WINGMAN SOFTWARE

Story Testing

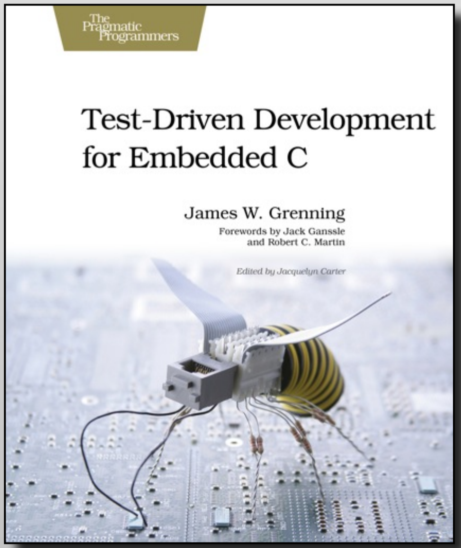
Executable Use Cases

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

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

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

http://www.wingman-sw.com
http:// www.jamesgrenning.com


The Pragmatic Programmers

Test-Driven Development for Embedded C

James W. Grenning

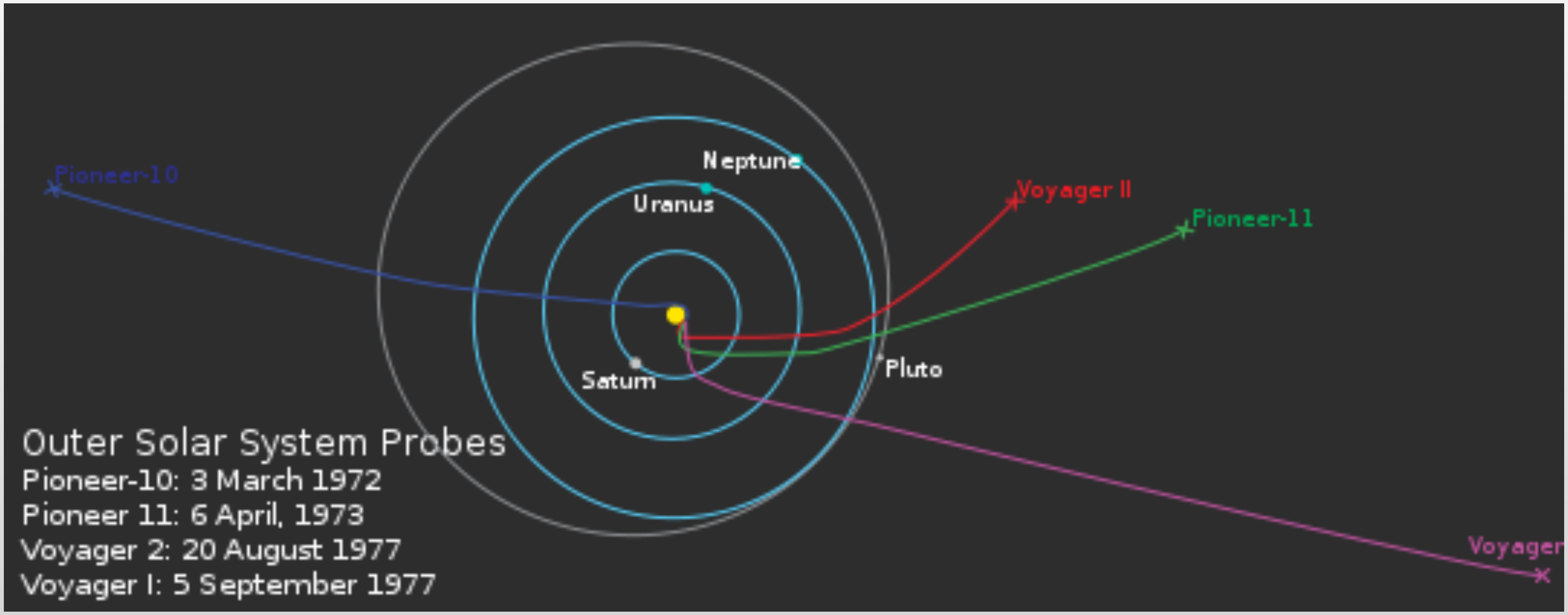
Forewords by Jack Ganssle and Robert C. Martin

Edited by Jacquelyn Carter

The Nature of Software

- Software is fragile
  - –The nature of discrete systems
- Any change can break just about anything
- Test and forget model leads to big surprises and problems

Voyager



Outer Solar System Probes
Pioneer-10: 3 March 1972
Pioneer 11: 6 April, 1973
Voyager 2: 20 August 1977
Voyager I: 5 September 1977

$$E_t = f(E_d)$$

Effort to test a new feature

$$E_t$$

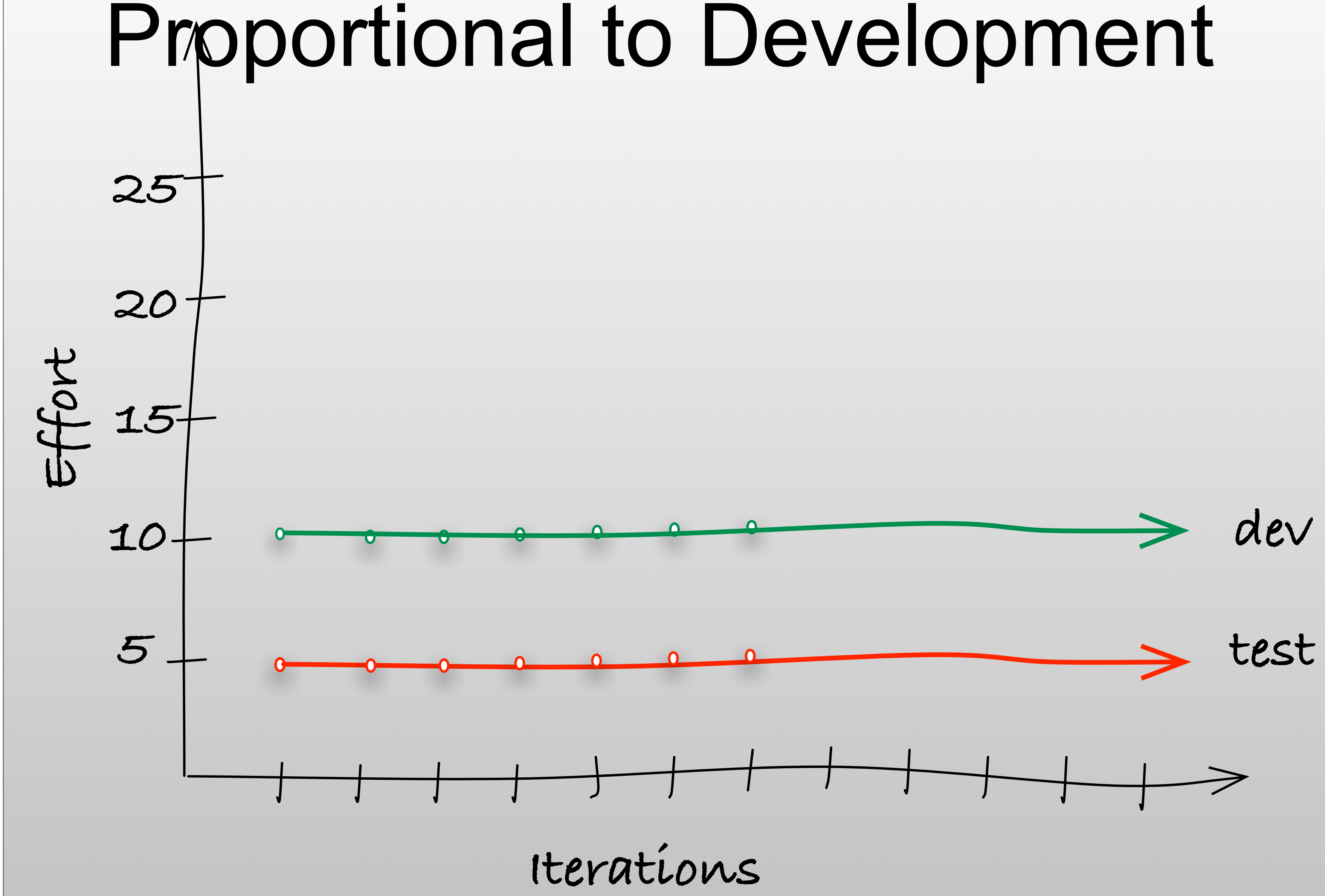
is a function of the effort to develop the feature.

$$f(E_d)$$

Assume a linear relationship

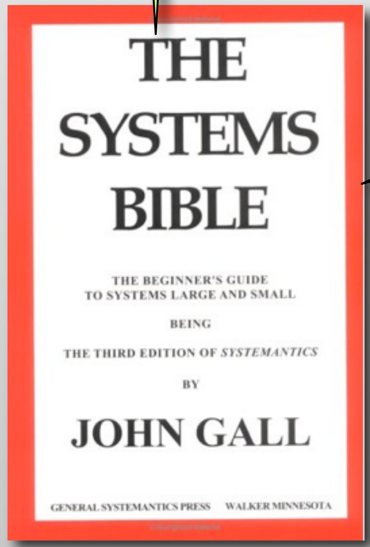
$$E_t = KE_d$$

Assume Test Effort is Proportional to Development



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]

$$E_{tn} = f(E_d) + g(E_{t(n-1)})$$

Effort to fully test a product at iteration N

$$E_{tn}$$

is a function of the effort to develop the feature

$$f(E_d)$$

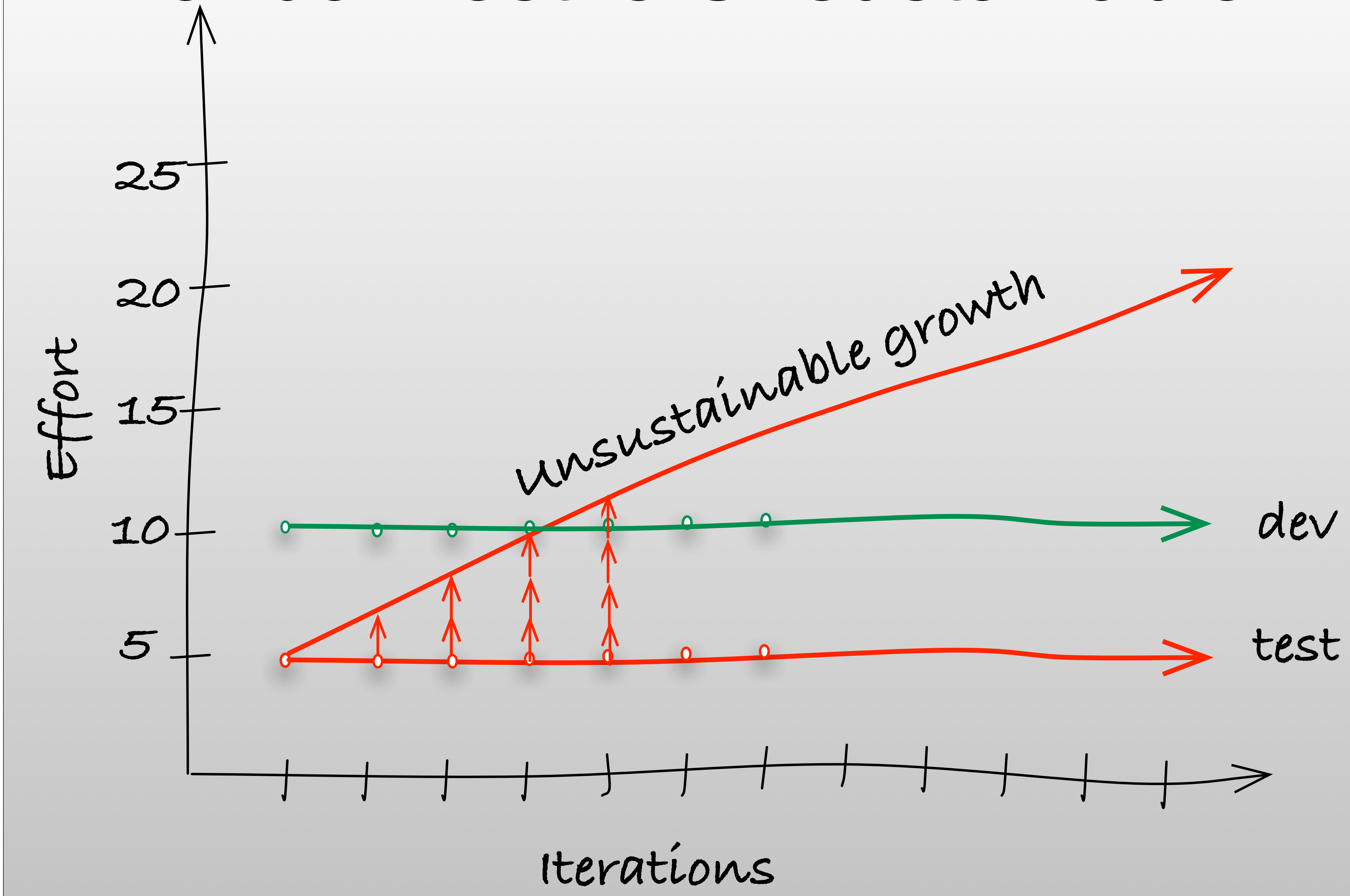
plus a function of the effort to test the previous iteration

$$g(E_{t(n-1)})$$

Assume a linear, and recursive, relationship

$$E_{tn} = CE_{t(n-1)}$$

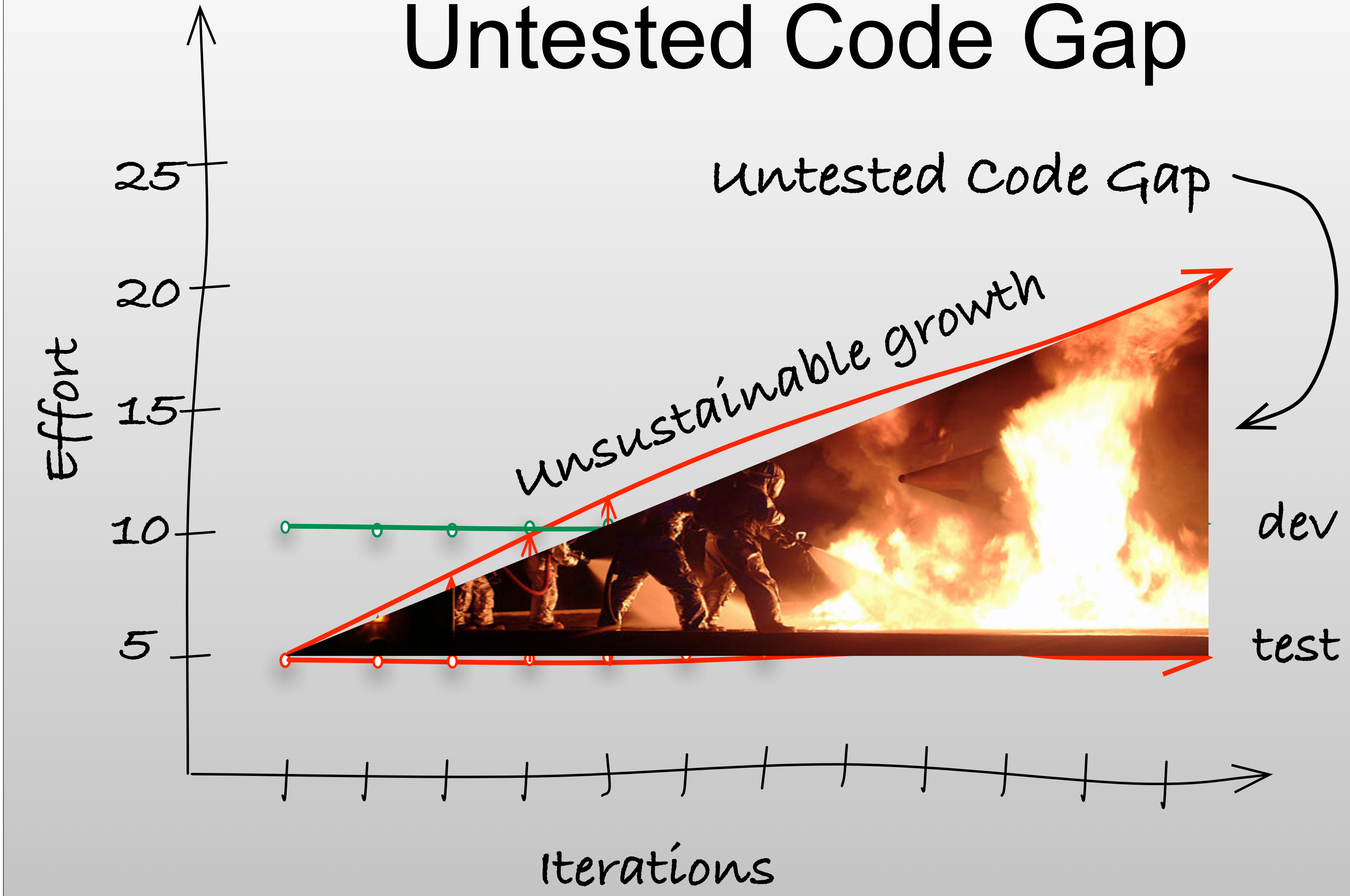
Manual Test is Unsustainable



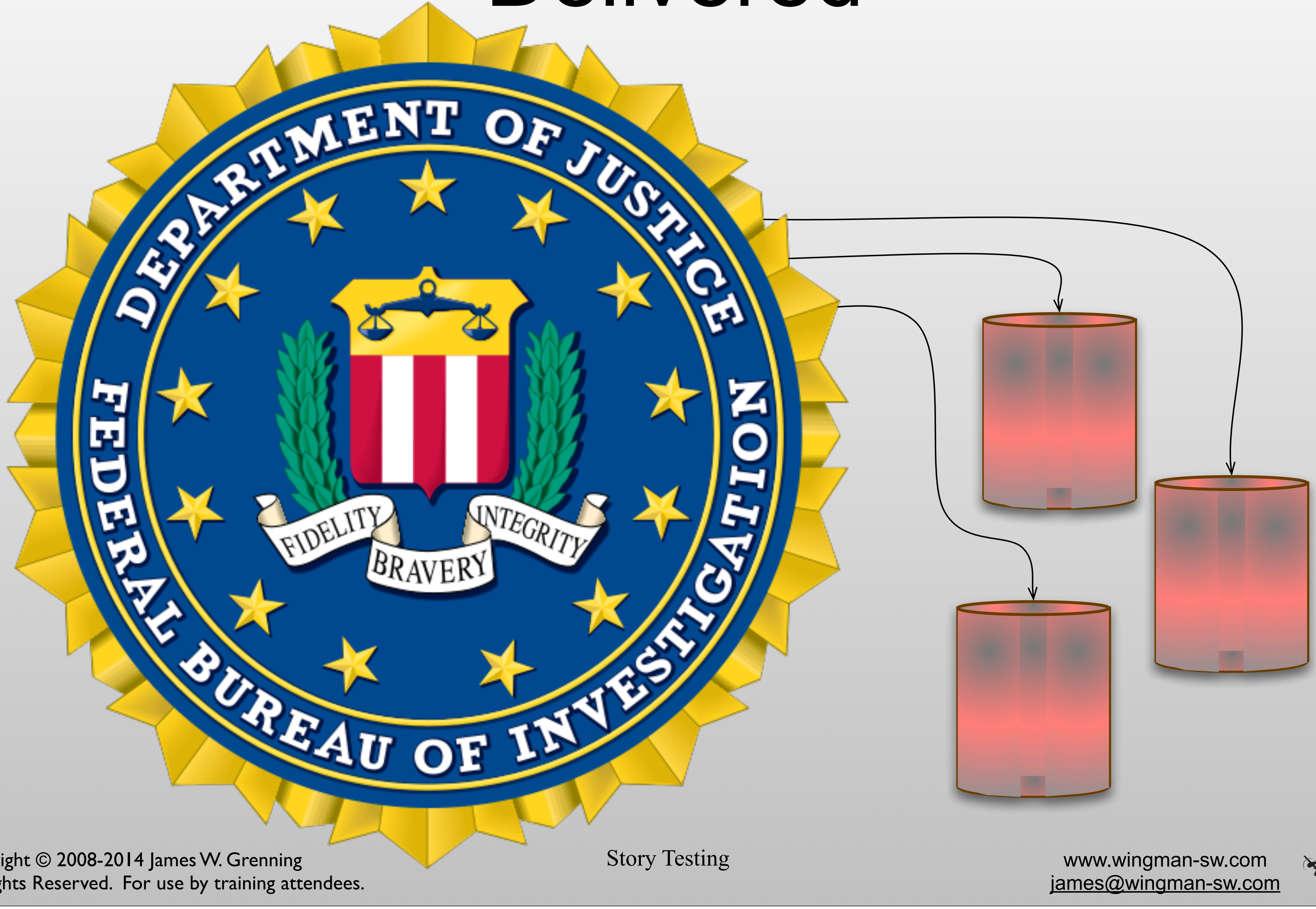
As the Effort to Test Grows...

- We only do partial regression tests
- Unintended side-effects (bugs) go unnoticed.

Risk Accumulates in the Untested Code Gap

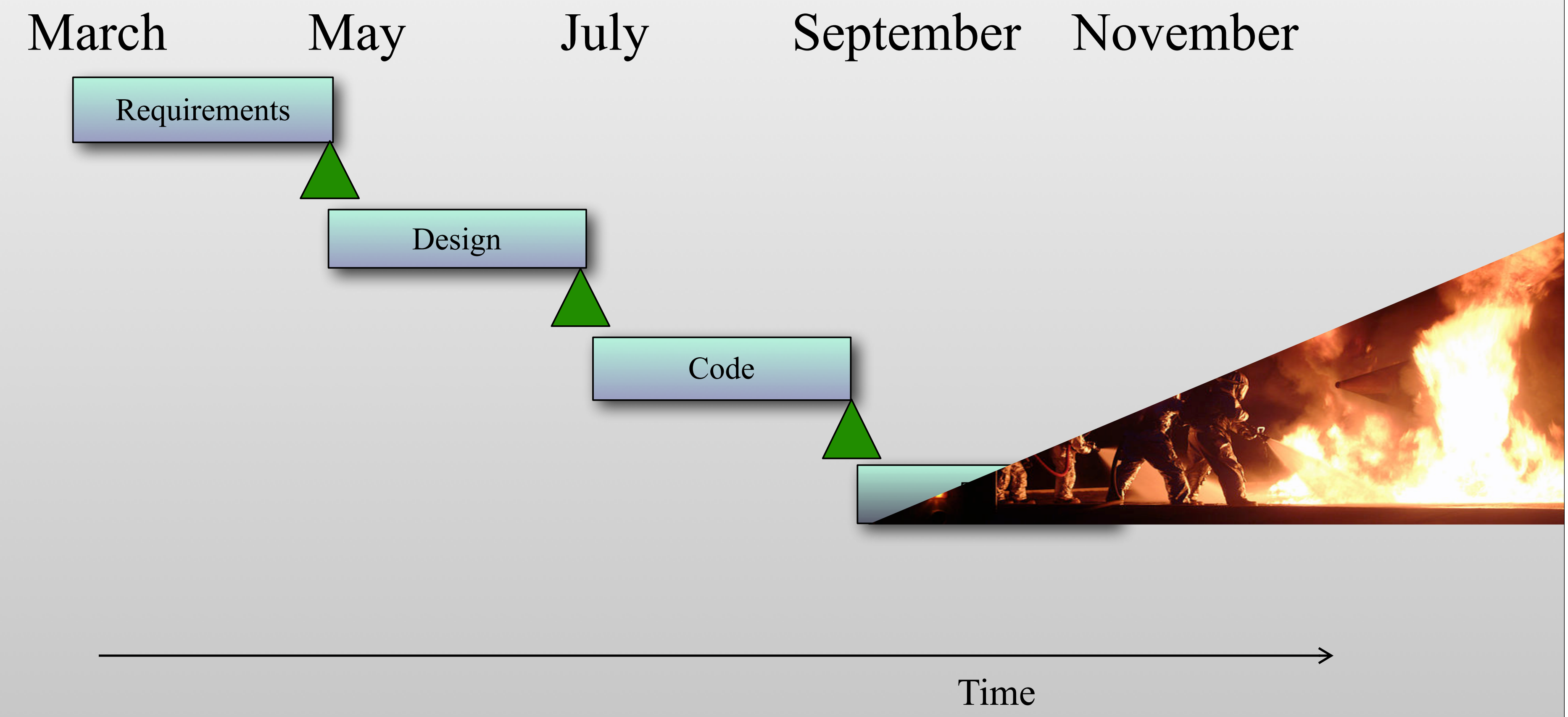


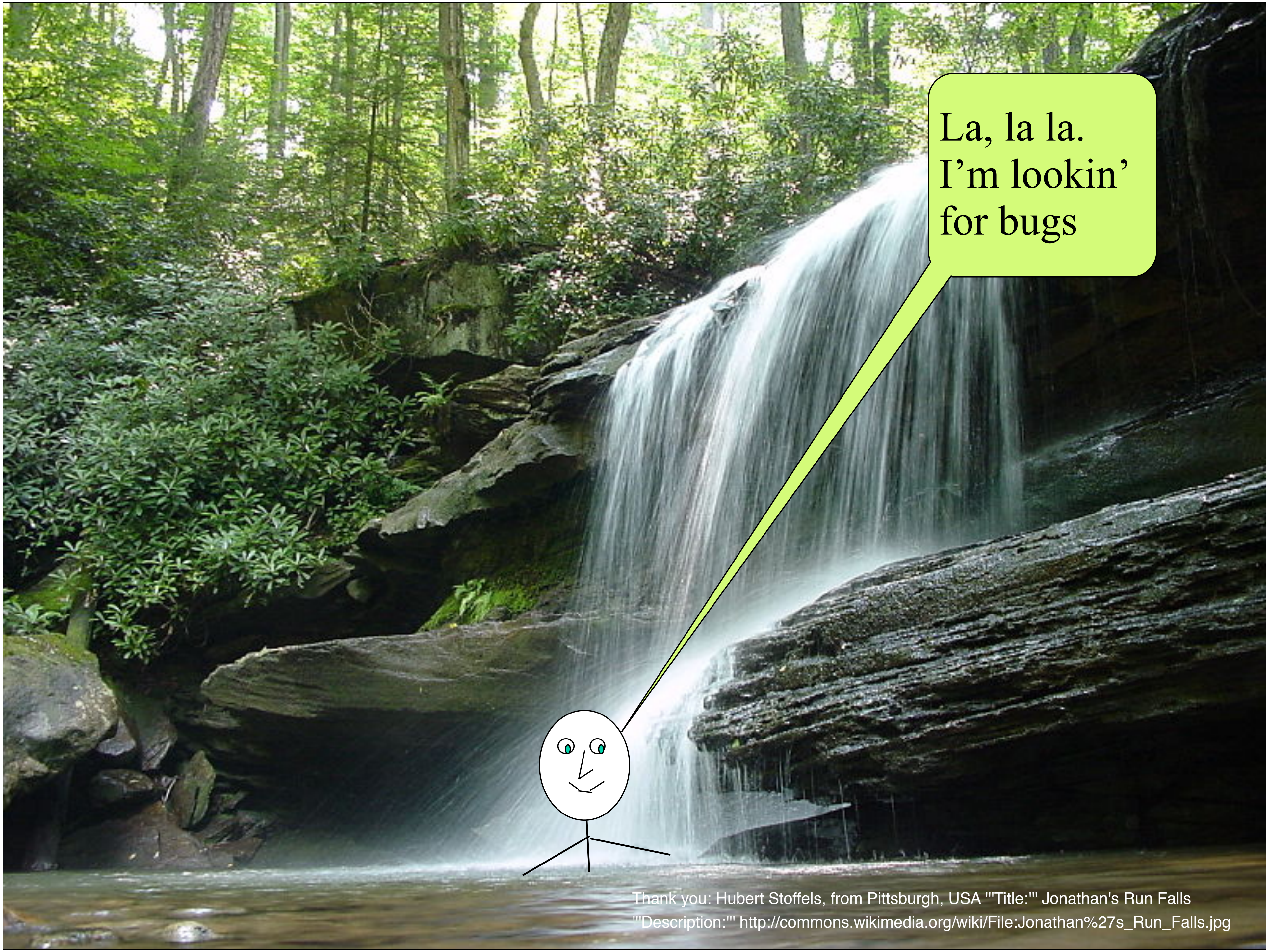
FBI Case-File System - Never Delivered



Denver Baggage System
Almost One Year Late

Why Don’t We Just Test at the End, Save all that





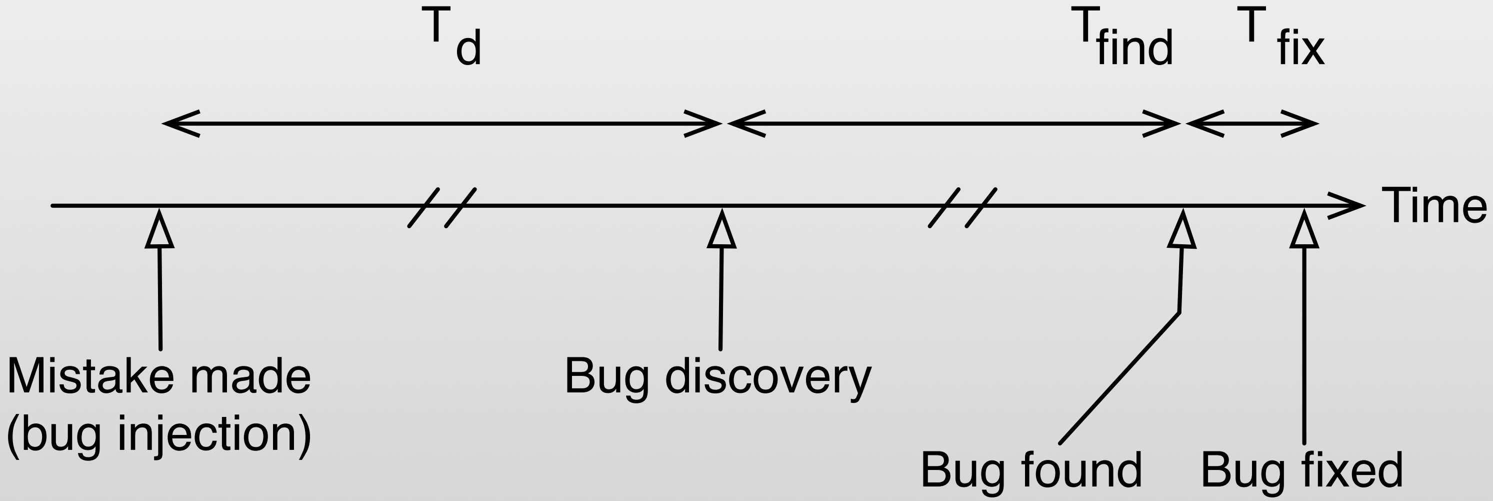
Thank you: Hubert Stoffels, from Pittsburgh, USA '''Title:''' Jonathan's Run Falls
'''Description:''' http://commons.wikimedia.org/wiki/File:Jonathan%27s_Run_Falls.jpg

Thanks to Yorian, Picture of a waterfall nearby Flam, Norway http://commons.wikimedia.org/wiki/File:Waterfall_in_Norway.jpg

nan Ward Photography

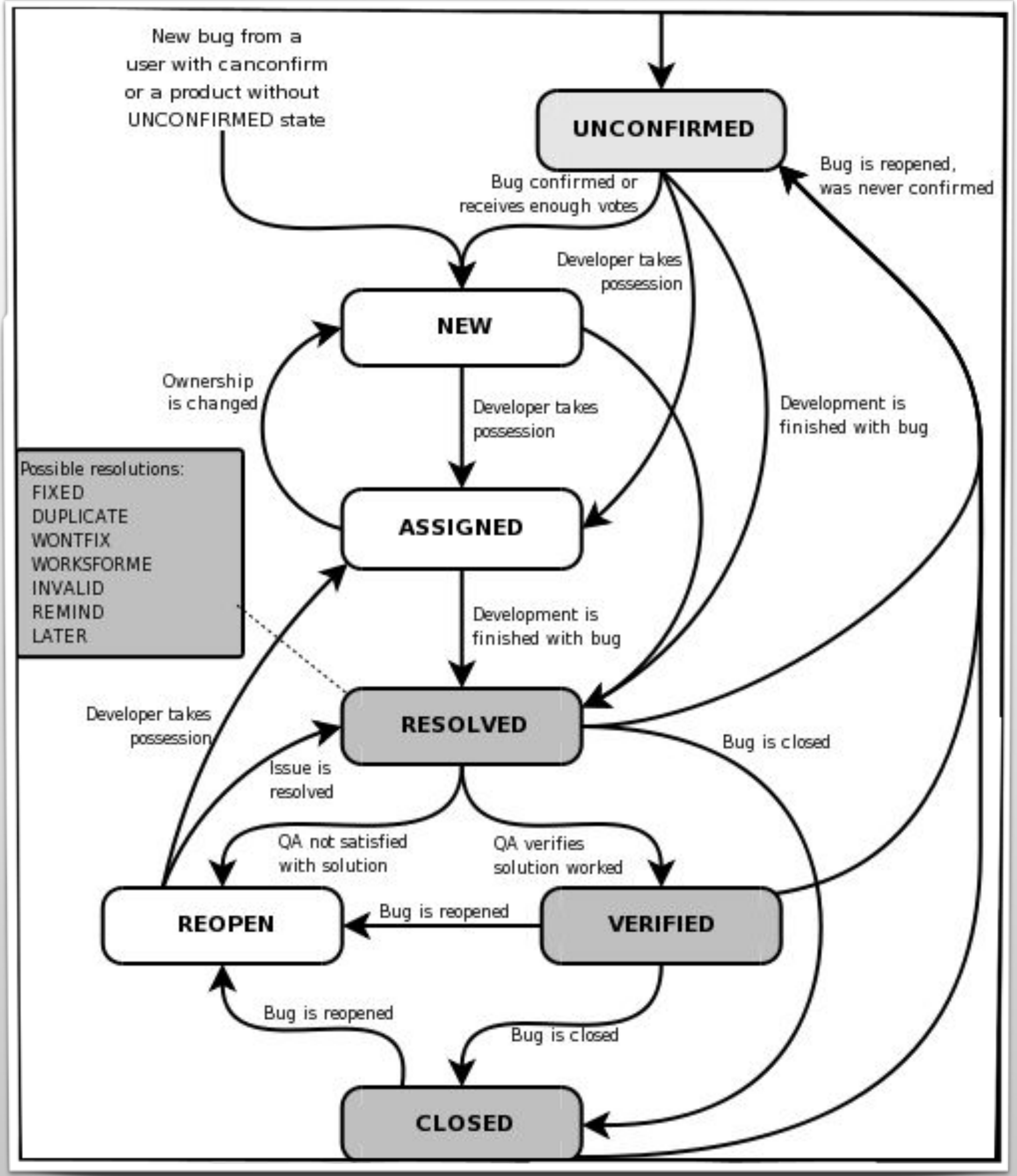
What if test moved upstream? Could we get features to flow?

The Physics of Debug Later Programming (DLP)



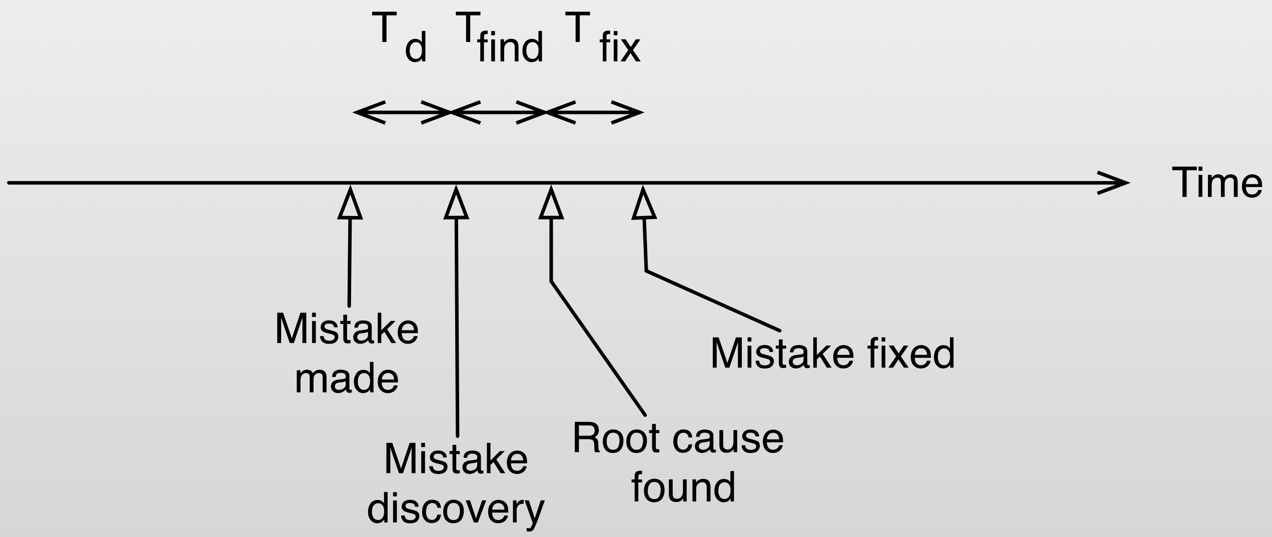
- As T_d increases, T_{find} increases dramatically
- T_{fix} is usually short, but can increase with T_d

A Bug's Life


New bug from a user with canconfirm or a product without UNCONFIRMED state
UNCONFIRMED
Bug confirmed or receives enough votes
Bug is reopened, was never confirmed
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
Developer takes possession
RESOLVED
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/

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.

What if test moved upstream from a reactive role to a proactive role defining detailed requirements in executable use cases?

Keep the Cost of Re-Test Low

- 25% of defects are introduced while changing existing code
- Automated tests keep that cost low
- Test are run with every change

Use Cases, Product Stories, and Story Tests

There are Different kinds of Automated Tests

- Unit Test
  - Feedback to the developer that the code does what is expected
  - Written using a Unit Test Harness (e.g. unity, CppUTest, …)

- Story Tests - Executable use cases - Our focus
  - Feedback to the *Customer* that the code meets the requirements
  - Used at many levels
    - Component
    - Groups of integrated components
    - System
  - Written in a domain specific language (e.g. FitNesse) [FITNESSE]

- Load Tests

Use-Case Template

- Name:
- Goal:
- Preconditions:
- Success End Condition:
- Failed End Condition:
- Primary Actor:
- Trigger:
- MAIN SUCCESS SCENARIO
- EXTENSIONS

Source Alistair Cockburn

http://alistair.cockburn.us/Basic+use+case+template

Use-Case Example

Information	Description
Name	Schedule light control
Goal	Allow system users to schedule lights to turn on, off, or dim
Preconditions	System has controllable lights attached
Success End Condition	The scheduled light has been controlled at the scheduled time
Failed End Condition	The scheduled light has not been controlled at the scheduled time
Primary Actor	Home owner
Trigger	Scheduled time is reached
Main Success Scenario	1.The home owner schedules a light to turn on at a specific time on a specific day 2.The scheduler wakes up at the right time of the right day 3.The light scheduled for this minute is turned on

Use-Case Example Continued

Information	Description
Extensions/Variations	1a. Homeowner can schedule the light to turn on 1b. Homeowner can schedule the light to turn off 1c. Homeowner can schedule the light to set to a dim level 1d. Homeowner can specify weekend schedule 1e. Homeowner can specify weekday schedule 2a - Scheduler does nothing when it wakes up and there are no scheduled controls. 3a - Light is turned on when on is scheduled 3b - Light is turned off when off is scheduled 3c - Light is set to a specified level when dim is scheduled

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

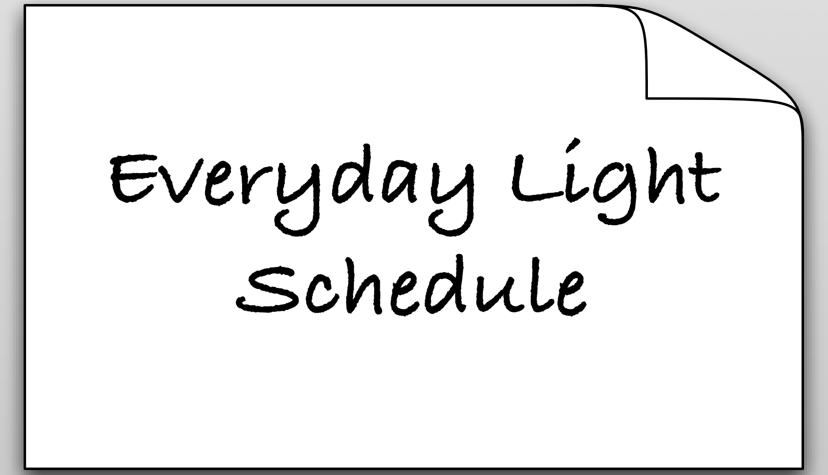
Chinese Holiday Light Schedule

Everyday Light Schedule

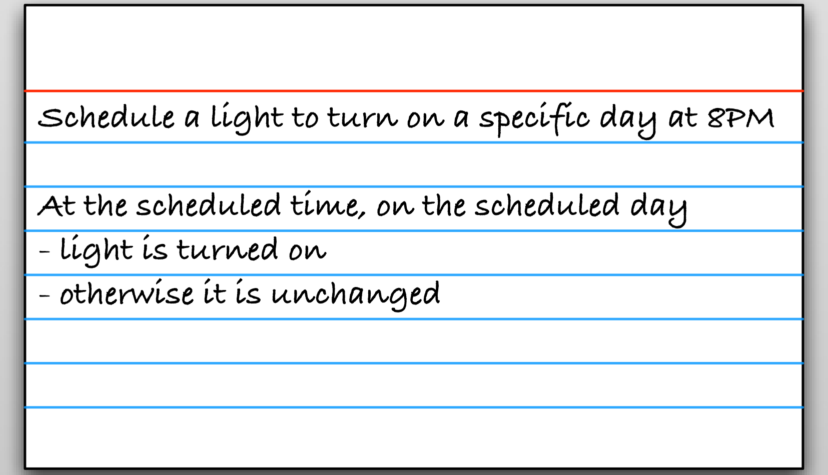
Everyday-but Light Schedule

Back of the Card

- Optionally, use the back of the card for details, and notes about acceptance criteria.
- Keeping the specification light until more detail is needed JIT.



Everyday Light Schedule



Schedule a light to turn on a specific day at 8PM

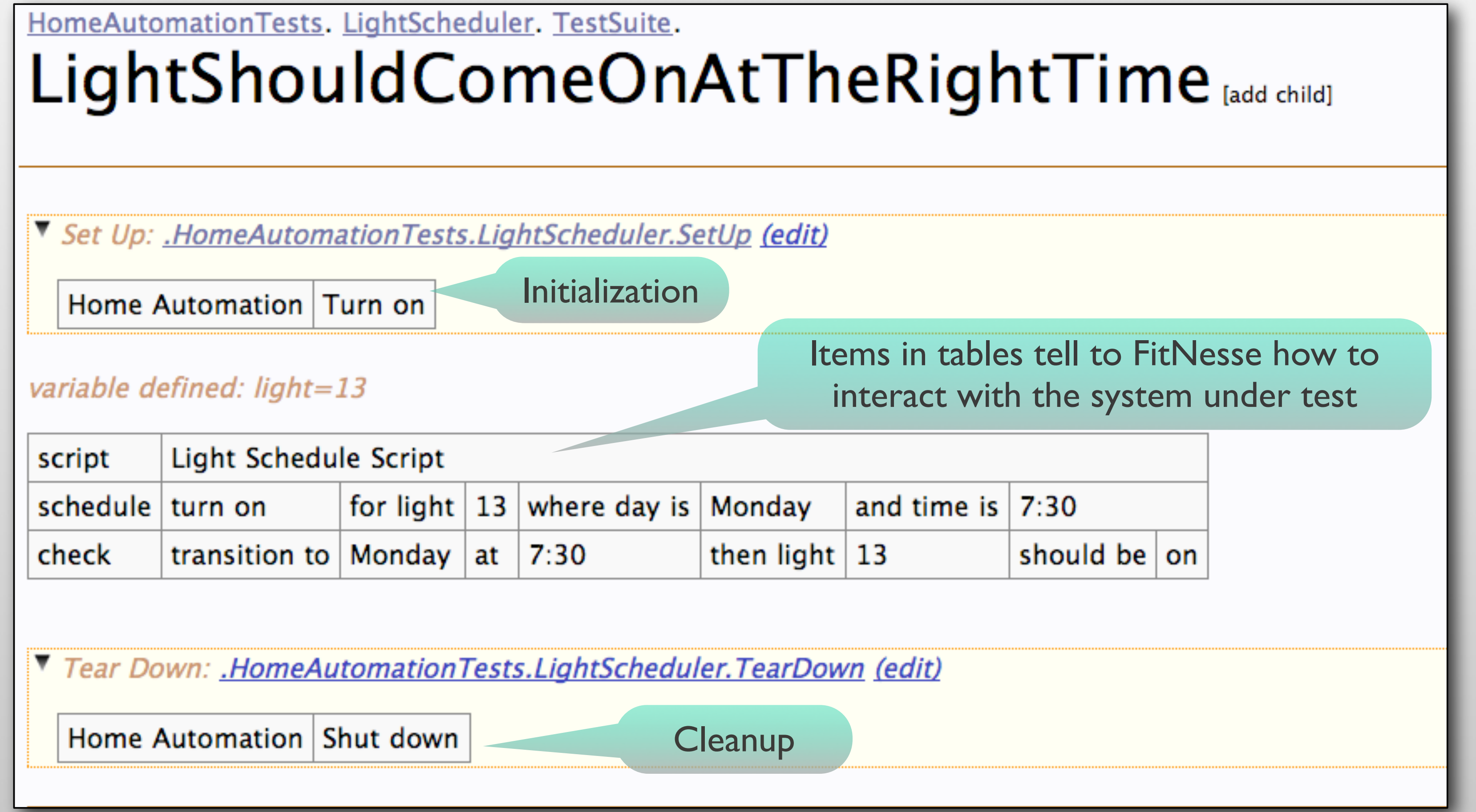
At the scheduled time, on the scheduled day

- light is turned on
- otherwise it is unchanged

Test Define Done

- Acceptance tests define done
- Customer, QA and development agree on how a story will be tested
- The story is considered *done* when it passes its acceptance tests

Executable Use-Case with FitNesse



HomeAutomationTests. LightScheduler. TestSuite.

LightShouldComeOnAtTheRightTime [add child]

▼ *Set Up: .HomeAutomationTests.LightScheduler.SetUp (edit)*

Home Automation	Turn on

Initialization

variable defined: light=13

script	Light Schedule Script								
schedule	turn on	for light	13	where day is	Monday	and time is	7:30		
check	transition to	Monday	at	7:30	then light	13	should be	on	

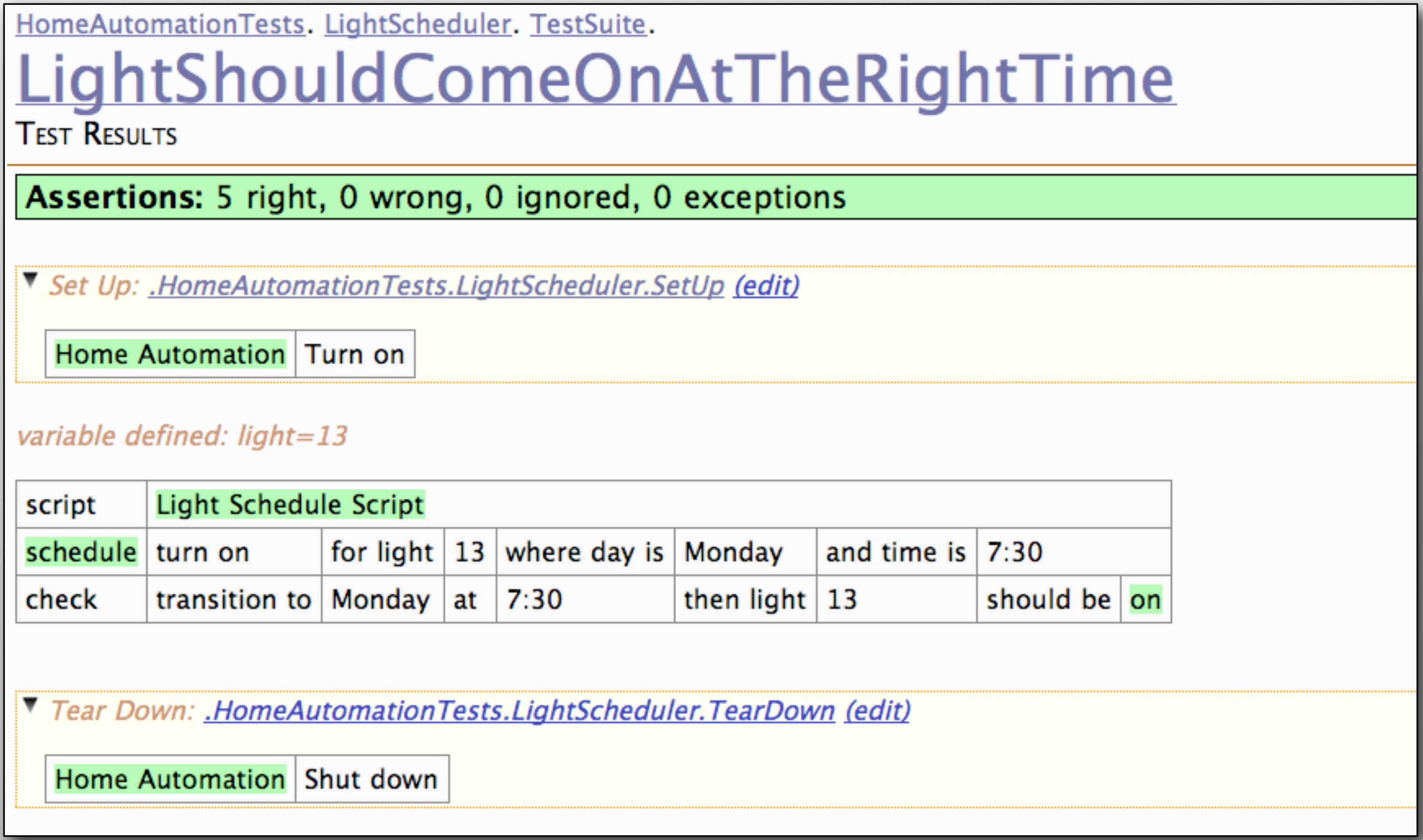
Items in tables tell to FitNesse how to interact with the system under test

▼ *Tear Down: .HomeAutomationTests.LightScheduler.TearDown (edit)*

Home Automation	Shut down

Cleanup

Passing Test



HomeAutomationTests. LightScheduler. TestSuite.

LightShouldComeOnAtTheRightTime

Test Results

Assertions: 5 right, 0 wrong, 0 ignored, 0 exceptions

▼ *Set Up: .HomeAutomationTests.LightScheduler.SetUp (edit)*

Home Automation	Turn on

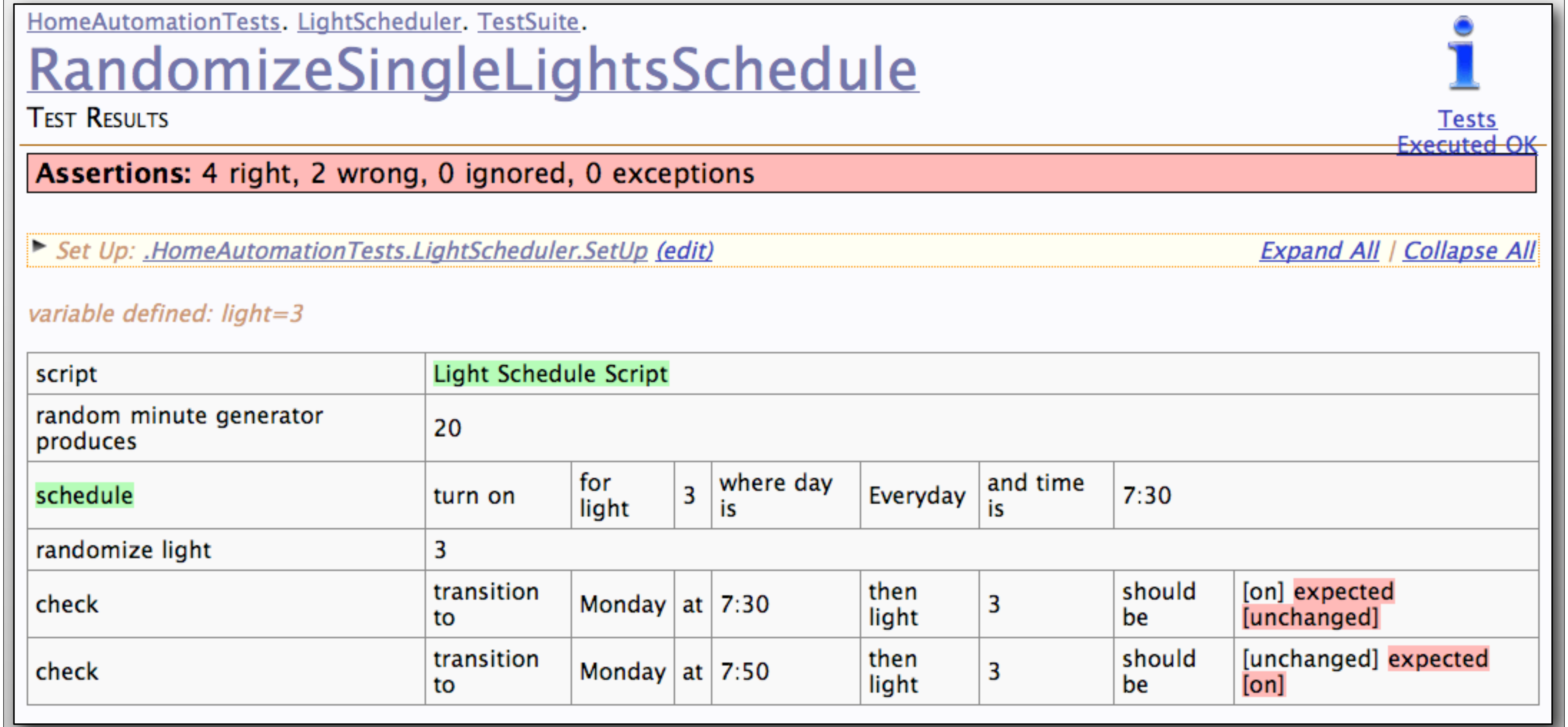
variable defined: light=13

script	Light Schedule Script							
schedule	turn on	for light	13	where day is	Monday	and time is	7:30	
check	transition to	Monday	at	7:30	then light	13	should be	on

▼ *Tear Down: .HomeAutomationTests.LightScheduler.TearDown (edit)*

Home Automation	Shut down

Failing Test



HomeAutomationTests. LightScheduler. TestSuite.

RandomizeSingleLightsSchedule

TEST RESULTS

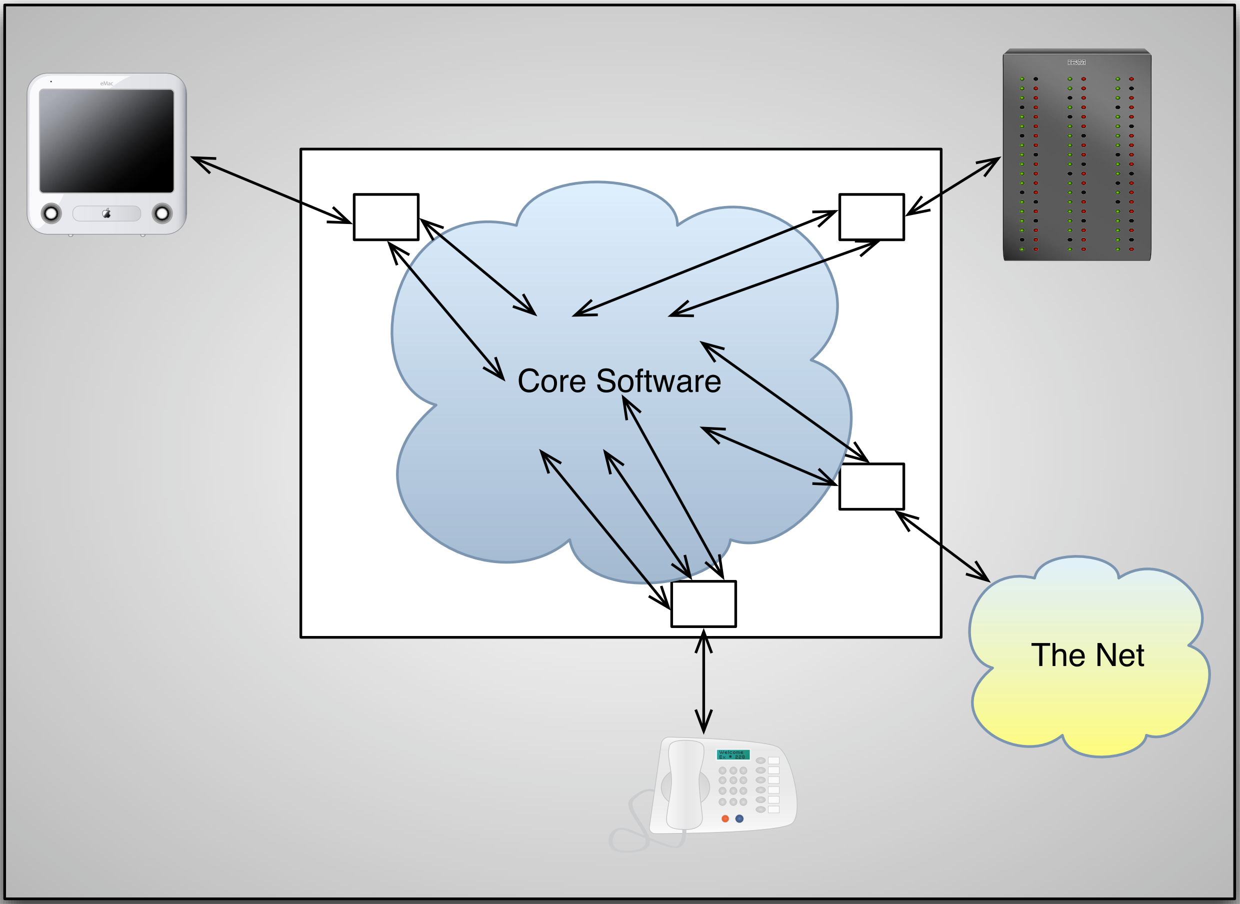
Assertions: 4 right, 2 wrong, 0 ignored, 0 exceptions

▶ *Set Up: .HomeAutomationTests.LightScheduler.SetUp (edit)*

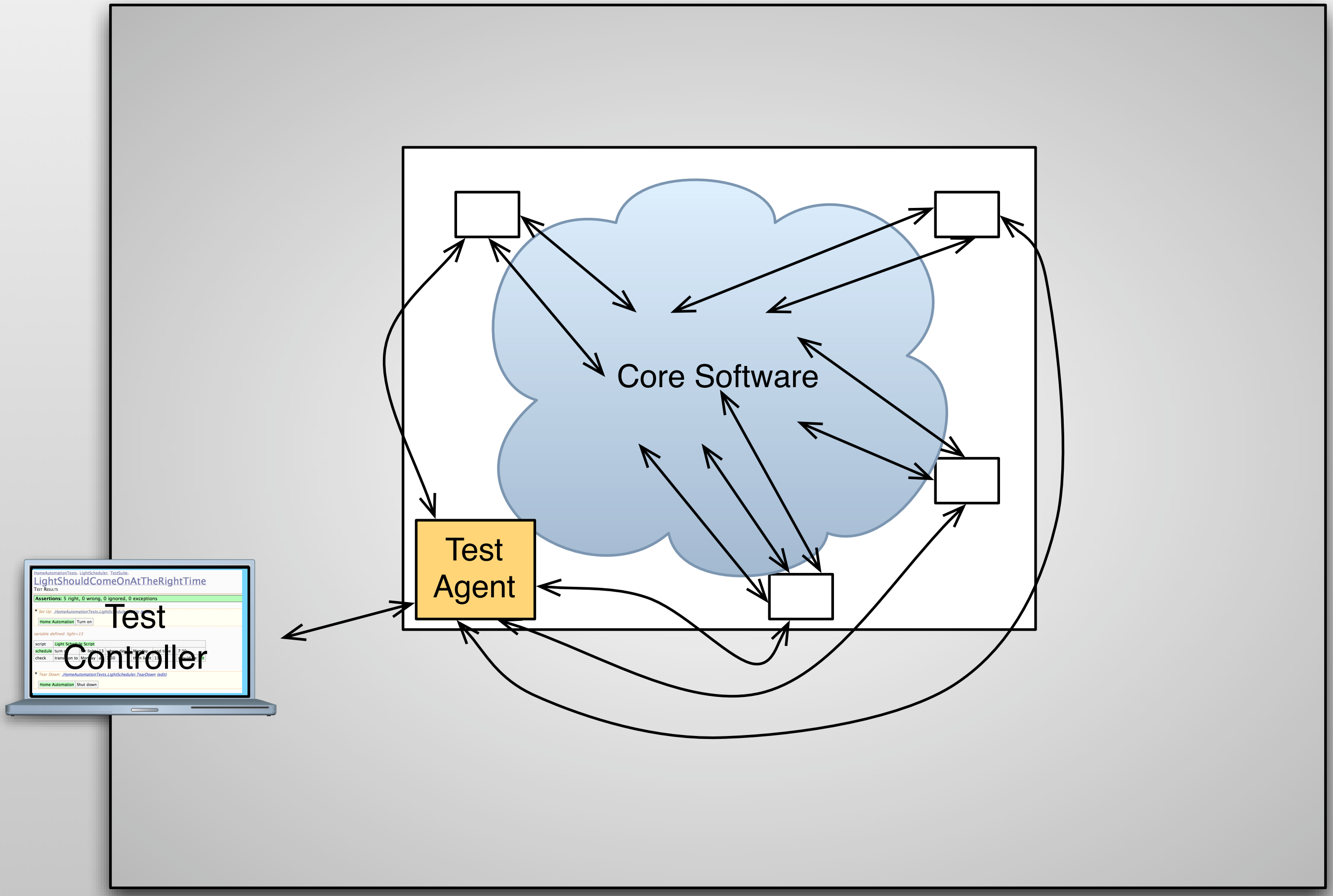
variable defined: light=3

script	Light Schedule Script							
random minute generator produces	20							
schedule	turn on	for light	3	where day is	Everyday	and time is	7:30	
randomize light	3							
check	transition to	Monday	at	7:30	then light	3	should be	[on] expected [unchanged]
check	transition to	Monday	at	7:50	then light	3	should be	[unchanged] expected [on]

Unmanaged Hardware Dependency Lead to Manual Testing



Managed Dependencies Enables Automated Testing



HomeAutomationTests. LightScheduler. TestSuite.

LightShouldComeOnAtTheRightTime

Test Results

Assertions: 5 right, 0 wrong, 0 ignored, 0 exceptions

▼ Set Up: .HomeAutomationTests.LightScheduler.SetUp (edit)

Home Automation	Turn on

variable defined: light=13

script	Light Schedule Script								
schedule	turn on	for light	13	where day	Monday	and time	7:30		
check	transition to	Monday	at	7:30	then light	13	should be	on	

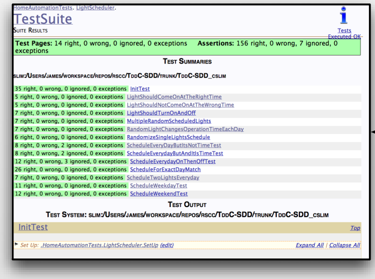
▼ Tear Down: .HomeAutomationTests.LightScheduler.TearDown (edit)

Home Automation	Shut down

Where do the Tests Run?

- Run tests on a host development machine
- Design the embedded application and tests so that many can run on or off the target environment
- Design Guidelines
  - –Isolate hardware dependencies
  - –Isolate OS dependencies
  - –Try to separate threading from core application

FitNesse Test Architecture



HomeAutomationTests. LightScheduler.

TestSuite

Suite Results

Tests Executed OK

Test Pages: 14 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 156 right, 0 wrong, 7 ignored, 0 exceptions

Test Summaries

slim:/Users/james/workspace/repos/rscc/TddC-SDD/trunk/TddC-SDD_cslim

35 right, 0 wrong, 0 ignored, 0 exceptions	InitTest
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldComeOnAtTheRightTime
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldNotComeOnAtTheWrongTime
7 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldTurnOnAndOff
7 right, 0 wrong, 0 ignored, 0 exceptions	MultipleRandomScheduledLights
7 right, 0 wrong, 0 ignored, 0 exceptions	RandomLightChangesOperationTimeEachDay
6 right, 0 wrong, 0 ignored, 0 exceptions	RandomizeSingleLightsSchedule
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEveryDayButItsNotTimeTest
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEverydayButAndItIsTimeTest
12 right, 0 wrong, 3 ignored, 0 exceptions	ScheduleEverydayOnThenOffTest
26 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleForExactDayMatch
7 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleTwoLightsEveryday
11 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekdayTest
12 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekendTest

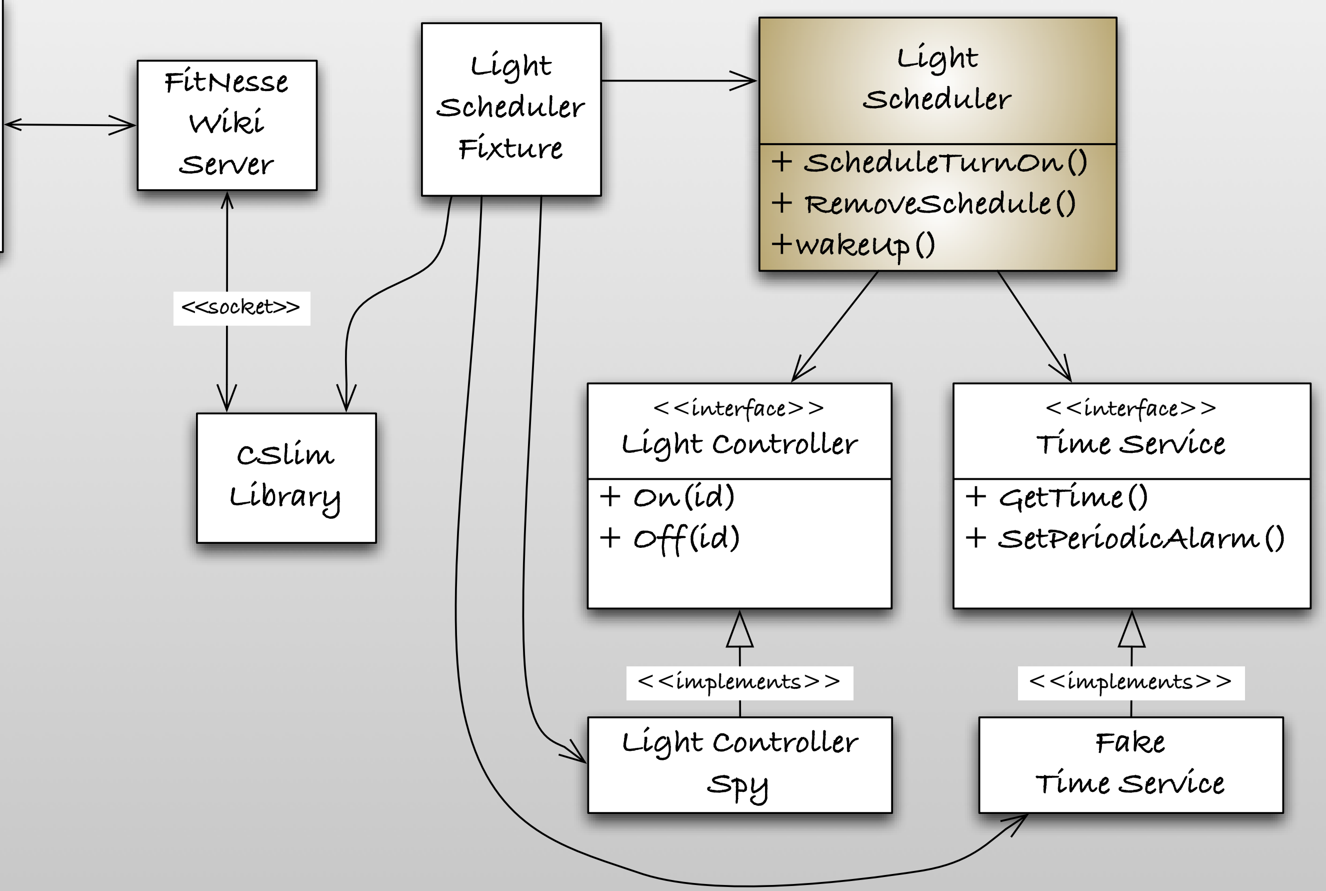
Test Output

Test System: slim:/Users/james/workspace/repos/rscc/TddC-SDD/trunk/TddC-SDD_cslim

InitTest Top

▸ Set Up: .HomeAutomationTests.LightScheduler.SetUp (edit) Expand All | Collapse All

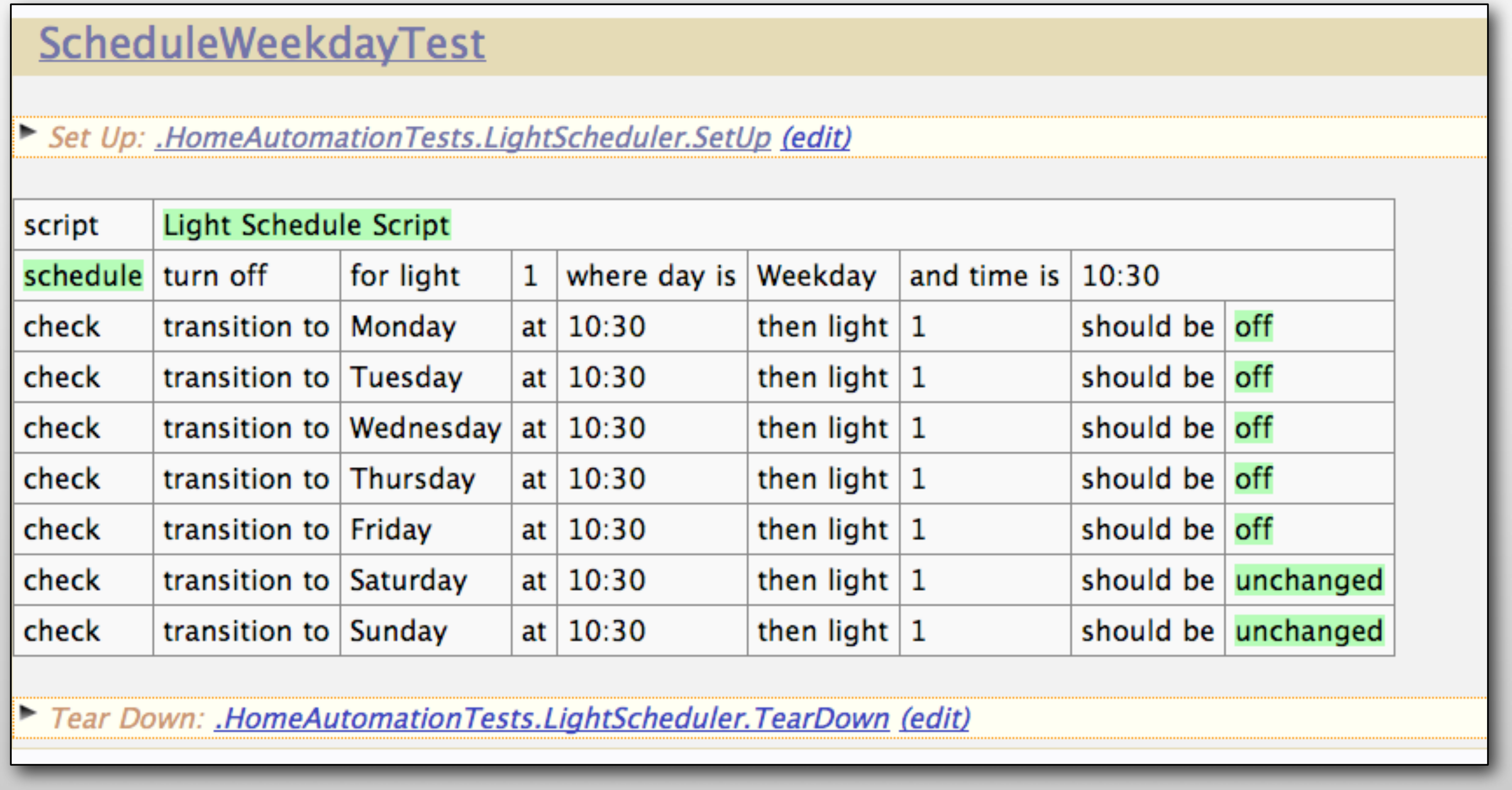
FitNesse Test Pages



FitNesse and CSlim

- FitNesse (open source)
  - –is a specialized wiki server that interprets test pages
  - –interacts with a SLIM server and its fixtures
- CSlim (open source)
  - –is the C implementation of SLIM
  - –it supports the development of fixtures
  - –it manages the FitNesse/SLIM message traffic
- Fixtures (you write these)
  - –are small bits of code that convert SLIM

Imagine the Equivalent Manual Test Procedure



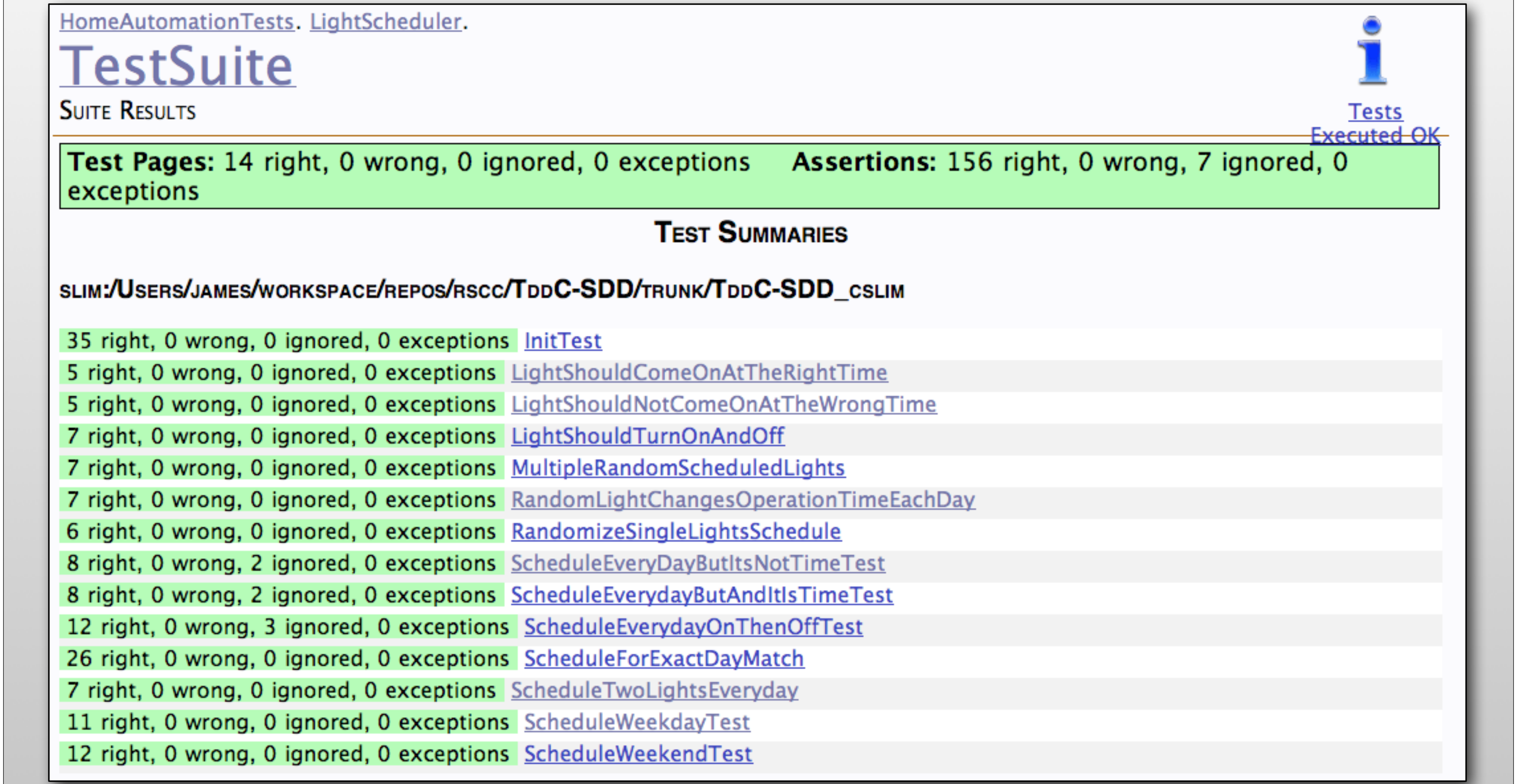
ScheduleWeekdayTest

▶ *Set Up: .HomeAutomationTests.LightScheduler.SetUp (edit)*

script	Light Schedule Script							
schedule	turn off	for light	1	where day is	Weekday	and time is	10:30	
check	transition to	Monday	at	10:30	then light	1	should be	off
check	transition to	Tuesday	at	10:30	then light	1	should be	off
check	transition to	Wednesday	at	10:30	then light	1	should be	off
check	transition to	Thursday	at	10:30	then light	1	should be	off
check	transition to	Friday	at	10:30	then light	1	should be	off
check	transition to	Saturday	at	10:30	then light	1	should be	unchanged
check	transition to	Sunday	at	10:30	then light	1	should be	unchanged

▶ *Tear Down: .HomeAutomationTests.LightScheduler.TearDown (edit)*

A Suite of Tests While all Test Pass



HomeAutomationTests. LightScheduler.

TestSuite

SUITE RESULTS

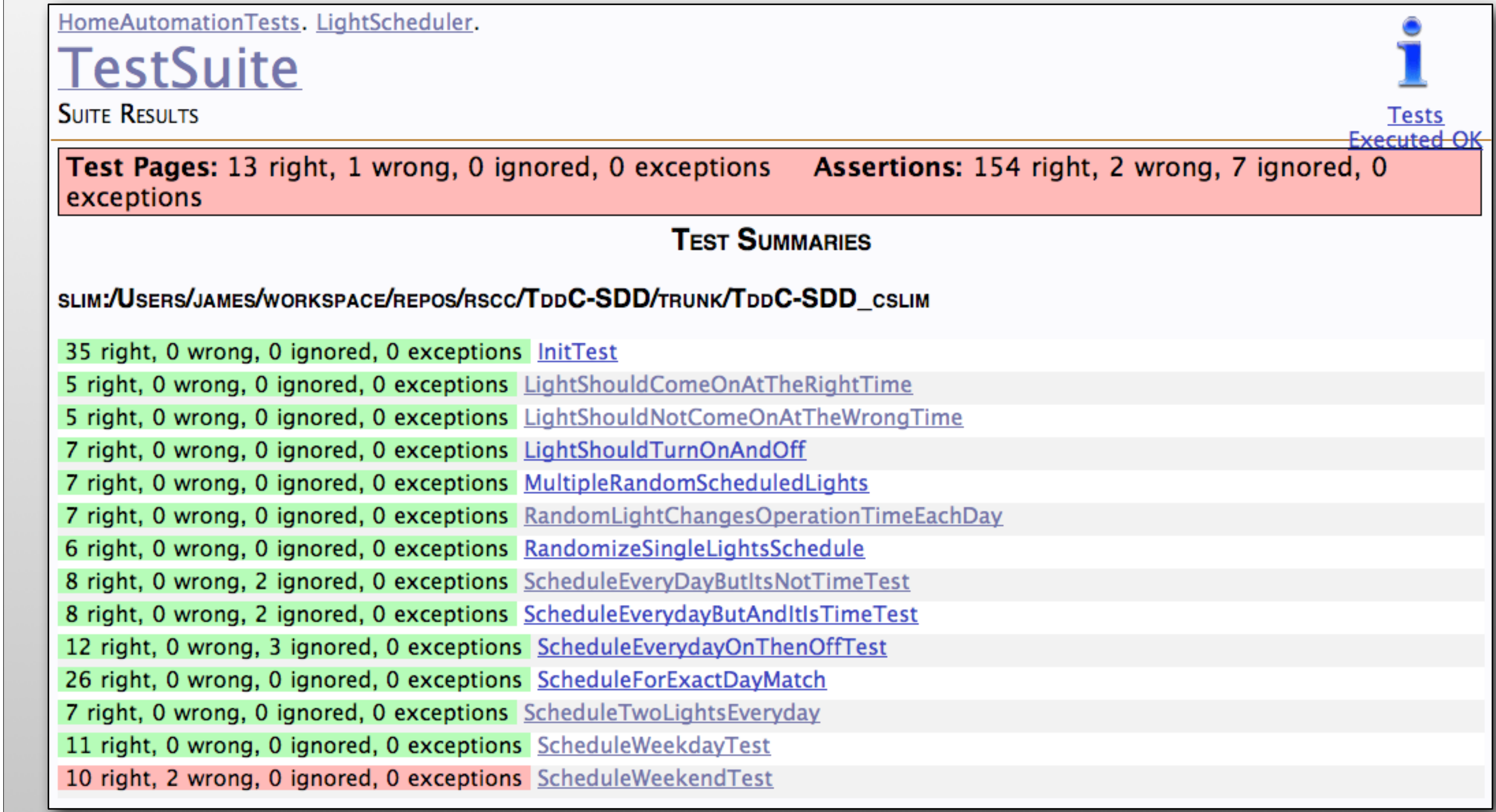
Test Pages: 14 right, 0 wrong, 0 ignored, 0 exceptions **Assertions:** 156 right, 0 wrong, 7 ignored, 0 exceptions

TEST SUMMARIES

SLIM:/USERS/JAMES/WORKSPACE/REPOS/RSCC/TDDC-SDD/TRUNK/TDDC-SDD_CSLIM

35 right, 0 wrong, 0 ignored, 0 exceptions	InitTest
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldComeOnAtTheRightTime
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldNotComeOnAtTheWrongTime
7 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldTurnOnAndOff
7 right, 0 wrong, 0 ignored, 0 exceptions	MultipleRandomScheduledLights
7 right, 0 wrong, 0 ignored, 0 exceptions	RandomLightChangesOperationTimeEachDay
6 right, 0 wrong, 0 ignored, 0 exceptions	RandomizeSingleLightsSchedule
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEveryDayButItsNotTimeTest
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEverydayButAndItIsTimeTest
12 right, 0 wrong, 3 ignored, 0 exceptions	ScheduleEverydayOnThenOffTest
26 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleForExactDayMatch
7 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleTwoLightsEveryday
11 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekdayTest
12 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekendTest

When the Story Test is Ready Before the Development



HomeAutomationTests. LightScheduler.

TestSuite

Suite Results

Tests Executed OK

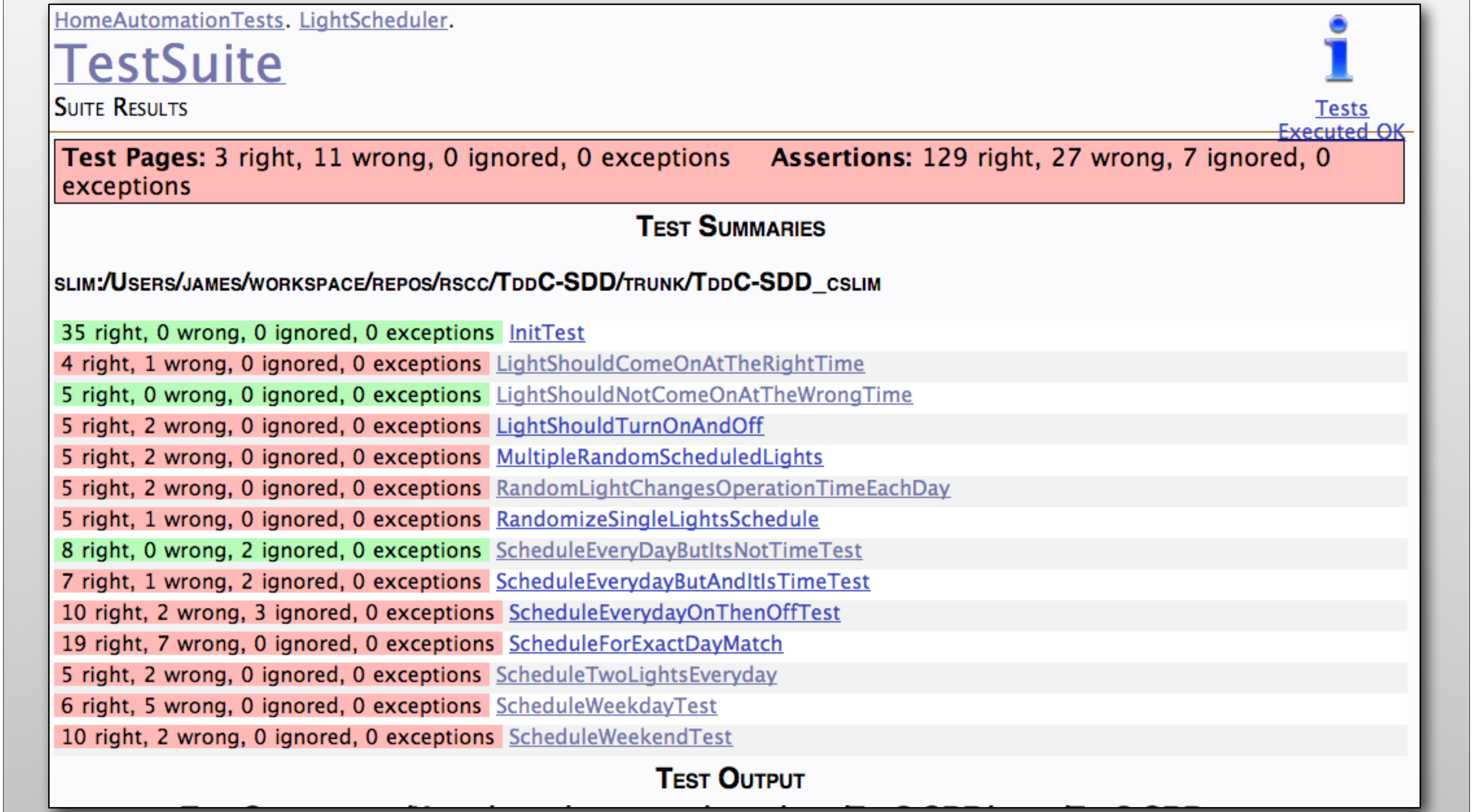
Test Pages: 13 right, 1 wrong, 0 ignored, 0 exceptions **Assertions:** 154 right, 2 wrong, 7 ignored, 0 exceptions

Test Summaries

slim:/Users/james/workspace/repos/rscc/TddC-SDD/trunk/TddC-SDD_cslim

35 right, 0 wrong, 0 ignored, 0 exceptions	InitTest
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldComeOnAtTheRightTime
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldNotComeOnAtTheWrongTime
7 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldTurnOnAndOff
7 right, 0 wrong, 0 ignored, 0 exceptions	MultipleRandomScheduledLights
7 right, 0 wrong, 0 ignored, 0 exceptions	RandomLightChangesOperationTimeEachDay
6 right, 0 wrong, 0 ignored, 0 exceptions	RandomizeSingleLightsSchedule
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEveryDayButItsNotTimeTest
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEverydayButAndItIsTimeTest
12 right, 0 wrong, 3 ignored, 0 exceptions	ScheduleEverydayOnThenOffTest
26 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleForExactDayMatch
7 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleTwoLightsEveryday
11 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekdayTest
10 right, 2 wrong, 0 ignored, 0 exceptions	ScheduleWeekendTest

When Things are not Working



HomeAutomationTests. LightScheduler.

TestSuite

SUITE RESULTS

Test Pages: 3 right, 11 wrong, 0 ignored, 0 exceptions **Assertions:** 129 right, 27 wrong, 7 ignored, 0 exceptions

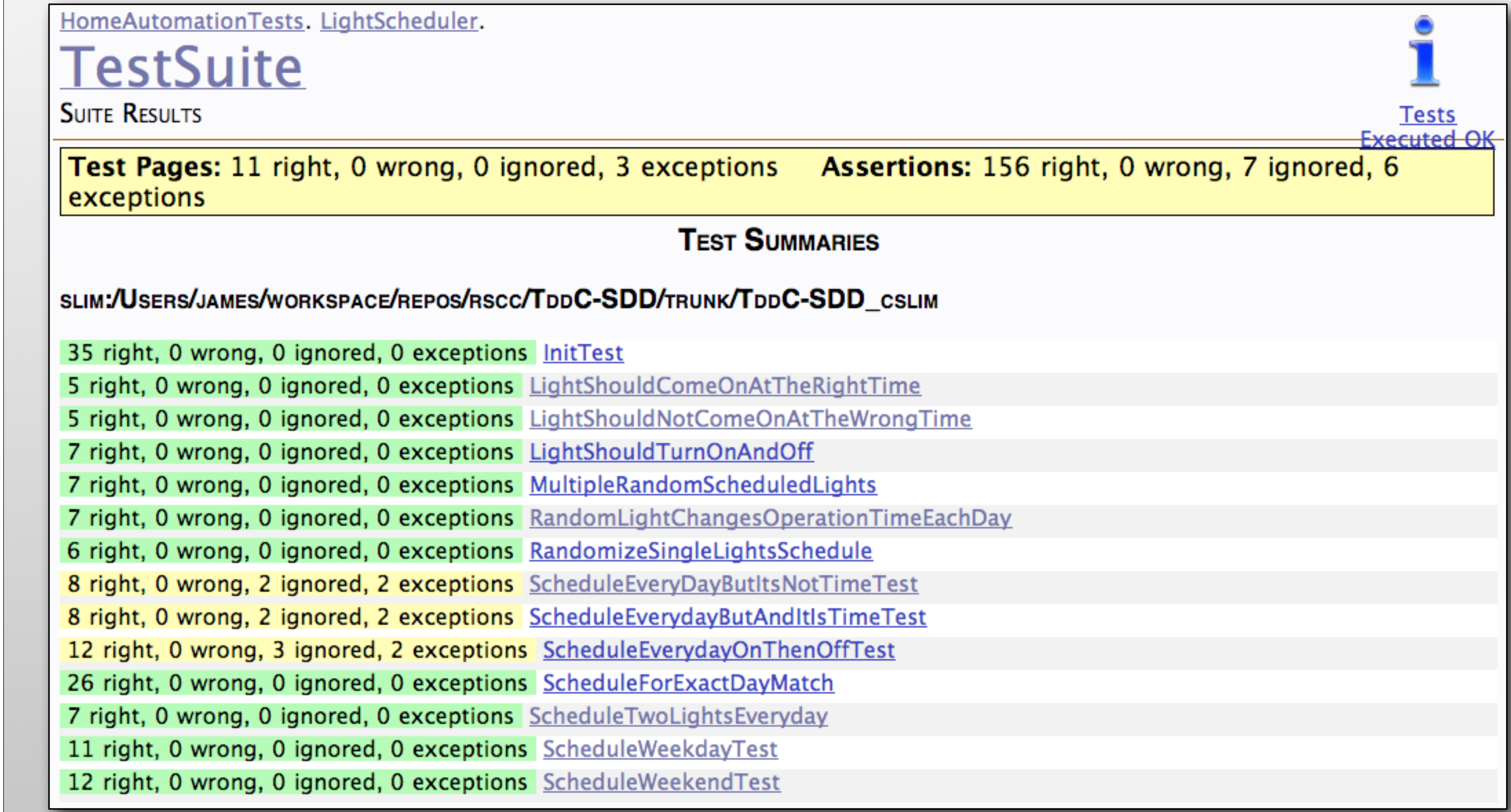
TEST SUMMARIES

SLIM:/USERS/JAMES/WORKSPACE/REPOS/RSCC/TDDC-SDD/TRUNK/TDDC-SDD_CSLIM

35 right, 0 wrong, 0 ignored, 0 exceptions	InitTest
4 right, 1 wrong, 0 ignored, 0 exceptions	LightShouldComeOnAtTheRightTime
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldNotComeOnAtTheWrongTime
5 right, 2 wrong, 0 ignored, 0 exceptions	LightShouldTurnOnAndOff
5 right, 2 wrong, 0 ignored, 0 exceptions	MultipleRandomScheduledLights
5 right, 2 wrong, 0 ignored, 0 exceptions	RandomLightChangesOperationTimeEachDay
5 right, 1 wrong, 0 ignored, 0 exceptions	RandomizeSingleLightsSchedule
8 right, 0 wrong, 2 ignored, 0 exceptions	ScheduleEveryDayButItsNotTimeTest
7 right, 1 wrong, 2 ignored, 0 exceptions	ScheduleEverydayButAndItIsTimeTest
10 right, 2 wrong, 3 ignored, 0 exceptions	ScheduleEverydayOnThenOffTest
19 right, 7 wrong, 0 ignored, 0 exceptions	ScheduleForExactDayMatch
5 right, 2 wrong, 0 ignored, 0 exceptions	ScheduleTwoLightsEveryday
6 right, 5 wrong, 0 ignored, 0 exceptions	ScheduleWeekdayTest
10 right, 2 wrong, 0 ignored, 0 exceptions	ScheduleWeekendTest

TEST OUTPUT

When Fixtures Need to be Written



HomeAutomationTests. LightScheduler.

TestSuite

Suite Results

Tests Executed OK

Test Pages: 11 right, 0 wrong, 0 ignored, 3 exceptions **Assertions:** 156 right, 0 wrong, 7 ignored, 6 exceptions

Test Summaries

slim:/Users/james/workspace/repos/rscc/TddC-SDD/trunk/TddC-SDD_cslim

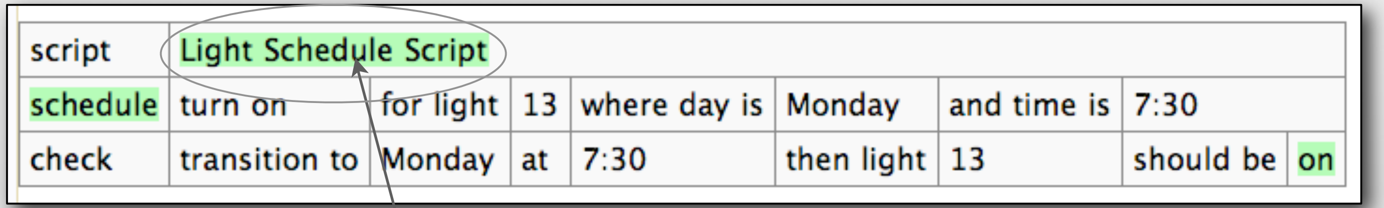
35 right, 0 wrong, 0 ignored, 0 exceptions	InitTest
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldComeOnAtTheRightTime
5 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldNotComeOnAtTheWrongTime
7 right, 0 wrong, 0 ignored, 0 exceptions	LightShouldTurnOnAndOff
7 right, 0 wrong, 0 ignored, 0 exceptions	MultipleRandomScheduledLights
7 right, 0 wrong, 0 ignored, 0 exceptions	RandomLightChangesOperationTimeEachDay
6 right, 0 wrong, 0 ignored, 0 exceptions	RandomizeSingleLightsSchedule
8 right, 0 wrong, 2 ignored, 2 exceptions	ScheduleEveryDayButItsNotTimeTest
8 right, 0 wrong, 2 ignored, 2 exceptions	ScheduleEverydayButAndItIsTimeTest
12 right, 0 wrong, 3 ignored, 2 exceptions	ScheduleEverydayOnThenOffTest
26 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleForExactDayMatch
7 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleTwoLightsEveryday
11 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekdayTest
12 right, 0 wrong, 0 ignored, 0 exceptions	ScheduleWeekendTest

Provide Fakes when Predictable Inputs are Needed

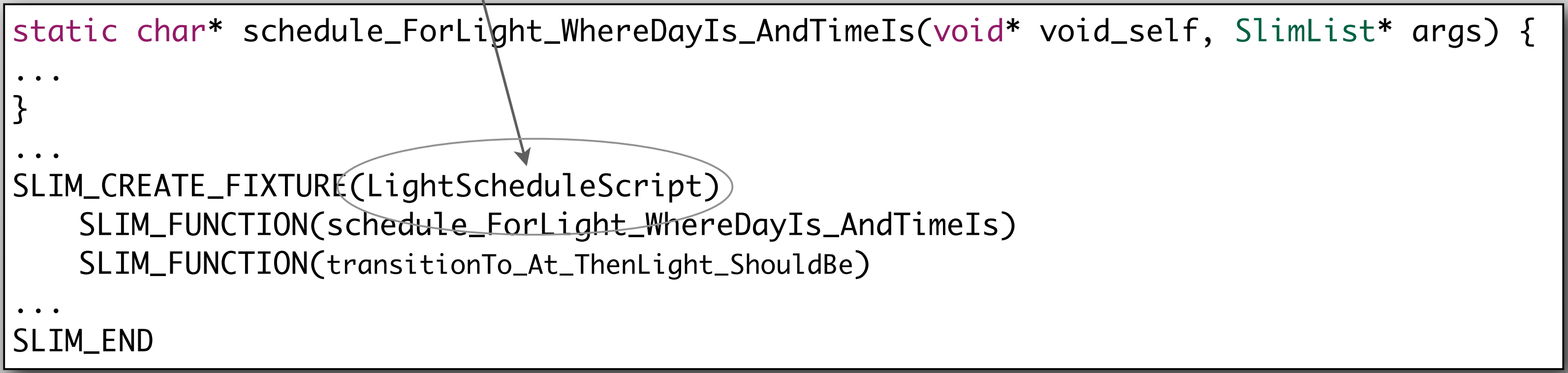
- Use fakes when the system under test cannot be fully tested with the real collaborators.
- Examples:
  - –When manual verification is needed (Printed output, LEDs)
  - –When the results change (Time, random events)
  - –When failures need to be simulated (Network down)
  - –When hardware is involved (LEDs, USB, Sensors, Motors, IO pins, Flash...)
  - –Operating system calls (RTOS)

The Code Behind

- The script table’s name maps to a SLIM fixture name



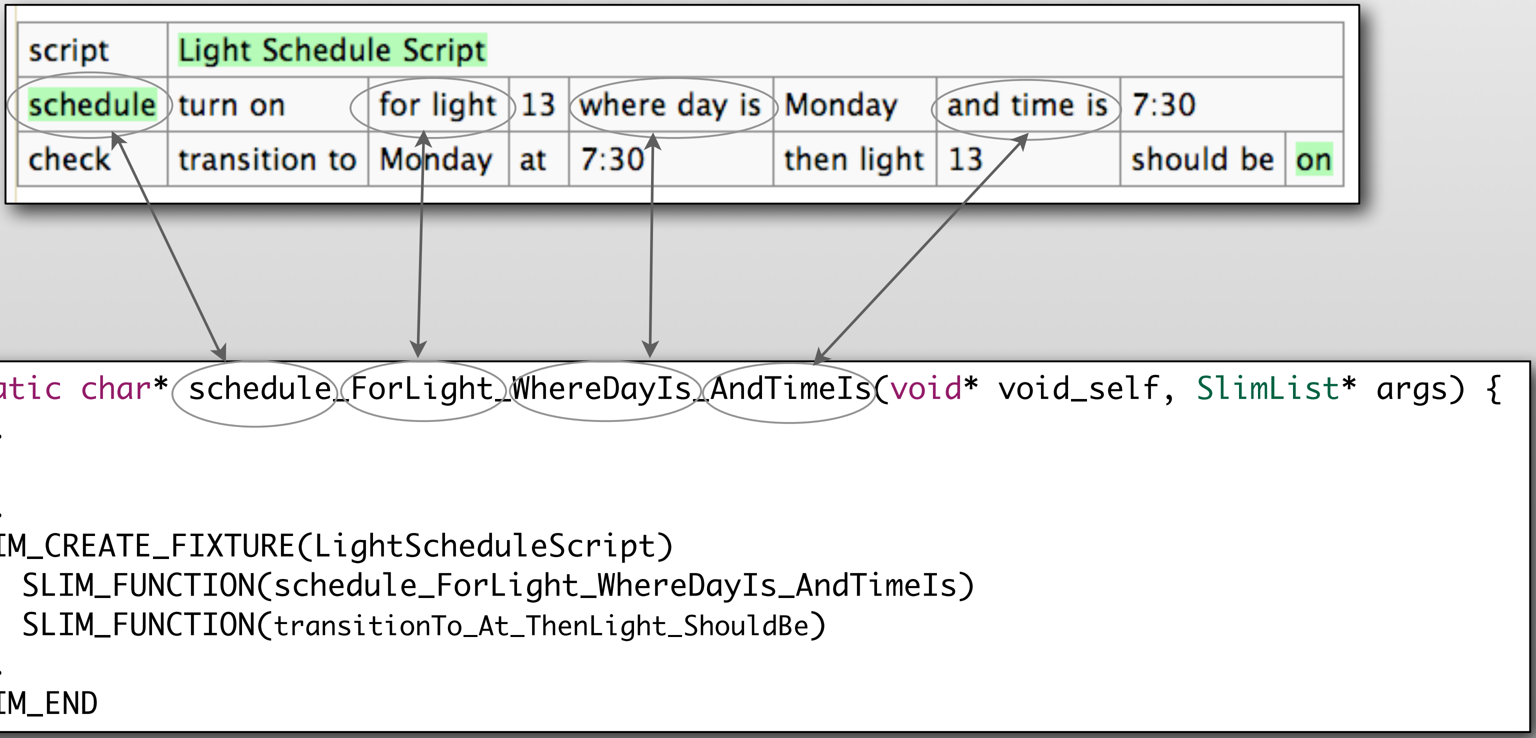
script	Light Schedule Script							
schedule	turn on	for light	13	where day is	Monday	and time is	7:30	
check	transition to	Monday	at	7:30	then light	13	should be	on



```
static char* schedule_ForLight_WhereDayIs_AndTimeIs(void* void_self, SlimList* args) {
...
}
...
SLIM_CREATE_FIXTURE(LightScheduleScript)
    SLIM_FUNCTION(schedule_ForLight_WhereDayIs_AndTimeIs)
    SLIM_FUNCTION(transitionTo_At_ThenLight_ShouldBe)
...
SLIM_END
```

The Code Behind

- A row in a script table maps to a function. Every other cell is a parameter



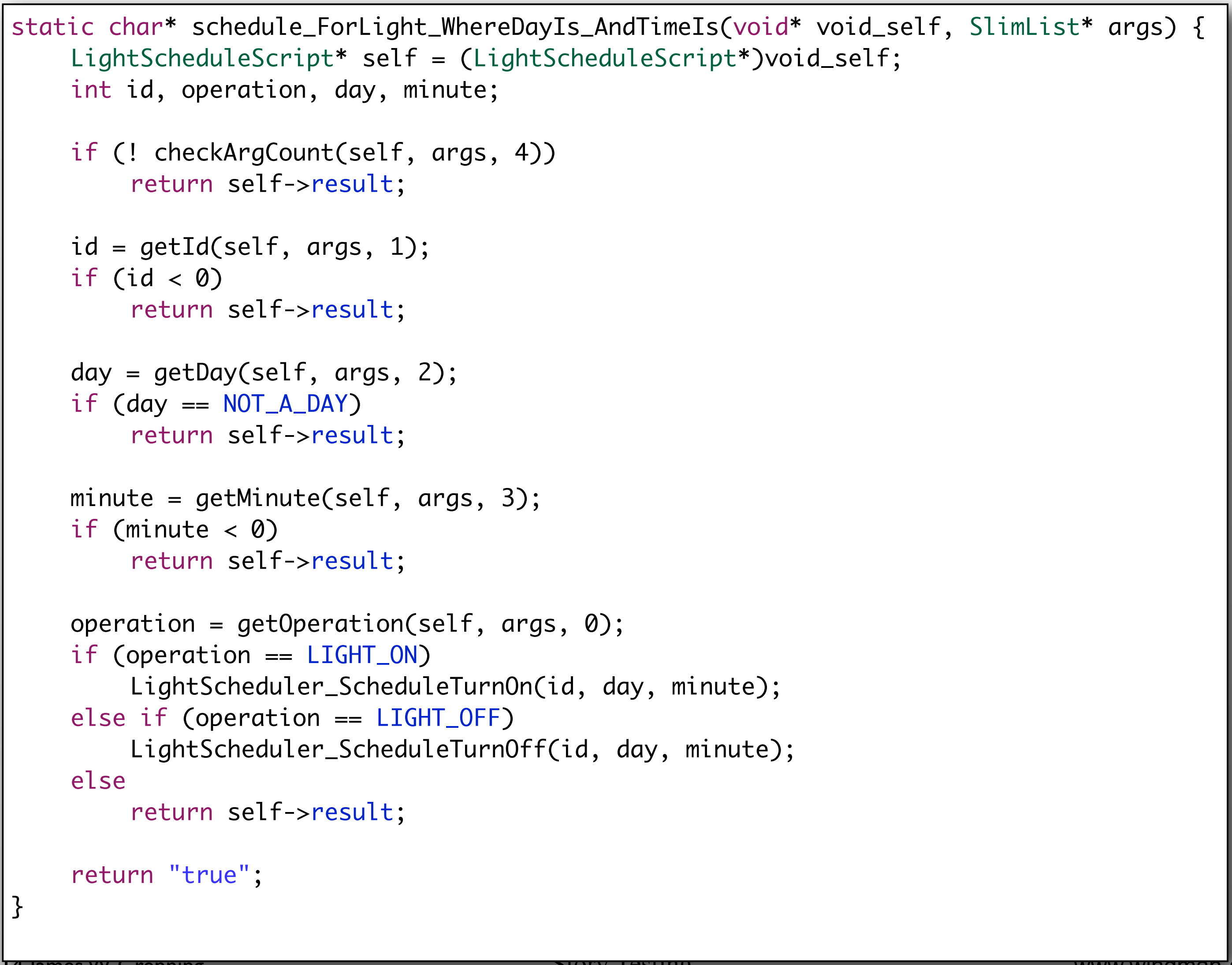
script	Light Schedule Script							
schedule	turn on	for light	13	where day is	Monday	and time is	7:30	
check	transition to	Monday	at	7:30	then light	13	should be	on

```
static char* schedule_ForLight_WhereDayIs_AndTimeIs(void* void_self, SlimList* args) {
...
}
...
SLIM_CREATE_FIXTURE(LightScheduleScript)
    SLIM_FUNCTION(schedule_ForLight_WhereDayIs_AndTimeIs)
    SLIM_FUNCTION(transitionTo_At_ThenLight_ShouldBe)
...
SLIM_END
```

SLIM Fixture Function Responsibility

- Unpack and check parameters
- Call production code
- Interact with stubs, and fakes
- Return results

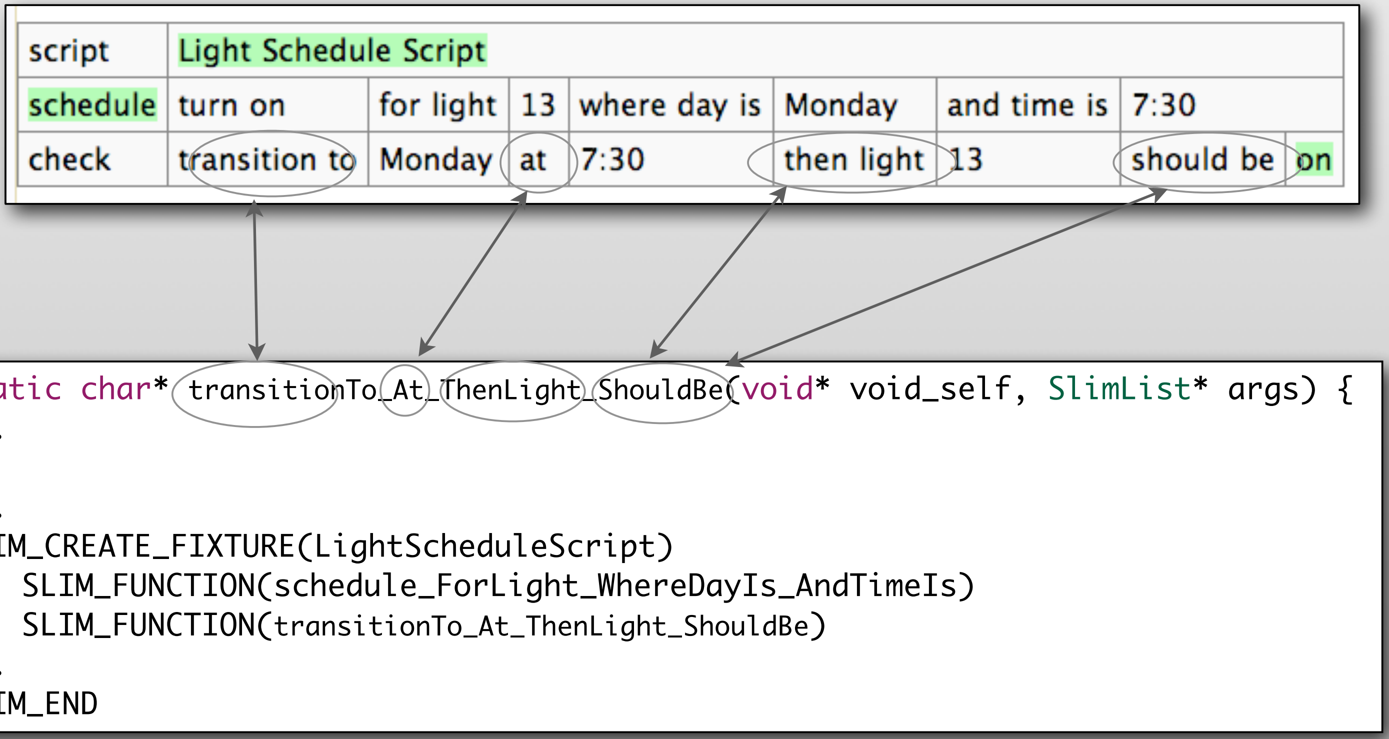
Simulating User Input



```
static char* schedule_ForLight_WhereDayIs_AndTimeIs(void* void_self, SlimList* args) {
    LightScheduleScript* self = (LightScheduleScript*)void_self;
    int id, operation, day, minute;

    if (! checkArgCount(self, args, 4))
        return self->result;

    id = getId(self, args, 1);
    if (id < 0)
        return self->result;

    day = getDay(self, args, 2);
    if (day == NOT_A_DAY)
        return self->result;

    minute = getMinute(self, args, 3);
    if (minute < 0)
        return self->result;

    operation = getOperation(self, args, 0);
    if (operation == LIGHT_ON)
        LightScheduler_ScheduleTurnOn(id, day, minute);
    else if (operation == LIGHT_OFF)
        LightScheduler_ScheduleTurnOff(id, day, minute);
    else
        return self->result;

    return "true";
}
```

The Code Behind

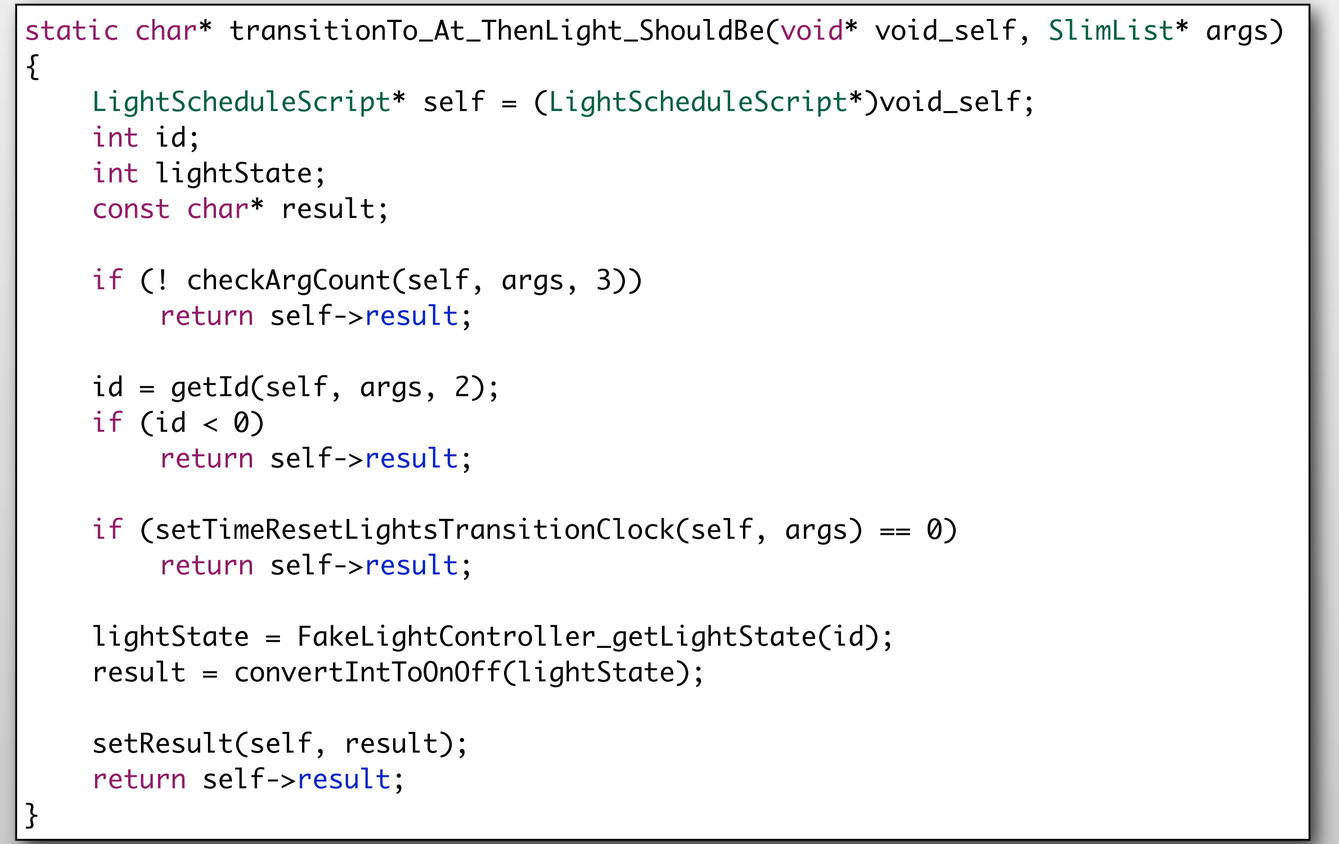
- A *check* row in a script table maps to a function with the last field being the return result.



script	Light Schedule Script							
schedule	turn on	for light	13	where day is	Monday	and time is	7:30	
check	transition to	Monday	at	7:30	then light	13	should be	on

```
static char* transitionTo_At_ThenLight_ShouldBe(void* void_self, SlimList* args) {
...
}
...
SLIM_CREATE_FIXTURE(LightScheduleScript)
    SLIM_FUNCTION(schedule_ForLight_WhereDayIs_AndTimeIs)
    SLIM_FUNCTION(transitionTo_At_ThenLight_ShouldBe)
...
SLIM_END
```

Trigger Clock Transition and Check System Response



```c
static char* transitionTo_At_ThenLight_ShouldBe(void* void_self, SlimList* args)
{
    LightScheduleScript* self = (LightScheduleScript*)void_self;
    int id;
    int lightState;
    const char* result;

    if (! checkArgCount(self, args, 3))
        return self->result;

    id = getId(self, args, 2);
    if (id < 0)
        return self->result;

    if (setTimeResetLightsTransitionClock(self, args) == 0)
        return self->result;

    lightState = FakeLightController_getLightState(id);
    result = convertIntToOnOff(lightState);

    setResult(self, result);
    return self->result;
}
```

Other Story Testing Frameworks

- Cucumber
  - –http://cukes.info/
- Robotframework
  - –http://code.google.com/p/robotframework/

See Related Blogs and Papers

http://www.wingman-sw.com
http://www.wingman-sw.com/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

Helpful References and Resources

- [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
- [TDD] Kent Beck, Test-Driven Development, 2003
- [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
- [XUNIT] Gerard Meszaros, xUnit Testing Patterns, 2008
- [PRAG] Andy Hunt, Dave Thomas, The Pragmatic Programmer
- [KANER] Cem Kaner, et. al. Lessons learned in Software Testing
- Lasse Koskela, Test Driven, 2007

On-line

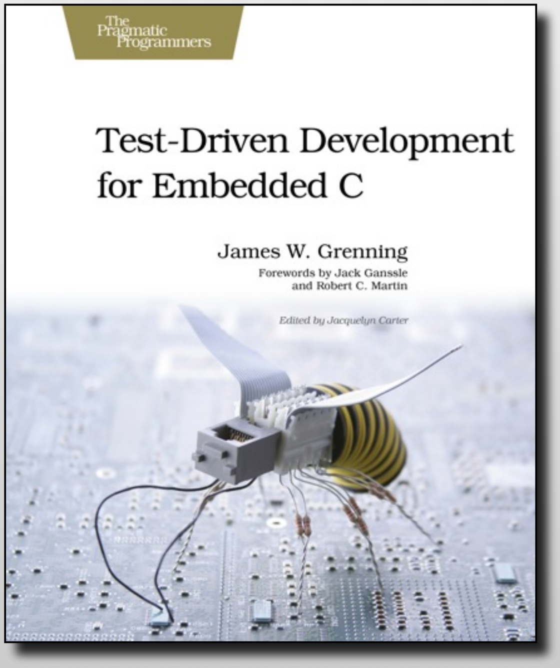
- Test harnesses
  - –[CPPTEST] github, cpputest
  - –[FITNESSE] www.fitnesse.org
- Groups
  - http://groups.yahoo.com/group/testdrivendevelopment
  - http://groups.yahoo.com/group/AgileEmbedded

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

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

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

http://www.wingman-sw.com
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