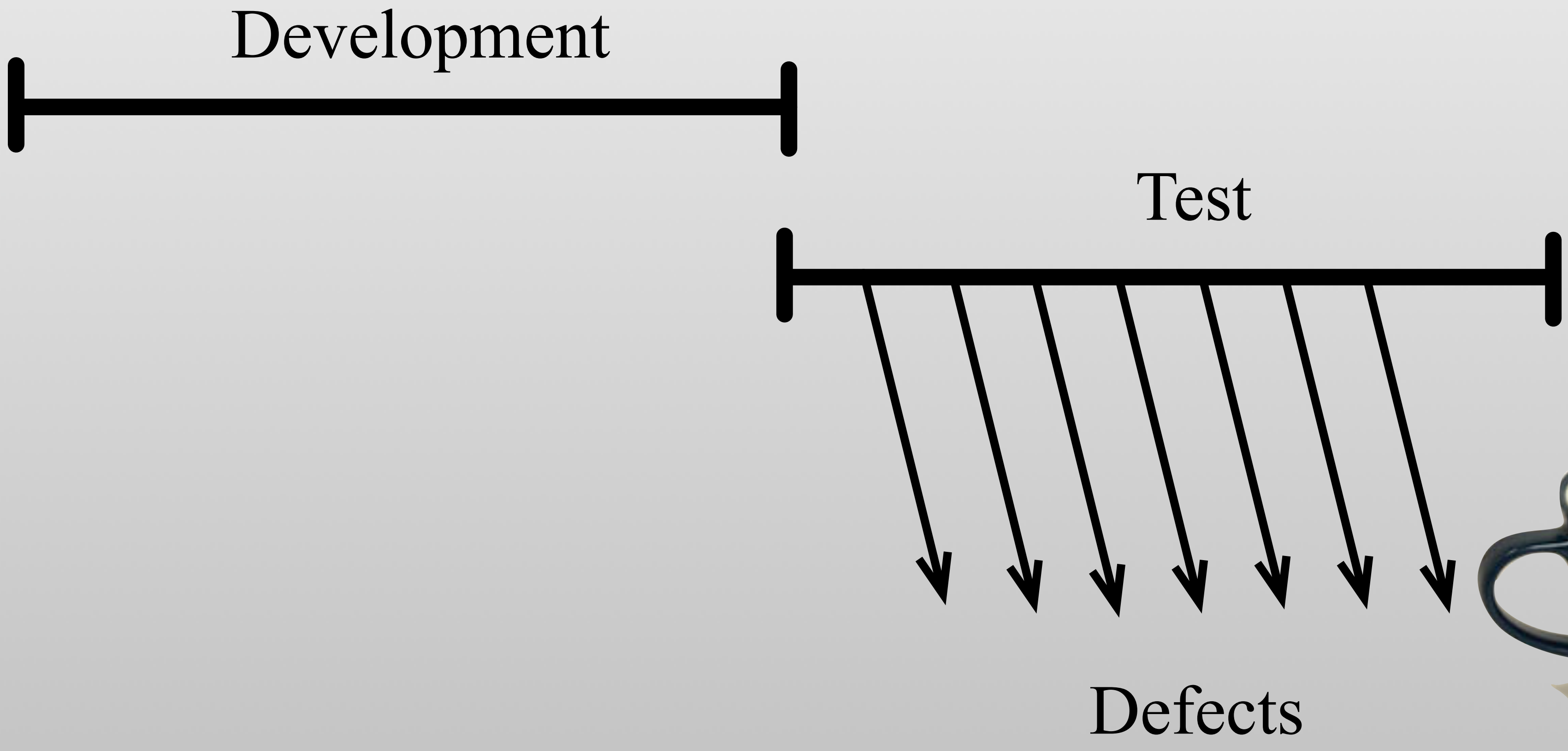
Edsger Dijkstra

Those who want really reliable software will discover that they must find means of avoiding the majority of bugs to start with, and as a result, the programming process will become cheaper. If you want more effective programmers, you will discover that they should not waste their time debugging, they should not introduce the bugs to start with.

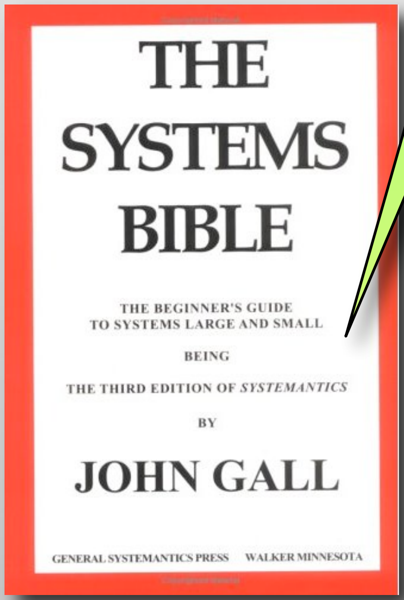


Can we Realize Dijkstra's Dream and Prevent Defects with Test Driven Development?

This Work Flow is Designed to Produce Defects



Your program will have bugs. And they will surprise you when you find them.


THE SYSTEMS BIBLE

THE BEGINNER'S GUIDE TO SYSTEMS LARGE AND SMALL

BEING

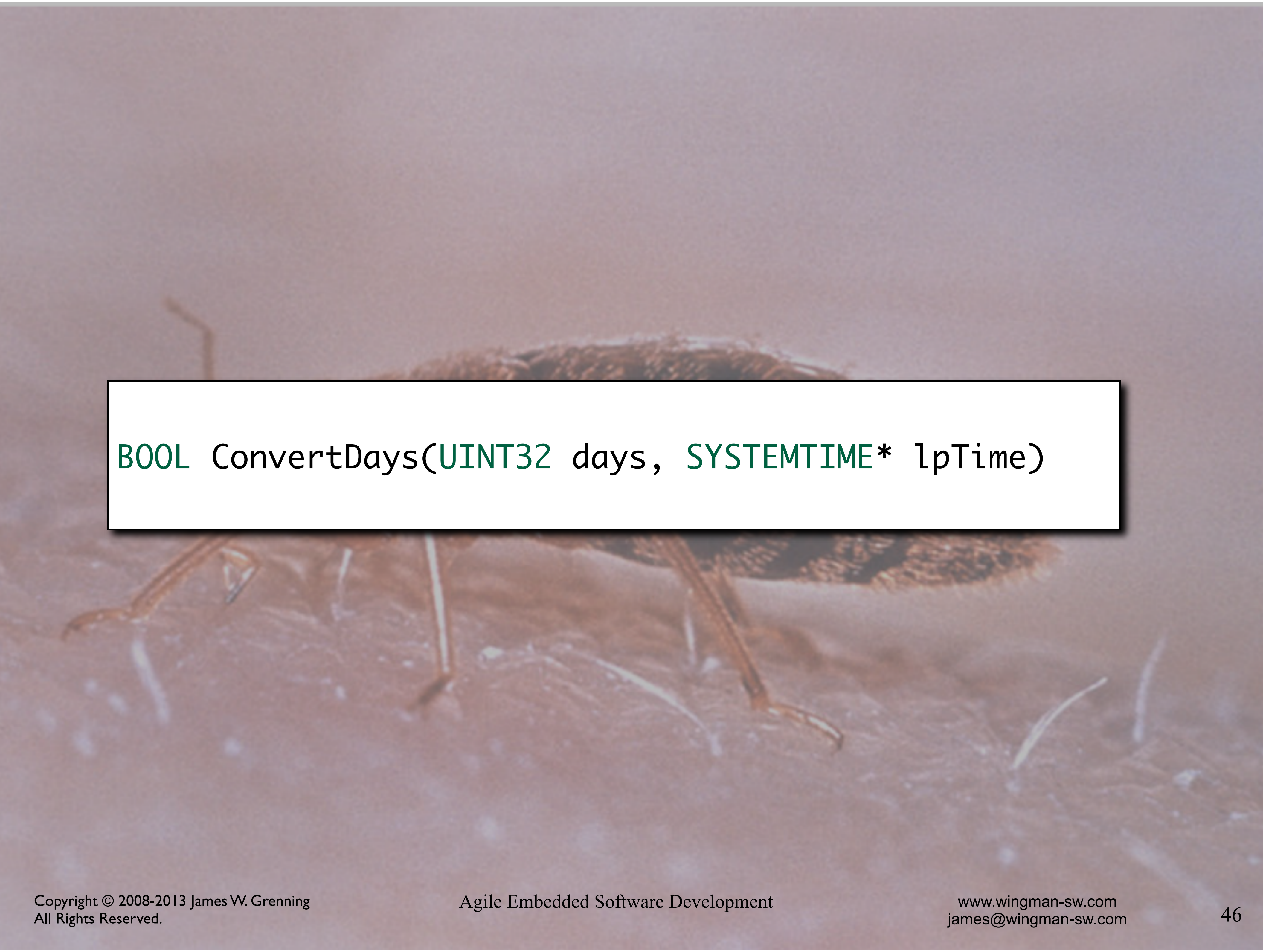
THE THIRD EDITION OF *SYSTEMANTICS*

BY

JOHN GALL

GENERAL SYSTEMANTICS PRESS WALKER MINNESOTA

2008
December 31, 2008

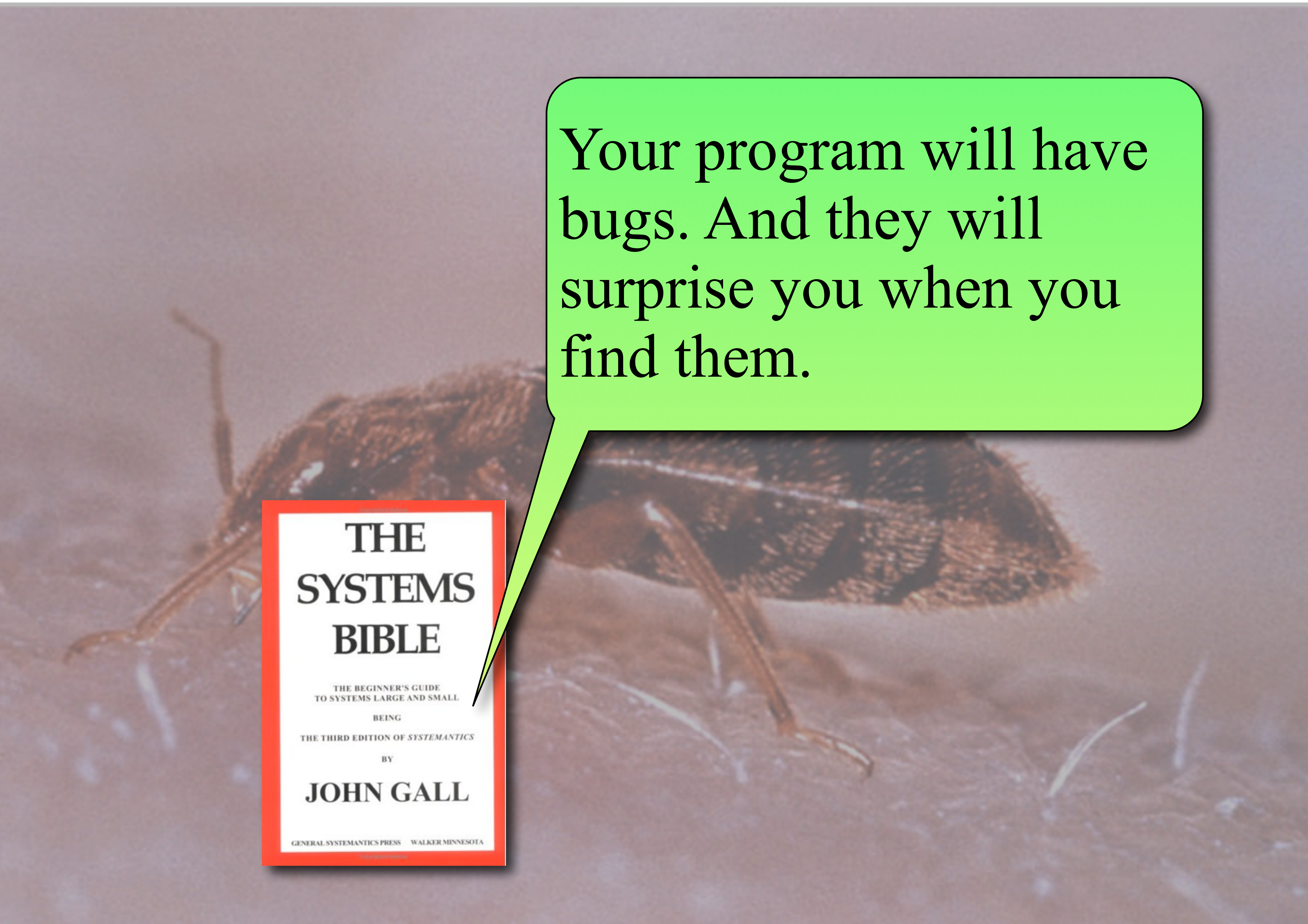


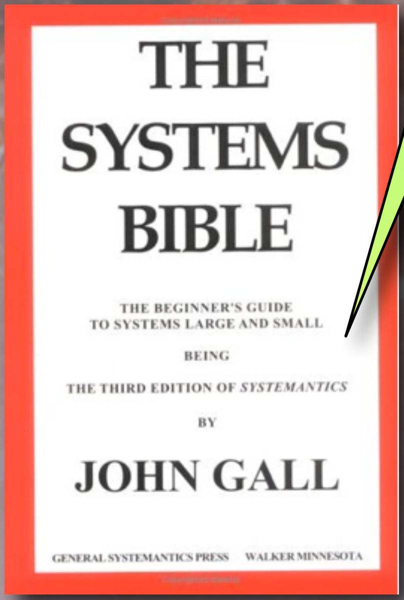
```
BOOL ConvertDays(UINT32 days, SYSTEMTIME* lpTime)
```

One That Got Away

```c
static void SetYearAndDayOfYear(RtcTime* time)
{
    int days = time->daysSince1980;
    int year = STARTING_YEAR;
    while (days > 365)
    {
        if (IsLeapYear(year))
        {
            if (days > 366)
            {
                days -= 366;
                year += 1;
            }
        }
        else
        {
            days -= 365;
            year += 1;
        }
    }

    time->dayOfYear = days;
    time->year = year;
}
```


Your program will have bugs. And they will surprise you when you find them.


THE SYSTEMS BIBLE

THE BEGINNER'S GUIDE TO SYSTEMS LARGE AND SMALL

BEING

THE THIRD EDITION OF *SYSTEMANTICS*

BY

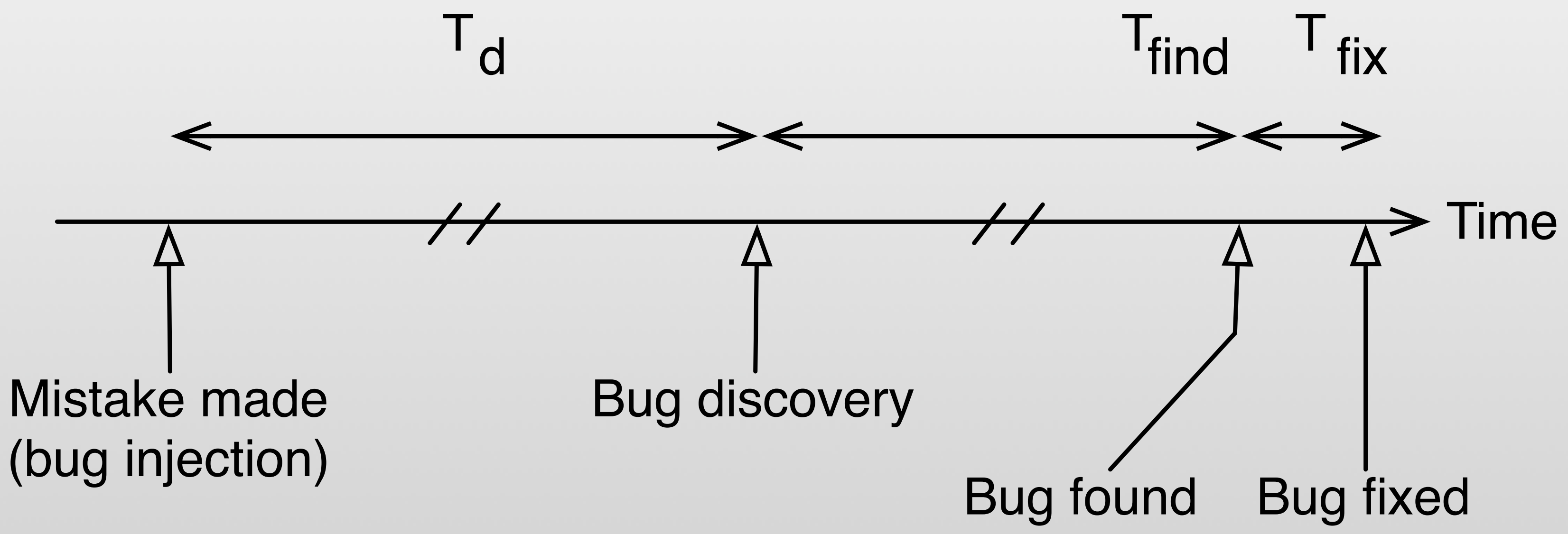
JOHN GALL

GENERAL SYSTEMANTICS PRESS WALKER MINNESOTA

This Test Could Have Prevented it

```
TEST(Rtc, check20081231)
{
    days = daysSince1980(2008, 366);
    CHECK(ConvertDays(days, &time));
    assertDate(WED, 2008, 12, 31);
}
```

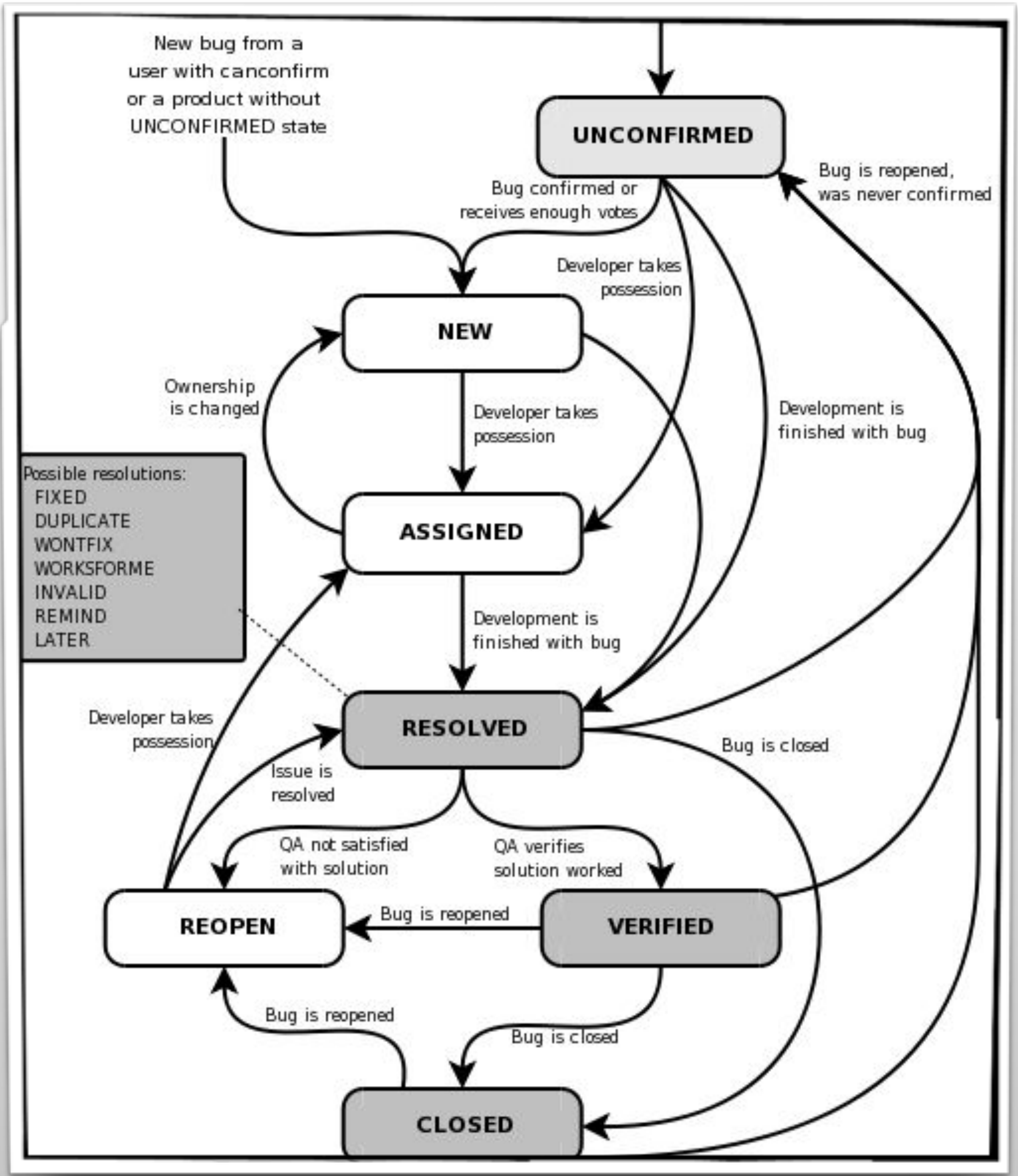
The Physics of Debug Later Programming (DLP)



- As T_d increases, T_{find} increases dramatically
- T_{fix} is usually short, but can increase with T_d

www.wingman-sw.com
james@wingman-sw.com

A Bug's Life



New bug from a user with canconfirm or a product without UNCONFIRMED state
UNCONFIRMED
Bug is reopened, was never confirmed
Bug confirmed or receives enough votes
Developer takes possession
NEW
Ownership is changed
Developer takes possession
Development is finished with bug
Possible resolutions:
FIXED
DUPLICATE
WONTFIX
WORKSFORME
INVALID
REMIND
LATER
ASSIGNED
Development is finished with bug
RESOLVED
Developer takes possession
Issue is resolved
Bug is closed
QA not satisfied with solution
QA verifies solution worked
REOPEN
Bug is reopened
VERIFIED
Bug is reopened
Bug is closed
CLOSED

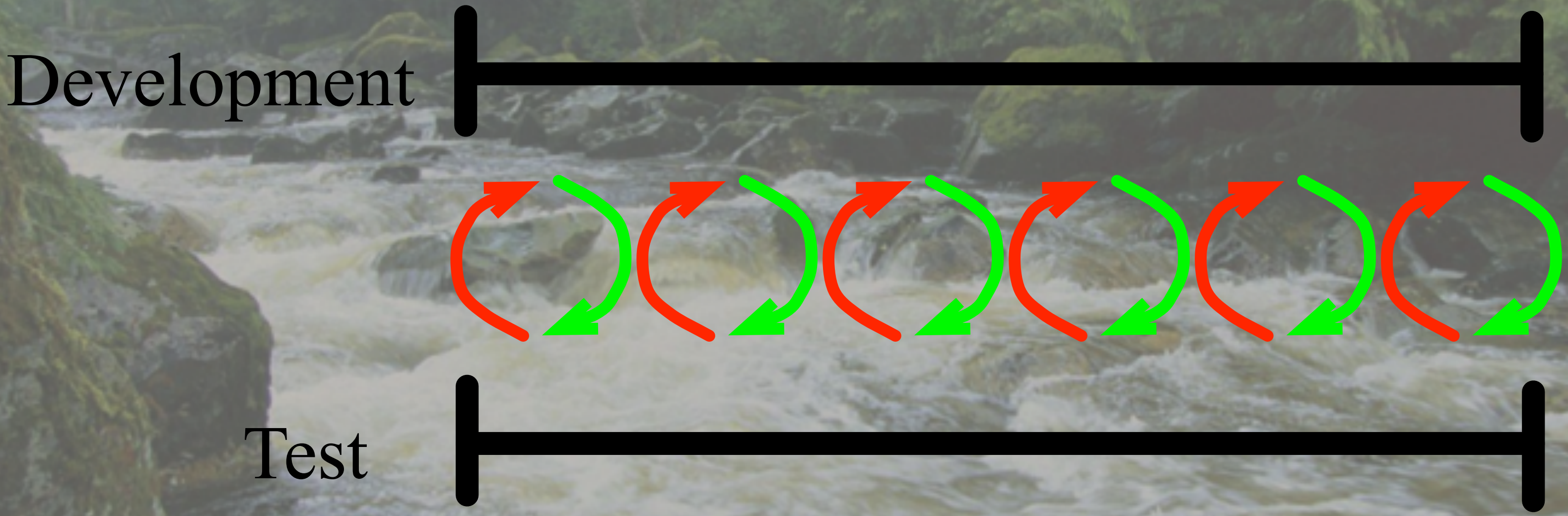
From http://www.softwaretestinghelp.com/bug-life-cycle/

T D D

- Write a test
- Watch it not build
- Make it build, but fail
- Make it pass
- Refactor (clean up any mess)
- Repeat until done

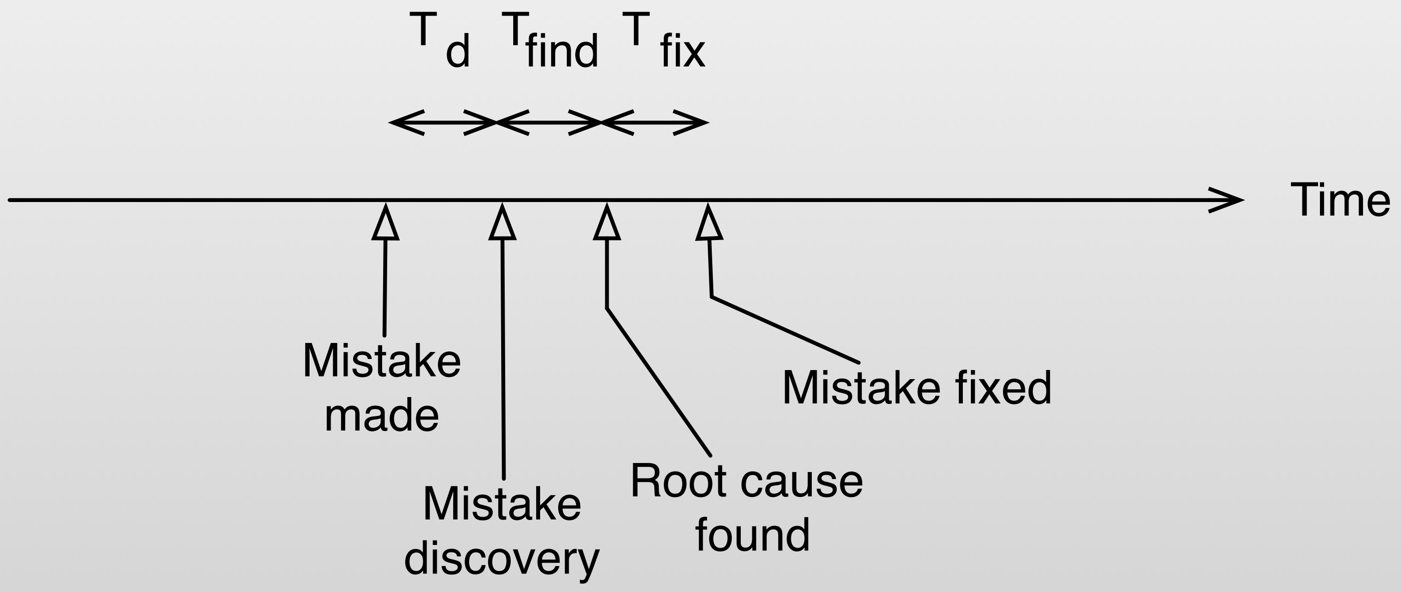
www.wingman-sw.com
james@wingman-sw.com

Development and Test are a Continuum preventing defects



www.wingman-sw.com
james@wingman-sw.com

The Physics of Test Driven Development

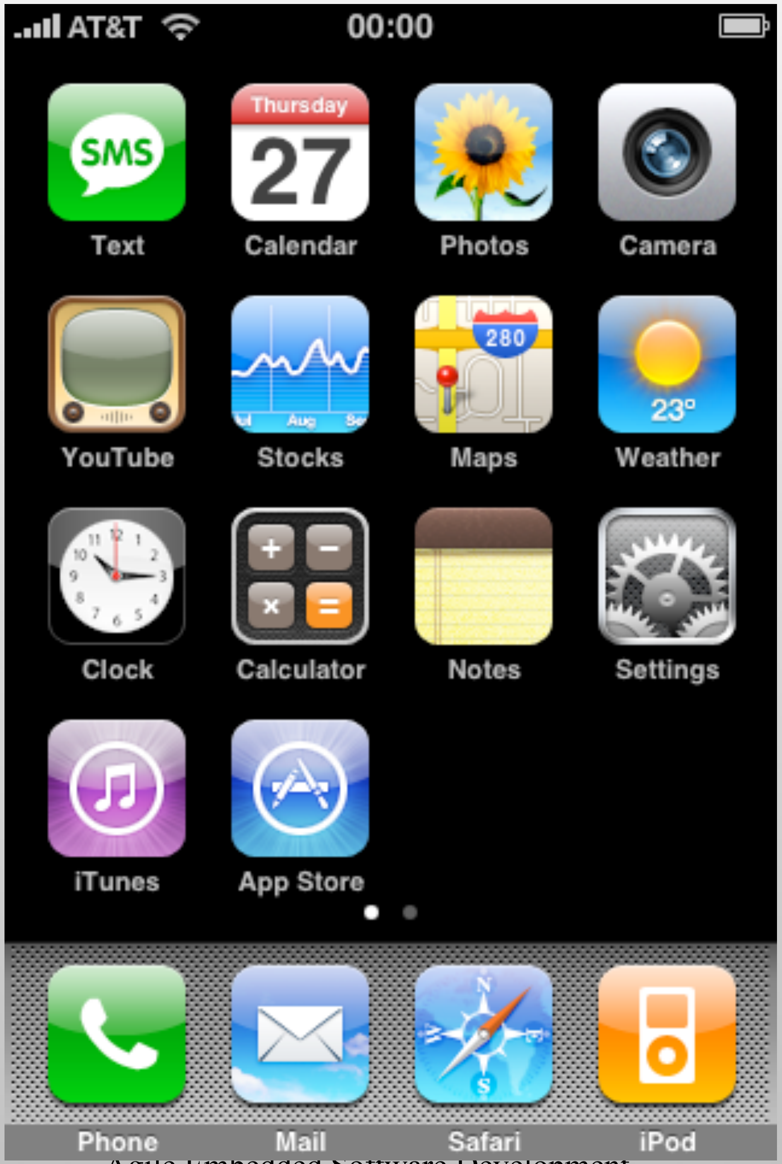


- When T_d approaches zero, T_{find} approaches zero
- In many cases, bugs are not around long enough to be considered bugs.
- See: http://www.renaissancesoftware.net/blog/archives/16

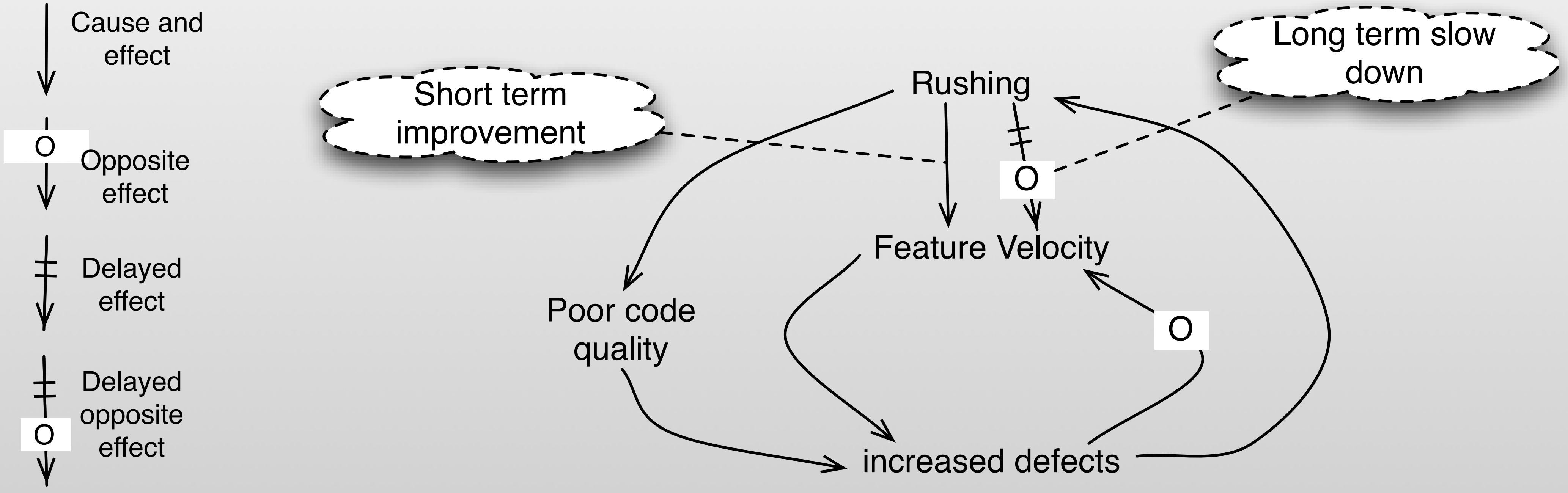
Testing is not a Phase

- Testing starts on day one
- Tests provide the specification of what is to be developed
- QA/System Test moves upstream.

The Two Values of Software



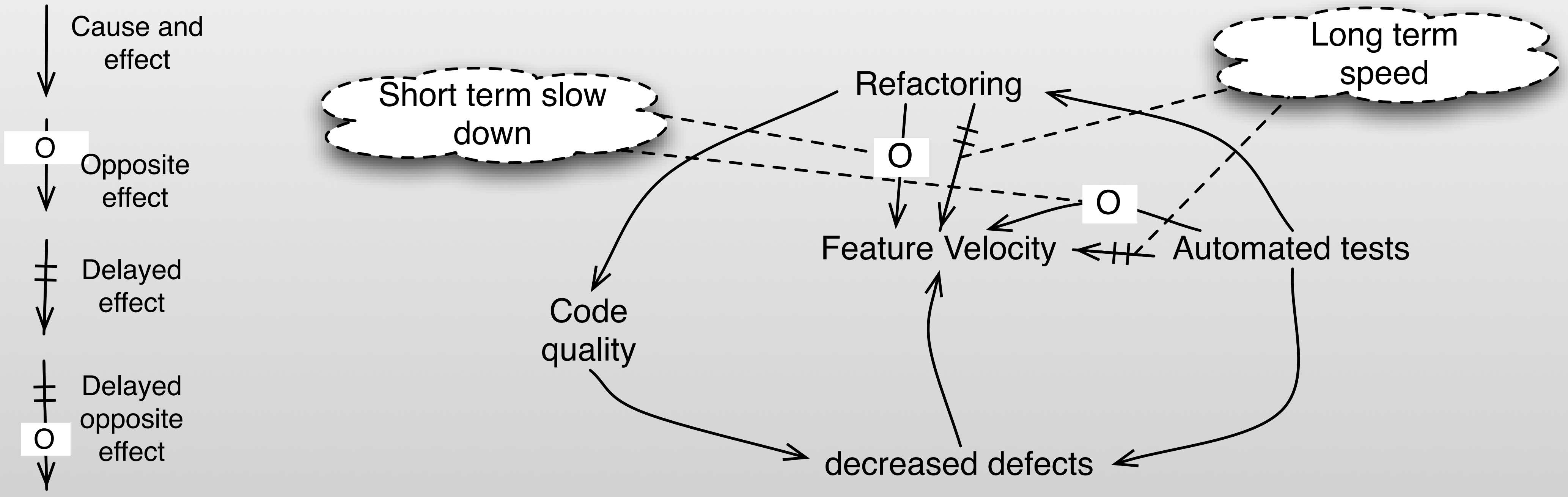
Rushing Slows You Down



Copyright (c) 2009 James Grenning

Inspired by Scaling Lean and Agile Development [SLAD]

Slow Down to Go Faster

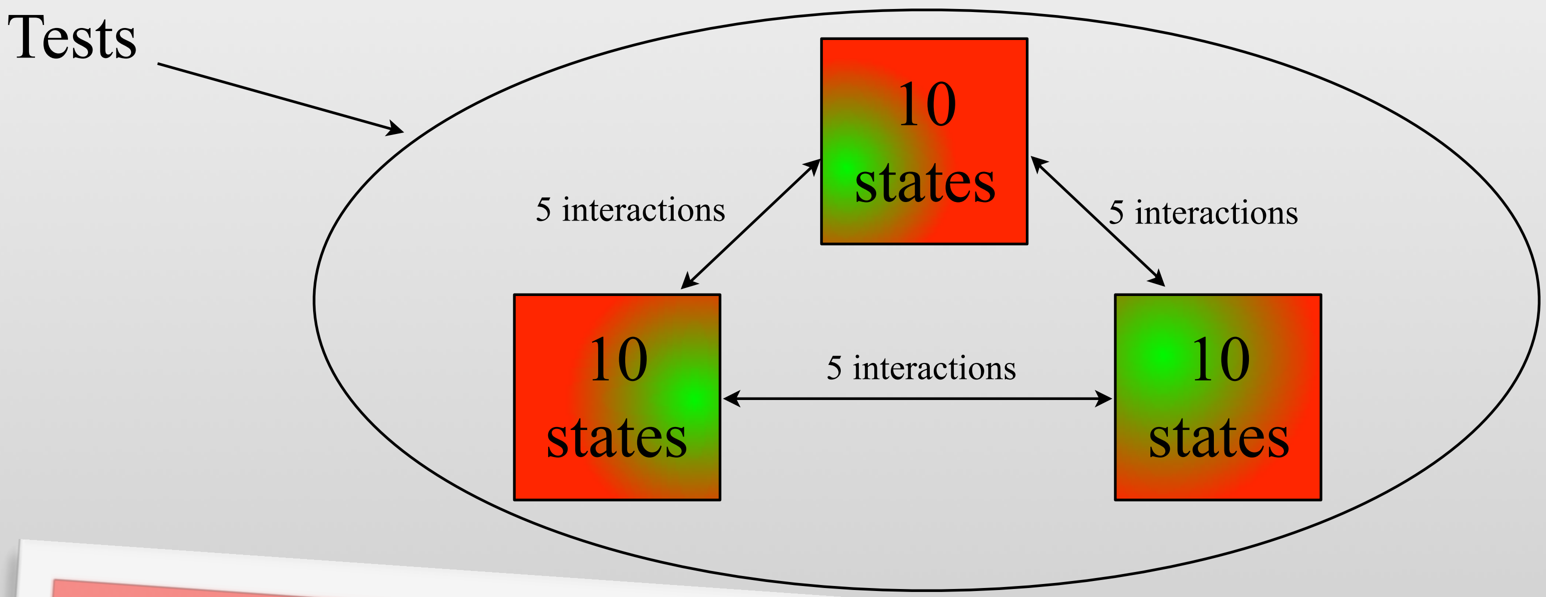


Copyright (c) 2009 James Grenning

Inspired by Scaling Lean and Agile Development [SLAD]

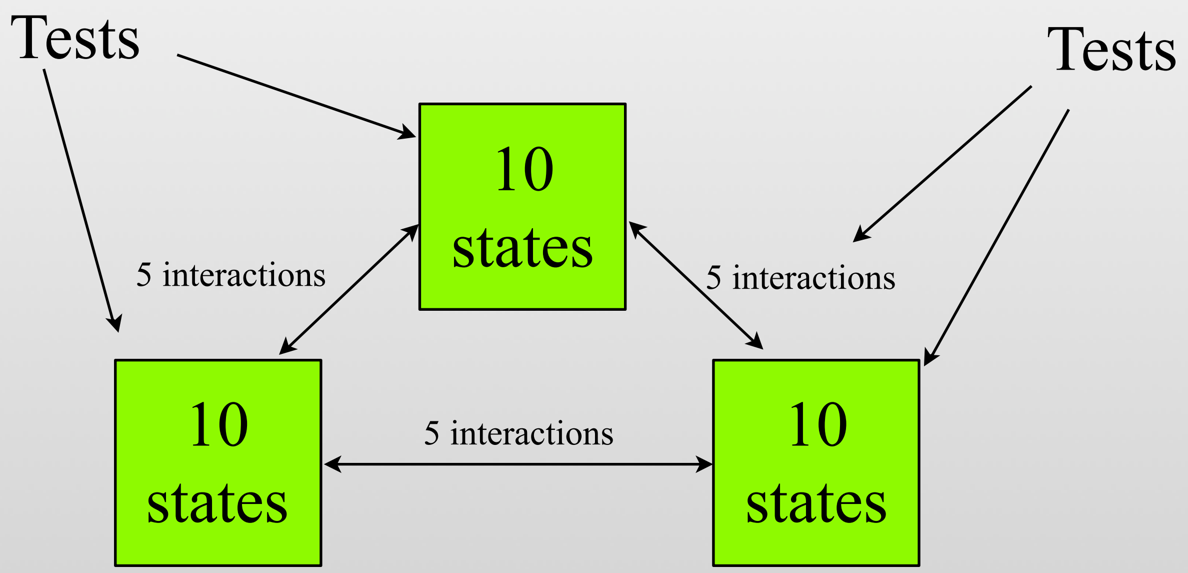
Unit Tests are Critical

Higher Level Tests Cannot be Broadly Thorough



1000 (or more) tests are needed to test this simple system

Unit Tests Can Be Thorough



As few as 30 unit tests and 15 integration test when tested as units

Manual Test is Not Sustainable

$E_t = f(E_d)$

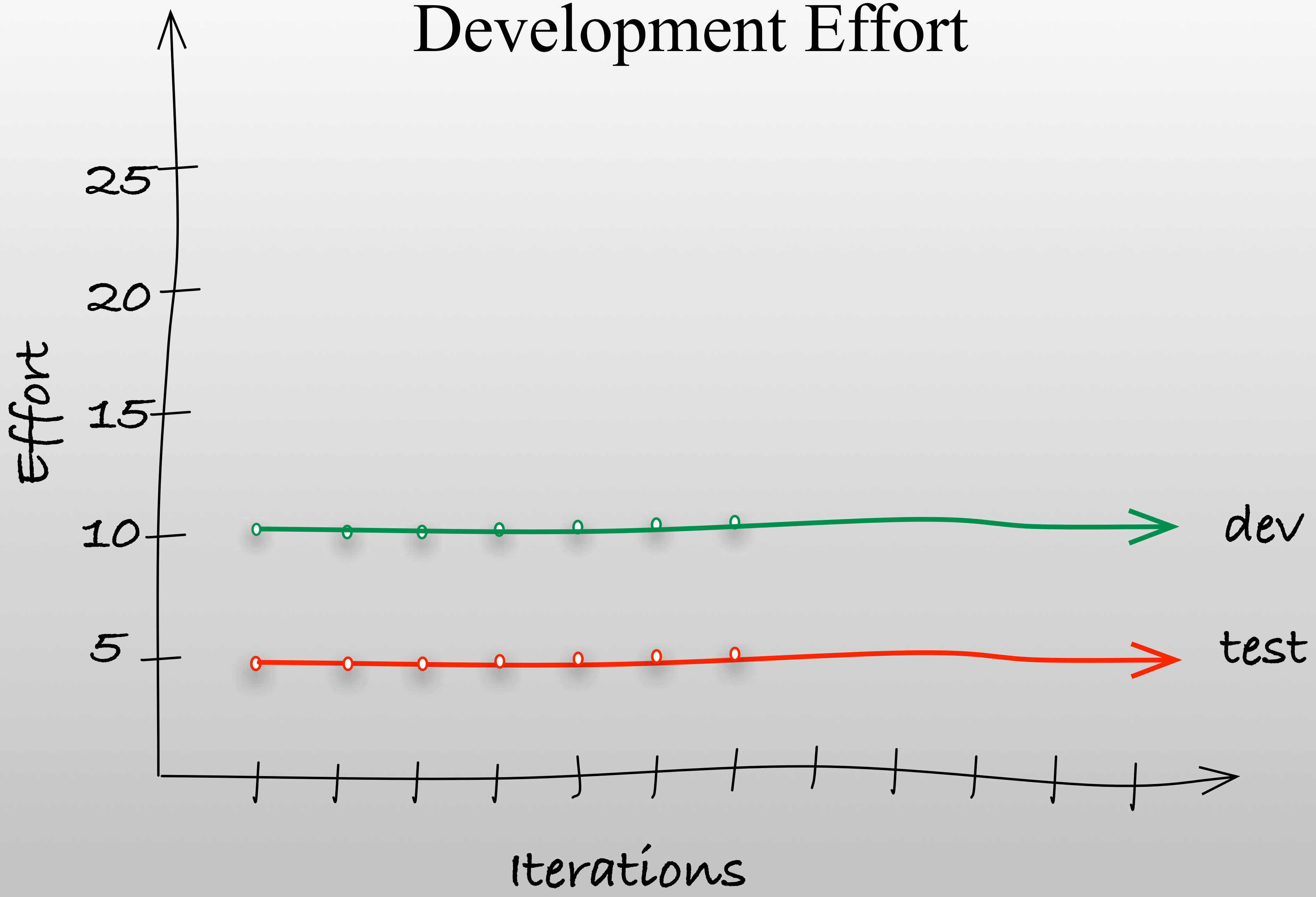
E_t is the effort to test a new feature, and is a function of the effort to develop the feature.

E_d is the effort to develop a new feature

Assume a constant linear relationship

$$E_t = f(E_d) = KE_d$$

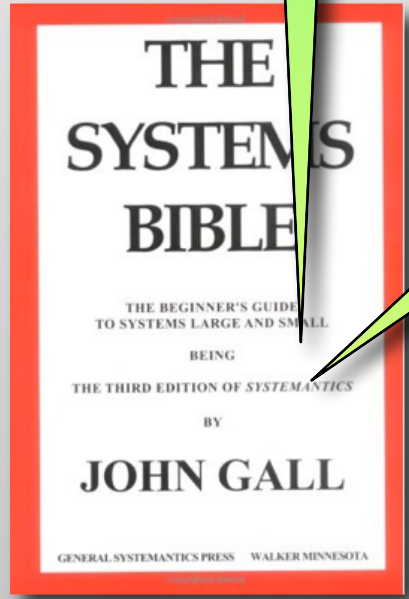
Assume Test Effort is Proportional to Development Effort



www.wingman-sw.com
james@wingman-sw.com

If a system is working, leave it alone. Don't change anything

Systems don't appreciate being fiddled and diddled with


THE SYSTEMS BIBLE
THE BEGINNER'S GUIDE TO SYSTEMS LARGE AND SMALL
BEING
THE THIRD EDITION OF SYSTEMANTICS
BY
JOHN GALL
GENERAL SYSTEMANTICS PRESS WALKER MINNESOTA

- 25% of all defects are introduced while changing and fixing code
[R.B Grady, Software Process Improvement]

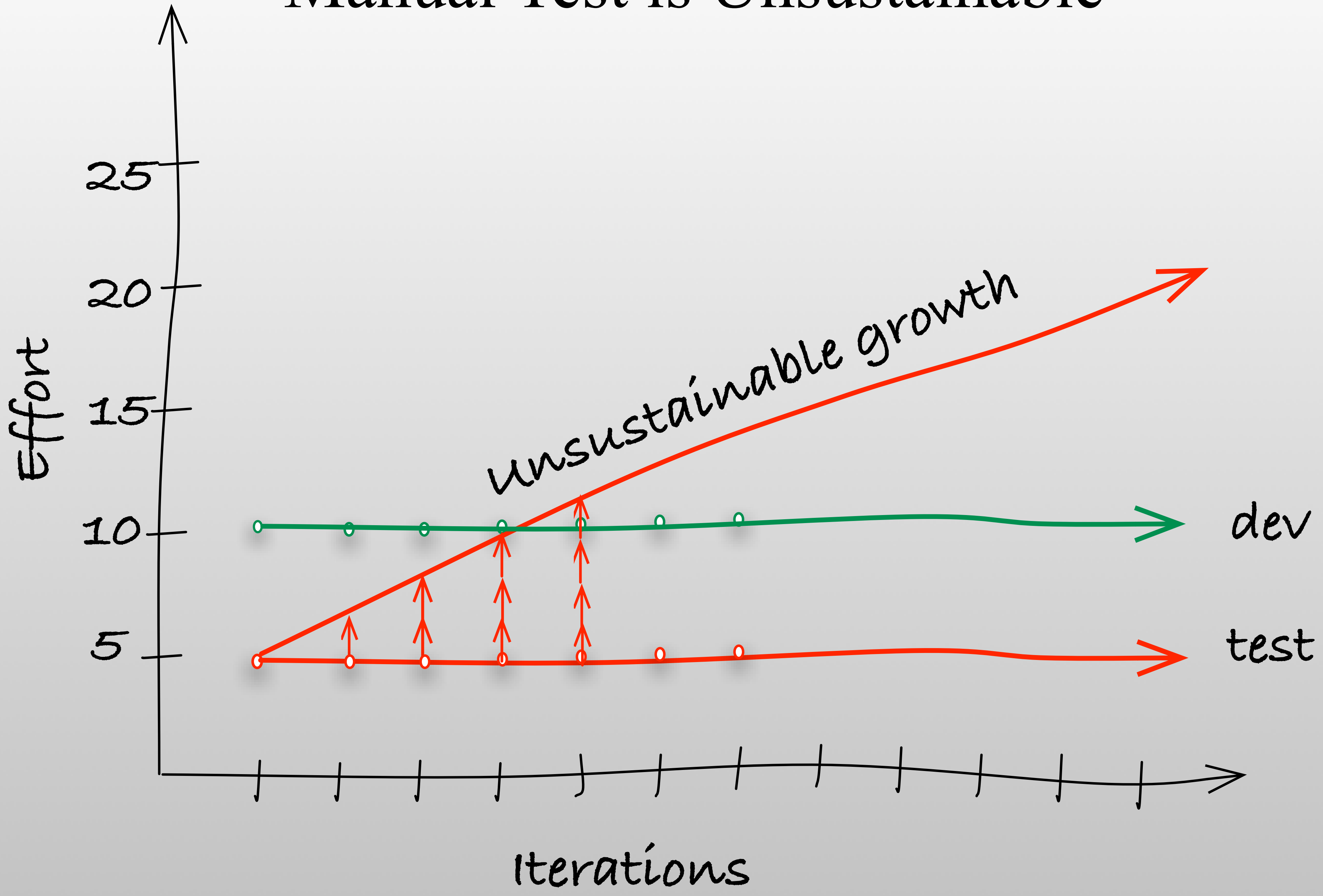
$$E_{tn} = f(E_d) + \sum_{i=0}^{n} C\,Et(i)$$

E_{tn} is the effort to fully test a product at iteration N

Because any change can break previously working software, we must retest.

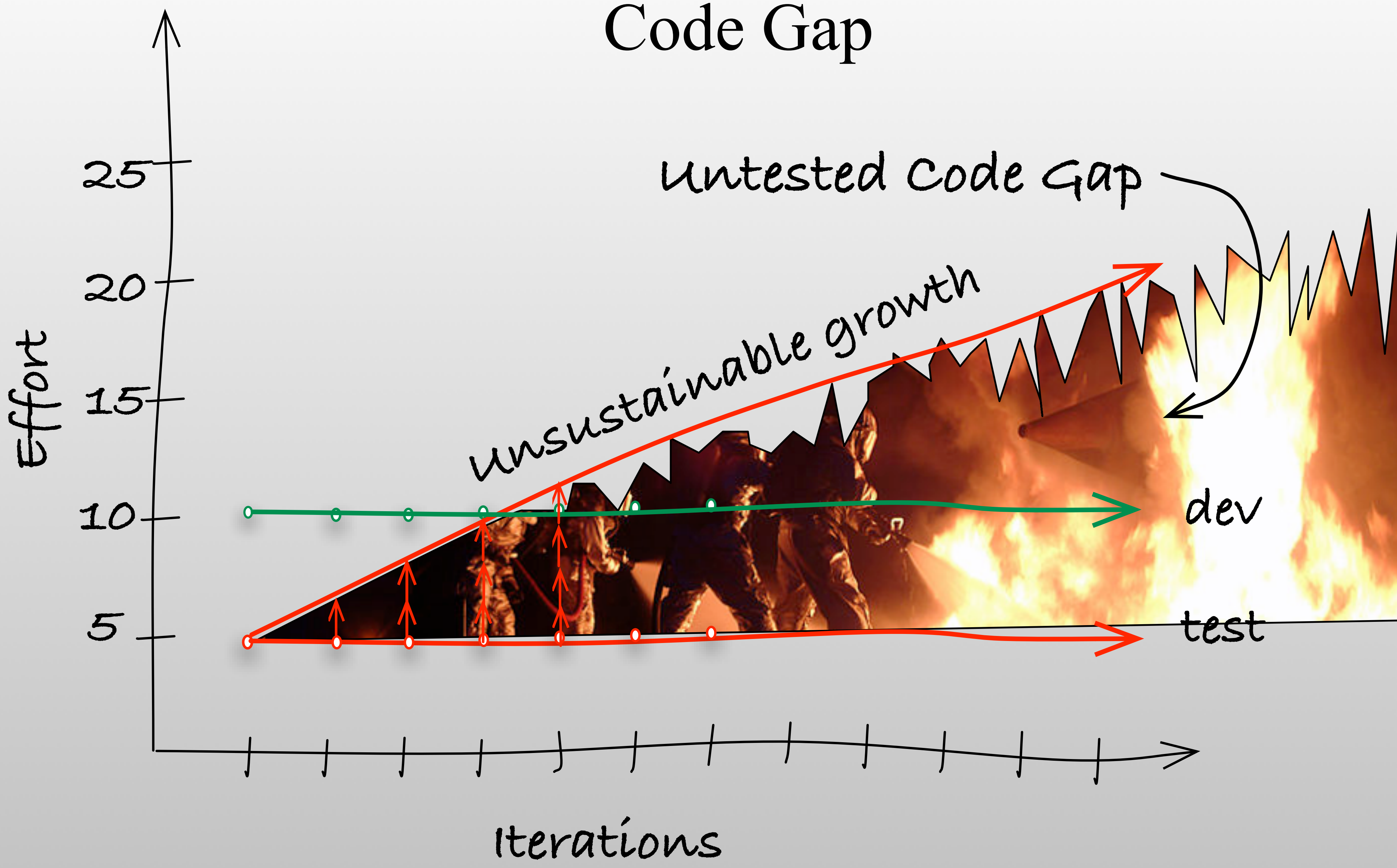
E_{tn} is a function of the effort to develop the feature plus some fraction of the effort to test all previous iterations.

Manual Test is Unsustainable



www.wingman-sw.com
james@wingman-sw.com

Risk Accumulates in the Untested Code Gap



www.wingman-sw.com
james@wingman-sw.com

Final Thoughts

Challenges

- Legacy C code
- Automated testing
- Stories
- Culture
- Getting out of the cube and into the team
- Engineers get too specialized

www.wingman-sw.com
james@wingman-sw.com

Getting Started

- Honest self-assessment
- Motivation for change
- Open to different approaches
- Learn
- Experiment
- TDD under the radar
- Stories for individual work
- Management support

References and Resources

- [XP] Kent Beck, Extreme Programming Explained, 1999
- [REF] Martin Fowler. Refactoring. Improving the Design of Existing Code. 1999
- [WELC] Michael Feathers, Working Effectively with Legacy Code
- [TDD] Kent Beck, Test-Driven Development, 2003
- [XUNIT] Gerard Meszaros, xUnit Testing Patterns, 2008
- [PRAG] Andy Hunt, Dave Thomas, The Pragmatic Programmer
- [SLAD] Craig Larman and Bas Voode, Scaling Lean & Agile Development
- [POP] Mary Poppendieck and Tom Poppendieck, Implementing Lean Software Development: From Concept to Cash, 2006
- [AGILE] Robert C. Martin, Agile Software Development: Principles, Patterns, and Practices, 2002
- [CLEAN] Robert C. Martin, Clean Code, 2008
- [KANER] Cem Kaner, et. al. Lessons learned in Software Testing
- [TD] Lasse Koskela, Test Driven, 2007

www.wingman-sw.com
james@wingman-sw.com

On-line

- Test harnesses
  - [CPPTEST] www.sourceforge.org, project CppUTest
  - [FITNESSE] www.fitnesse.org
- Groups
  - http://groups.yahoo.com/group/testdrivendevelopment
  - http://groups.yahoo.com/group/AgileEmbedded

See Embedded TDD and Related Blogs and Papers

http://www.renaissanceSoftware.net
http://www.renaissancesoftware.net/blog/

- Embedded TDD
- Zune Bug: Test Driven Bug Fix
- Learning Tests are Free!
- TDD as a Design Rot Prevention System
- Crashing Your Way to Great Legacy C Tests
- TDD and the Big Framework Part
- Bug Fixes and TDD
- Physics of Test Driven Development
- Tests vs. Short Term Cache Between Your Ears
- Embedded Systems Conference FAQ
- I miss constructors
- Who says you can’t test drive a device driver?
- Why are You Still Using C?
- Planing Poker
- Agile Embedded Software Development (ESC)
- Launching Extreme Programming at a Process Intensive Company (IEEE)
- Test Driven Development for Embedded Software
- Progress Before Hardware
- Agile Times - Containing Progress Before Hardware
- Test-Driven Development for Embedded C++ Programmers

Renaissance Software Consulting

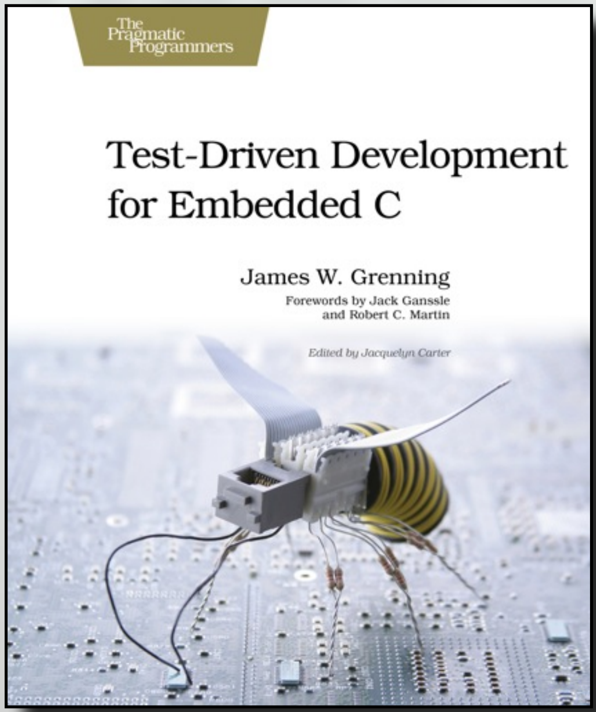
Please come to my other sessions

Talk to me on Twitter
http://twitter.com/jwgrenning

Find my book at
http://www.pragprog.com/titles/jgade

Find me on linkedin.com
http://www.linkedin.com/in/jwgrenning
Please remind me how we met.

http://www.renaissancesoftware.net
http:// www.jamesgrenning.com


The Pragmatic Programmers
Test-Driven Development for Embedded C
James W. Grenning
Forewords by Jack Ganssle and Robert C. Martin
Edited by Jacquelyn Carter