Test-first Practices for Safety-Critical Development

An Agile Approach

Brian Shoemaker

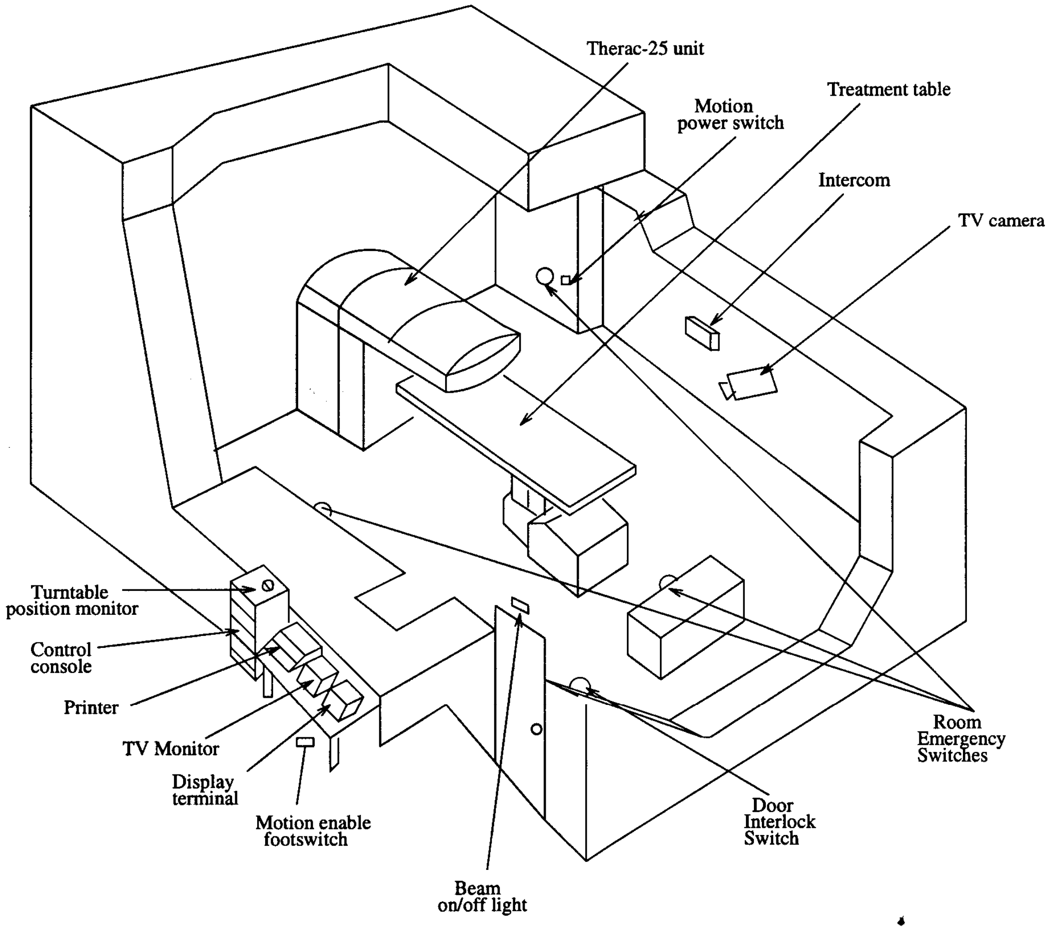
ShoeBar Associates

ShoeBar
Associates

Test First Practices for Safety-Critical Development: an Agile Approach

- **Safety doesn’t reside in any distinct function but in the entire system**
- FDA and international guidances set clear expectations for safety
- Hazard mitigation and validation – closely related – benefit from iteration and user feedback
- Classical hazard analysis may seem completely “up front”, but isn’t
- Consider an interactive environment for meeting regulatory expectations
- Payoff for our effort: safety mitigation, like validation, becomes integral to development

Therac-25: The lessons are still valid

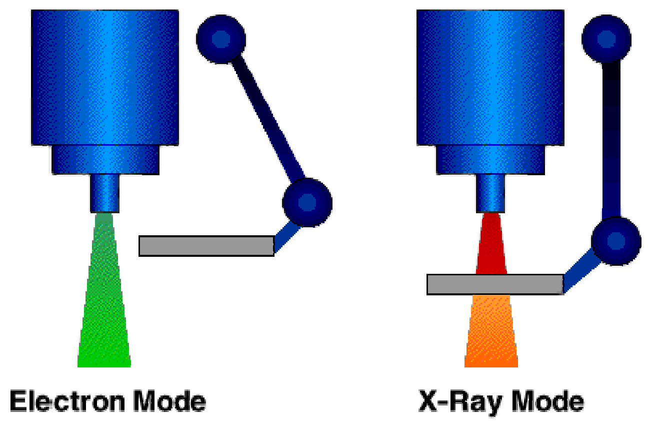


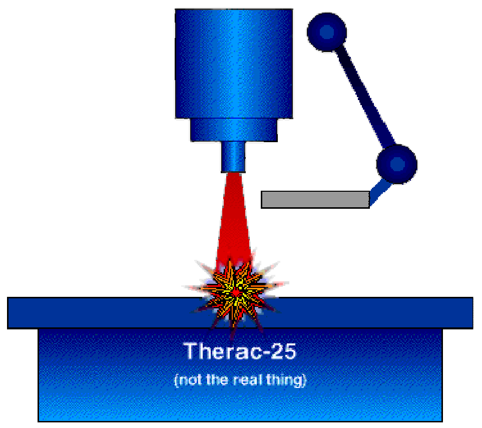
Therac-25: brief summary

- Linear accelerator system built for cancer therapy
- Instrument was further advancement of earlier models (controlled entirely through software)
- 11 Units installed in US / Canada; hundreds of patients treated (thousands of treatments)
- Mechanism: radiation beam destroys cancer tissue
  - Electron beam treats shallow tissue
  - X-rays penetrate deeper, minimal damage to overlying area
  - X-rays produced by hitting metal target with high-energy electrons
- Six overdose accidents (3 fatal): June 1985, July 1985, December 1985, March 1986, April 1986, January 1987
- Overdoses (~100x intended dose, ~20x lethal whole-body dose) traced to two specific software errors

The Therac-25 danger

- Single electron gun produces both modes
- In x-ray mode, electron energy must be ~100x higher (target is a good attenuator)
- Low energy + target = underdose
  High energy + no target = huge overdose





Therac-25: no single safety issue

System:

- Cryptic error messages; errors and malfunctions common
- Operators could, and often did, override Treatment Pause
- Did not produce audit trail that could help diagnose problems
- Relied entirely on software – removed electromechanical safety interlocks that were in previous models

Quality Practices:

- Little documentation during development; no problem tracking after release
- Minimal unit and integration testing
- QA was primarily 2700 hours of use as integrated system
- Failed to look for root causes - seized on each issue as ***the*** problem
- Treated requirements / design / directed testing as troublesome afterthought

S/W Hazards: not just history

- March 6, 2007: AED recalled
  http://www.fda.gov/oc/po/firmrecalls/defibtech03_07.html
  Self-test software may allow a self-test to clear a previously detected low battery condition
- September 2006: Infusion pump recalled
  http://www.fda.gov/cdrh/recalls/recall-081006.html
  Touch-sensitive programming keypad can register a number twice when pressed once (“key bounce”). Pump would deliver more than intended amount of medication, leading to over-infusion and serious harm or death.
- June 6, 2006: Ventilator recalled
  http://www.fda.gov/oc/po/firmrecalls/hamilton06_06.html
  Older generation software - incorrect oxygen cell calibration (without compressed air supply) - can ***disable*** all alarms
- March 6, 2006: Dialysis device recalled
  http://www.fda.gov/cdrh/recalls/recall-081605.html
  Class I recall of dialysis device (11 injuries, 9 deaths): excessive fluid loss may result if caregiver overrides "incorrect weight change detected" alarm. (Device use: continuous solute / fluid removal, acute renal failure patients.)

ShoeBar Associates

Safety: an *emergent* property

- No single system caused Therac-25 safety issues (same software problem in Therac-20!)
- Similarly, no specific element or subsystem caused alarm disabling, pump freeze-up, over-dialysis, or “forgetting” low-battery state
- Ensuring systems are safe requires analyzing how the ***whole*** system behaves (and could misbehave), that could cause harm, and designing to avoid or prevent that behavior

Test First Practices for Safety-Critical Development: an Agile Approach

- *Safety doesn't reside in any distinct function but in the entire system*
- **FDA and international guidances set clear expectations for safety**
- Hazard mitigation and validation – closely related – benefit from iteration and user feedback
- Classical hazard analysis may seem completely "up front", but isn't
- Consider an interactive environment for meeting regulatory expectations
- Payoff for our effort: safety mitigation, like validation, becomes integral to development

FDA concerned if product affects health

Primary concern: software developed as part of a medical product, or to manufacture medical products. Examples:

- S/W to control or communicate with med devices (implantable pacemakers, diagnostic instruments, image analysis systems, therapeutic devices)
- S/W which by itself constitutes a med device
- S/W used to manufacture or to manage quality data for medical products
- S/W to collect and manage clinical trial data

ShoeBar Associates

Guidances / standards address safety (1)

- FDA: Design Control, Medical Devices (March 11, 1997)
- FDA: General Principles of Software Validation (Jan 11, 2002)
- FDA: Premarket Submissions, Software Contained in Medical Devices (May 11, 2005)
- FDA: Off-The-Shelf Software Use in Medical Devices (Sep 9, 1999)
- FDA: Cybersecurity for Networked Medical Devices, OTS Software (Jan 14, 2005)
- FDA: Computerized Systems Used in Clinical Trials (April 1999; May 2007)

Guidances / standards address safety (2)

- ISO 13485: Medical devices – Quality management systems – Requirements for regulatory purposes
- ISO 62304: Medical Device Software – Software Life Cycle Processes
- ISO 14971: Medical devices – Application of risk management to medical devices
- AAMI TIR32:2004 Medical device software risk management
- IEEE Std 1228, Software Safety Plans
- IEEE/ISO/IEC 16085, Software & Systems Life Cycle Processes- Risk Management (based on IEEE 1540 Software Risk Management)

Test First Practices for Safety-Critical Development: an Agile Approach

- *Safety doesn't reside in any distinct function but in the entire system*
- *FDA and international guidances set clear expectations for safety*
- **Hazard mitigation and validation – closely related – benefit from iteration and user feedback**
- Classical hazard analysis may seem completely "up front", but isn't
- Consider an interactive environment for meeting regulatory expectations
- Payoff for our effort: safety mitigation, like validation, becomes integral to development

S/W is covered by design control

Medical device QSR (21 CFR pt 820) mandates ISO-9000 style design control activities for any system design:

- Design / development planning
- Design input
- Design output
- Design review
- Design verification
- Design validation
- Design transfer
- Design change control
- Design History File

ShoeBar Associates

Up-Front Analyses: crucial, difficult

- Requirements
  - Multiple sources (end users, mgmt, regulations, SMEs, “adjacent systems”)
  - Many elements (“actors” and their jobs, system boundary, normal vs. exception cases)
  - Characteristics: Measurable, Coherent, Consistent, Complete, Relevant, Solution Neutral, Ranked, Traceable
- Hazard Analysis
  - FMEA: Identify subsystem failure modes in a system that hasn’t been designed yet
  - FTA: Imagine potential dangers / harmful situations for a product that is still only a concept

Classical engineering: linear progression

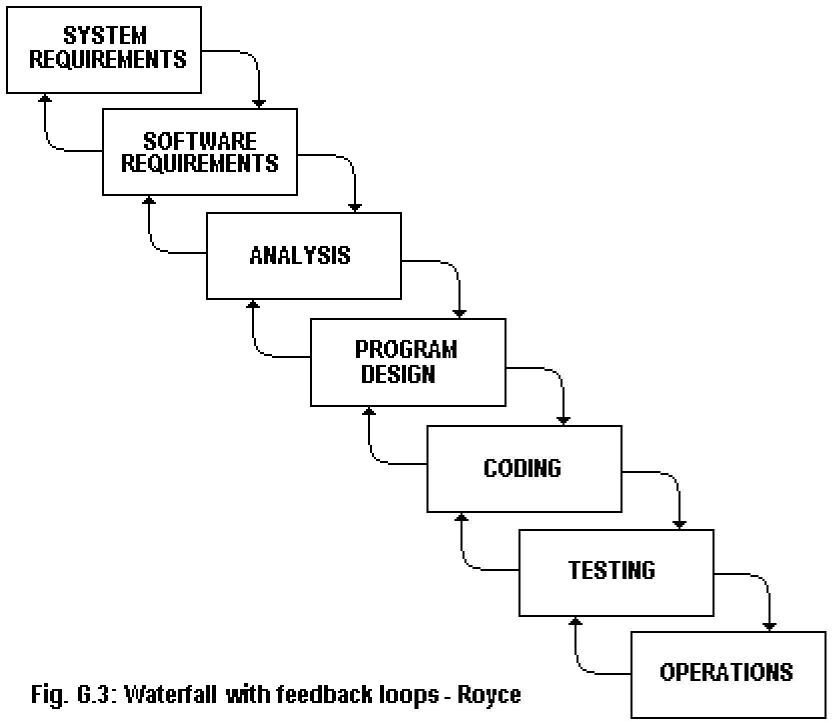
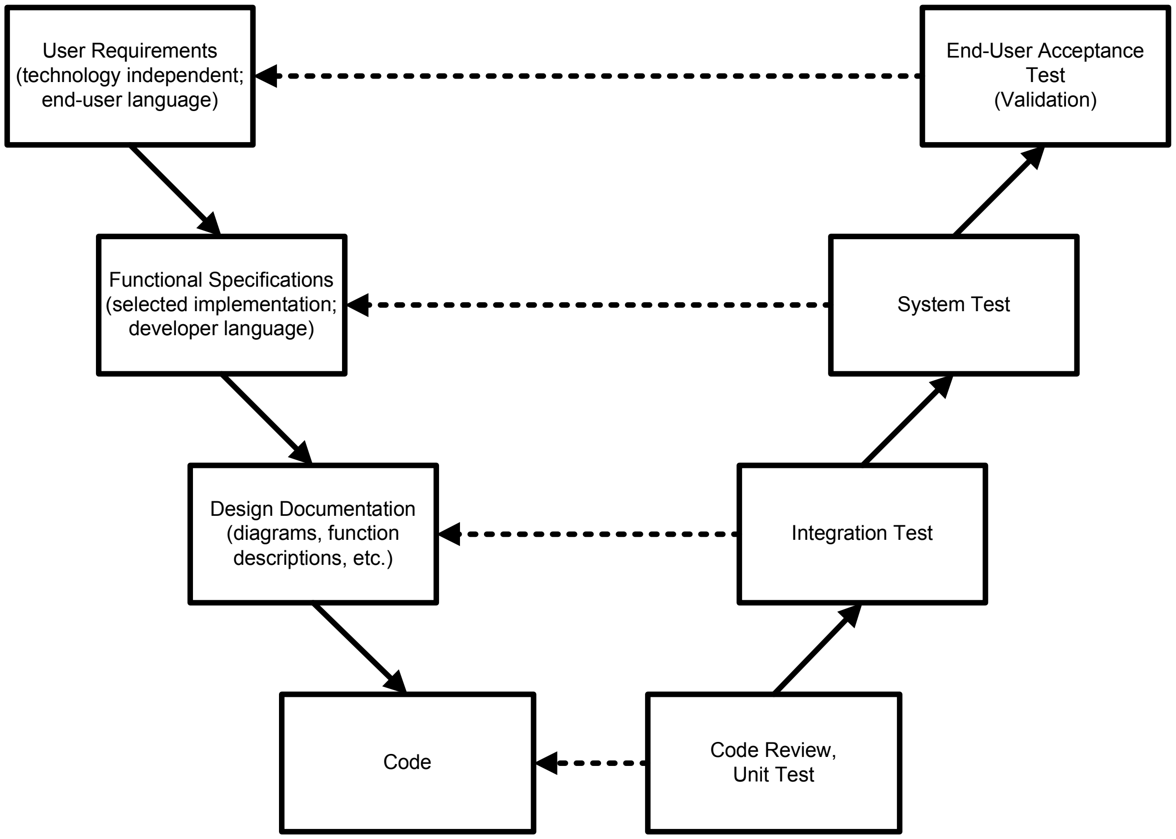


Fig. G.3: Waterfall with feedback loops - Royce

Even with feedback between adjacent steps, adjusting is difficult late in a project.

“V” model: little change in sequence


User Requirements (technology independent; end-user language)
End-User Acceptance Test (Validation)
Functional Specifications (selected implementation; developer language)
System Test
Design Documentation (diagrams, function descriptions, etc.)
Integration Test
Code
Code Review, Unit Test

BUT notice how this ties together activities at corresponding “levels”

ShoeBar Associates

Validation / Testing: welcome to my world!

- Validation: tie “what was built” to “what was needed” (i.e. requirements)
- Testing: part of what proves validation
- Requirements: we often learn them ***in context***
- Planning to a “linear” process almost always curtails testing!

ShoeBar Associates

Late learned: not addressed?

Immunoassay Instrument:

ShoeBar Associates

Late learned: not addressed?

Immunoassay Instrument:

- Instrument transmits measurements to external computer (via RS232)
- Sandwich assay for hCG is known to “hook” (extremely high concentrations give signal as if low concentration)
- Test is therefore run at several dilutions – but the measurement is meaningless unless accompanied by dilution factor
- Ver. 5 software – transmitted results had concentration but not dilution factor

Test First Practices for Safety-Critical Development: an Agile Approach

- *Safety doesn't reside in any distinct function but in the entire system*
- *FDA and international guidances set clear expectations for safety*
- *Hazard mitigation and validation – closely related – benefit from iteration and user feedback*
- **Classical hazard analysis may seem completely "up front", but isn't**
- Consider an interactive environment for meeting regulatory expectations
- Payoff for our effort: safety mitigation, like validation, becomes integral to development

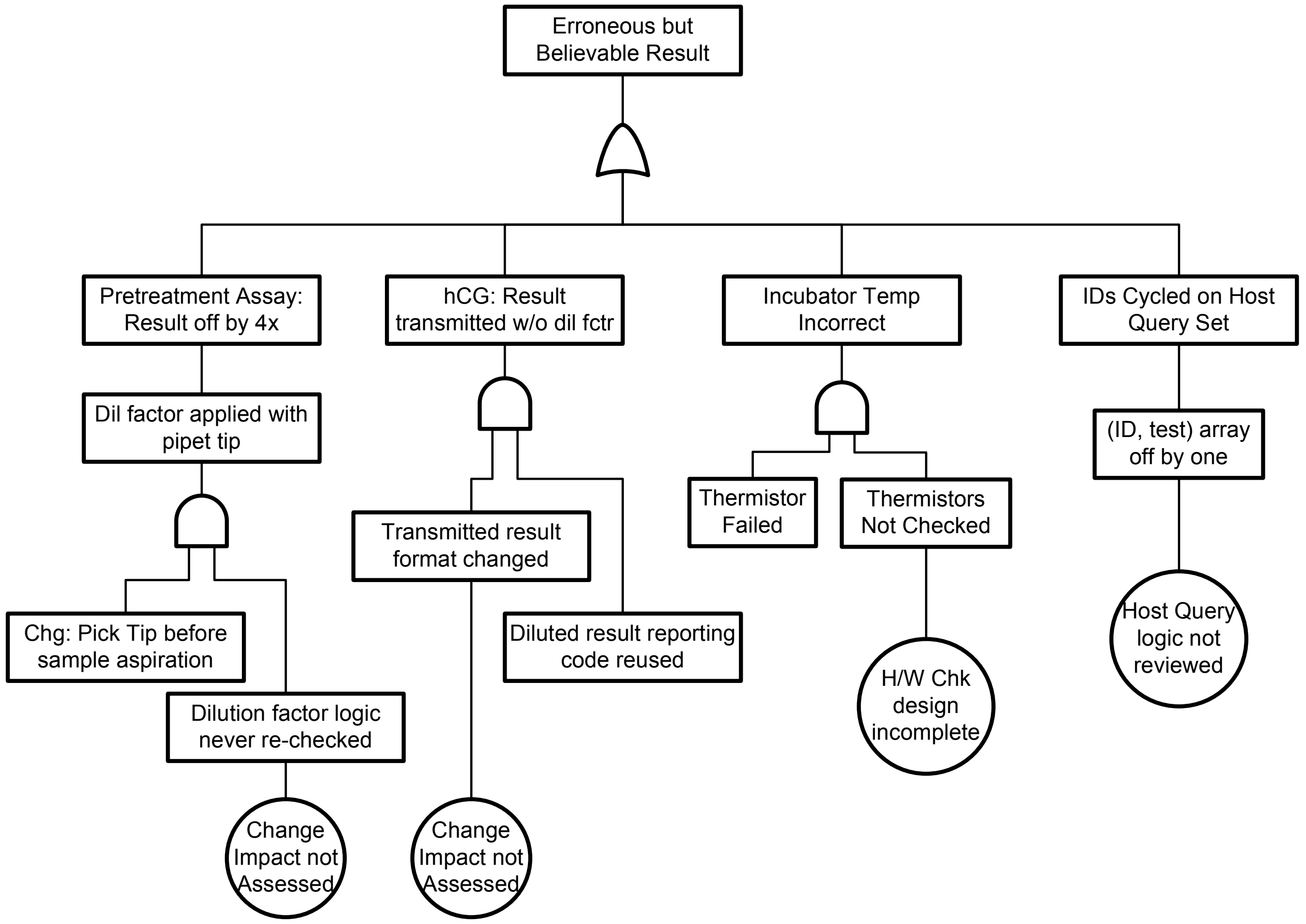
FMEA: Build up from component failures

Failure Mode	Effect	Causes	SR1	Mitigation	SR2
Sample ID / results array off by one	Wrong results reported	Inconsistent array logic; incorrect initialization	5	Optional – operator approve results before saving	2
Initialization fails to warm up lamp	Can't perform analyses	Startup logic can be set to skip steps and left that way	4	Reset all startup parameters on initialization	1
Dilution factor associated with pipet tip, picked in advance	Wrong result reported (dilution applied to wrong sample)	Counting logic not rechecked when pick-in-advance process introduced	5	(a) Track dilution and pipet tip separately; (b) show dilution with reported result	1

SR1 = Severity rating before mitigation; SR2 = severity rating after mitigation
Severity (sample values only): 5 = critical; 1 = nuisance
Note this analysis does not include two of the standard engineering estimates: occurrence (probability) or detection.

ShoeBar Associates

FTA: work back from potential hazards


Erroneous but Believable Result
Pretreatment Assay: Result off by 4x
hCG: Result transmitted w/o dil fctr
Incubator Temp Incorrect
IDs Cycled on Host Query Set
Dil factor applied with pipet tip
(ID, test) array off by one
Thermistor Failed
Thermistors Not Checked
Transmitted result format changed
Chg: Pick Tip before sample aspiration
Diluted result reporting code reused
Host Query logic not reviewed
H/W Chk design incomplete
Dilution factor logic never re-checked
Change Impact not Assessed
Change Impact not Assessed

ShoeBar Associates

These methods ***can*** fit in an Agile context!

- Analyzing hazards up front isn't *wrong* – it's a head start
- Every reference on FMEA / FTA says the same thing: do the analysis early and revisit it often
- The mistake many engineering groups commit is to analyze hazards only once (worst: at the end of design!)
- Effective hazard analysis permits learning, and acting on knowledge, as a design is refined

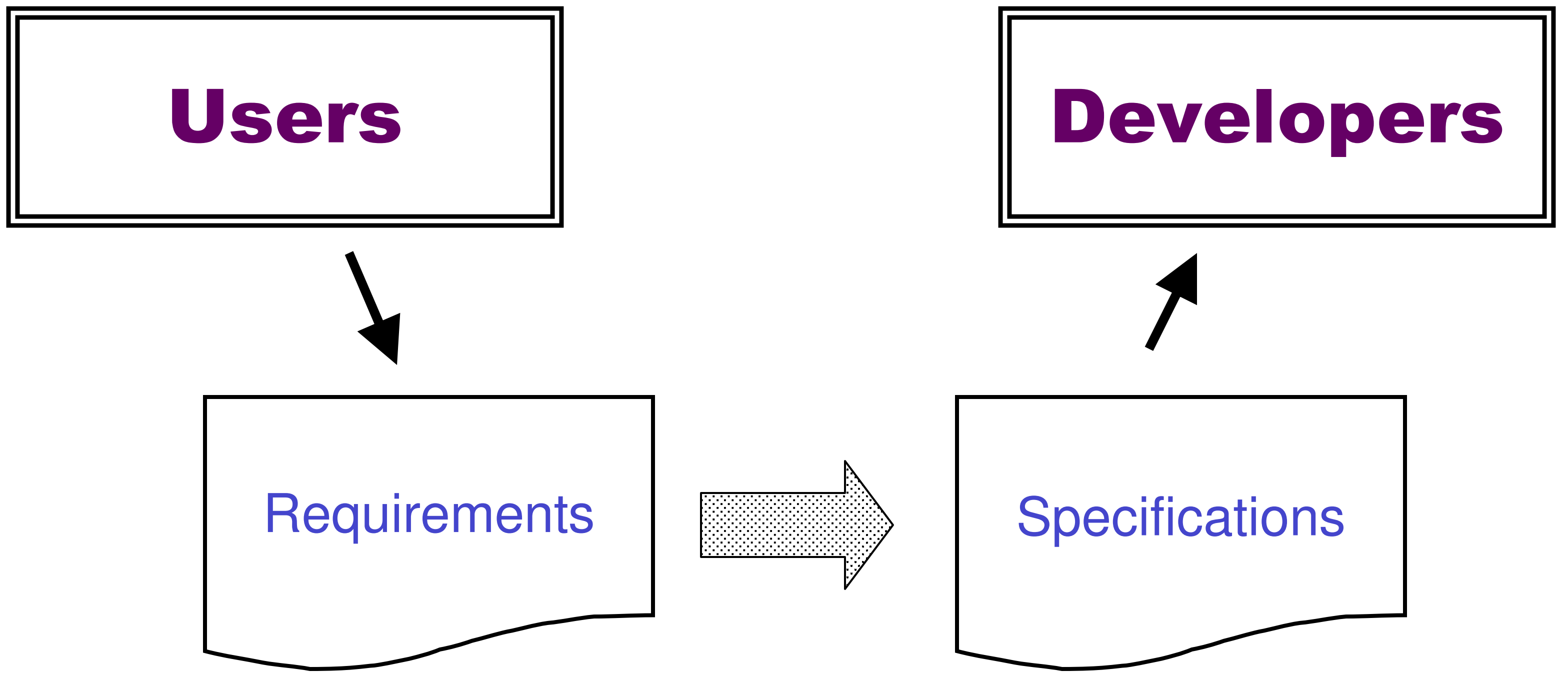
ShoeBar Associates

Test First Practices for Safety-Critical Development: an Agile Approach

- *Safety doesn't reside in any distinct function but in the entire system*
- *FDA and international guidances set clear expectations for safety*
- *Hazard mitigation and validation – closely related – benefit from iteration and user feedback*
- *Classical hazard analysis may seem completely "up front", but isn't*
- **Consider an interactive environment for meeting regulatory expectations**
- Payoff for our effort: safety mitigation, like validation, becomes integral to development

ShoeBar Associates

Requirements vs. Specification



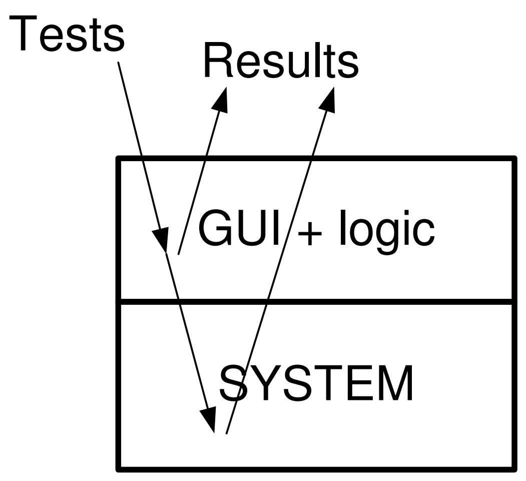
ShoeBar Associates

Test directly from functional specification?

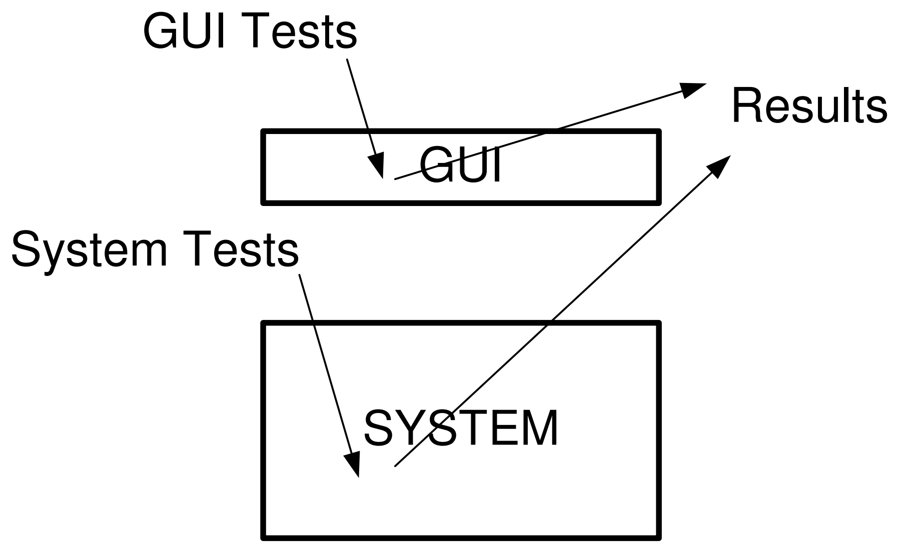
- Requirements must be testable
- We make our implementation choices
- Functional spec / use case / story, same intent: known inputs → expected output
  (We only <u>understand</u> the requirements when they're written that way!)
- To make this possible requires discipline in layered design (UI | Working logic | Data Store)

Our goal: to test the logic directly

- Don't:



- Do:



Environment: write tests first, then code

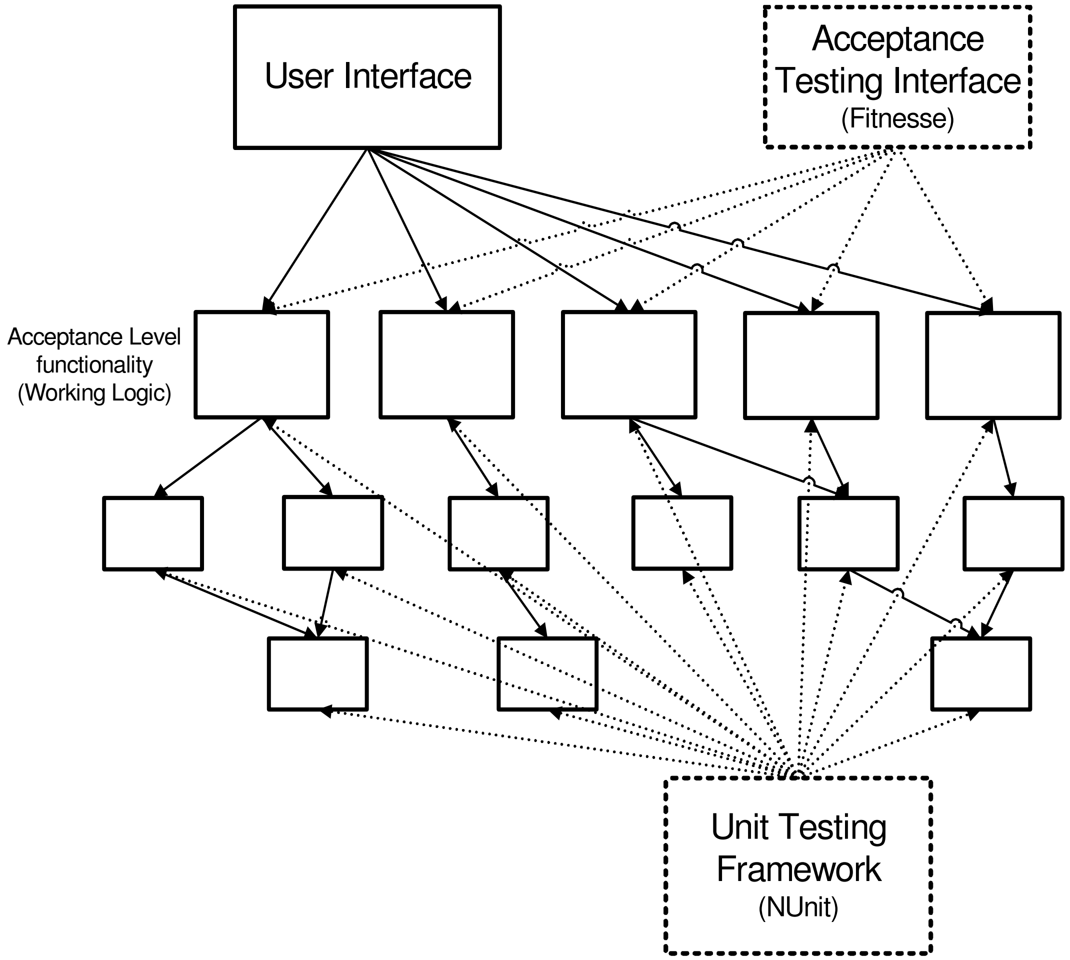
a) Develop high-level design at block-diagram level.

b) Write functional specification as executable tests.

c) Write unit tests for intended functions

d) Write code and debug until tests pass

e) Review, repeat as necessary

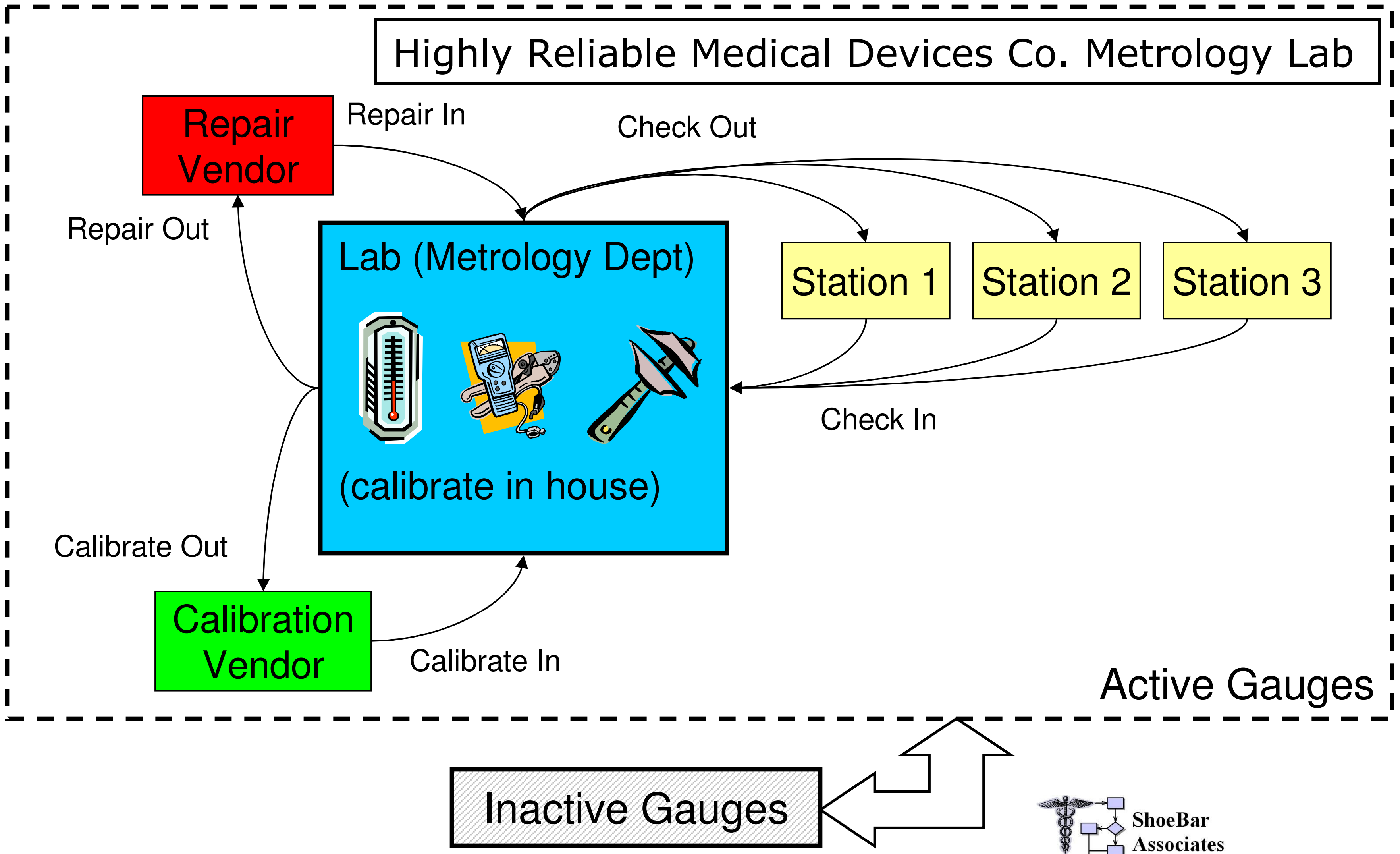

User Interface
Acceptance Testing Interface (Fitnesse)
Acceptance Level functionality (Working Logic)
Unit Testing Framework (NUnit)

ShoeBar Associates

Development = Discovery

- Refine requirements to make functional specs
- Elaborate functional specs: ask what-if questions
- Examine hazards via FMEA / FTA methods
- Specify acceptance tests to express functional specs; correct these as requirements are revised
- Build safety tests to check for identified hazards (simulate as necessary)
- Test and re-test code with each addition or correction
- Move ahead *iterativel<u>y</u>*: implement a selected group of features / functions in each cycle

Consider a practical example



ShoeBar Associates

Gauge Program User Requirements

User Security

R_S1: The metrology lab needs to limit use of the calibration system to authorized users.

R_S2: The metrology lab needs to restrict specific actions according to the employee's job function.

Gauge Data

R_D1: The metrology lab needs to maintain an up-to-date list of all gauges.

R_D2: The metrology lab needs to store properties, calibration dates, and calibration data for each gauge.

Gauge Operations

R_O1: The metrology lab needs the ability to set and track status of each gauge (in lab, checked out, out for repair, out for calibration, inactive).

R_O2: The metrology lab needs to be able to calibrate any gauge which is considered available (i.e. in lab and Master Gauge calibration is in date.)

R_O3: The metrology lab needs to have the calibration routine set a calibration to PASS only if all measurements are within tolerance.

R_O4: The metrology lab needs to prevent any gauge from being checked out for use if a calibration has not been carried out and passed within the calibration period.

R_O5: The metrology lab needs to record a history of all calibrations and status changes for every gauge.

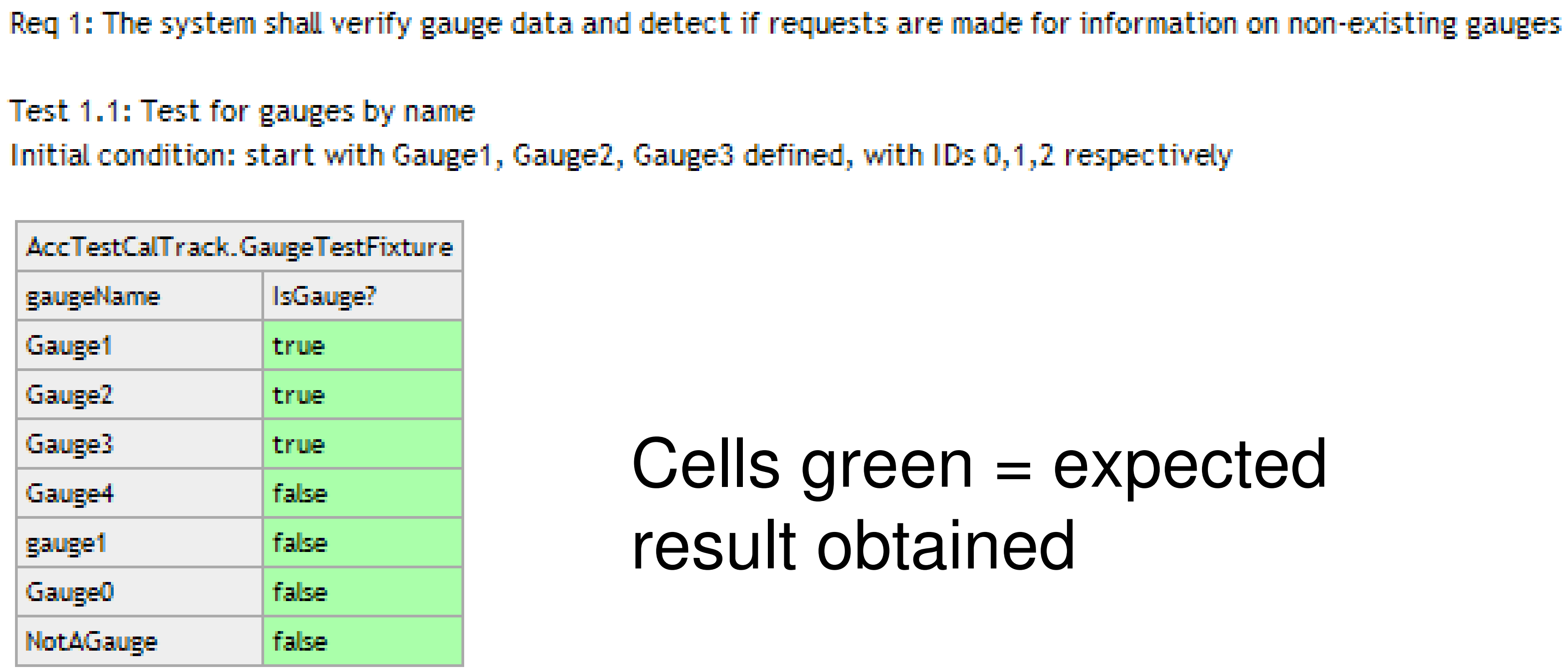
Reporting

(There will be a set of reporting requirements.)

Tests include functional spec, inputs, expected results

```
Test capability of Calibration Tracking System:

Gauge names are case sensitive


!define COMMAND_PATTERN {%m %p}
!define TEST_RUNNER {dotnet\FitServer.exe}
!define PATH_SEPARATOR {;}

!path dotnet\*.dll
!path C:\Agile
Rules\CSharp_NUnitFitNesse_Solns\CalTrack\AccTestCalTrack\bin\Debug\AccTestCalTrack.dll

Req 1: The system shall verify gauge data and detect if requests are made for information
on non-existing gauges

Test 1.1: Test for gauges by name
Initial condition: start with Gauge1, Gauge2, Gauge3 defined, with IDs 0,1,2 respectively

!|AccTestCalTrack.GaugeTestFixture|
|gaugeName | IsGauge? |
|Gauge1 | true |
|Gauge2 | true |
|Gauge3 | true |
|Gauge4 | false |
|gauge1 | false |
|Gauge0 | false |
|NotAGauge | false |
```

Run test directly from this page



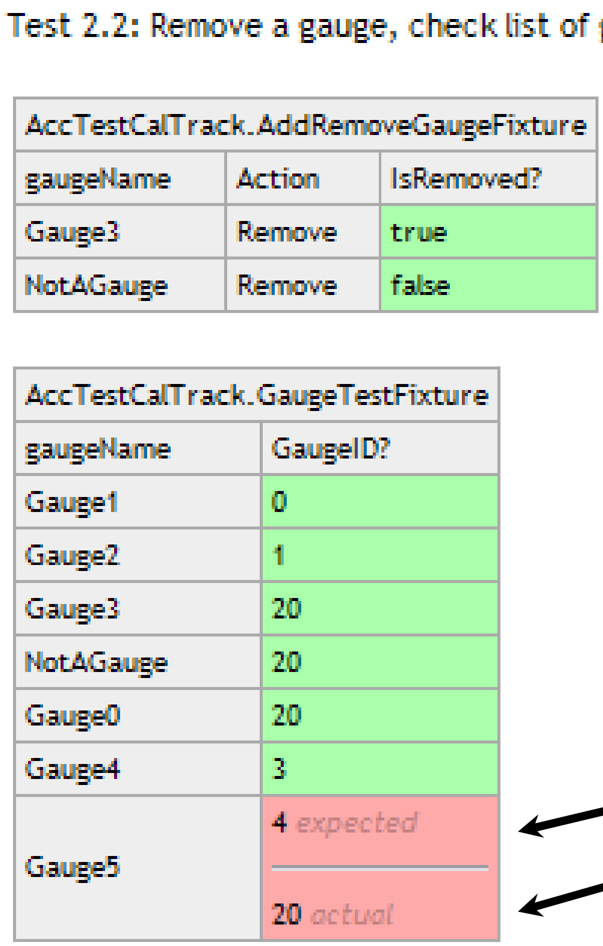
Req 1: The system shall verify gauge data and detect if requests are made for information on non-existing gauges

Test 1.1: Test for gauges by name
Initial condition: start with Gauge1, Gauge2, Gauge3 defined, with IDs 0,1,2 respectively

AccTestCalTrack.GaugeTestFixture	
gaugeName	IsGauge?
Gauge1	true
Gauge2	true
Gauge3	true
Gauge4	false
gauge1	false
Gauge0	false
NotAGauge	false

Cells green = expected result obtained

We readily see when actual ≠ expected



Test 2.2: Remove a gauge, check list of gauges

AccTestCalTrack.AddRemoveGaugeFixture		
gaugeName	Action	IsRemoved?
Gauge3	Remove	true
NotAGauge	Remove	false

AccTestCalTrack.GaugeTestFixture	
gaugeName	GaugeID?
Gauge1	0
Gauge2	1
Gauge3	20
NotAGauge	20
Gauge0	20
Gauge4	3
Gauge5	4 *expected* / 20 *actual*

Cells green = results OK

In this case, instead of expected ID, result was a "doesn't exist" value – issue with code!

Pieces we need for this approach

- User requirements
- Close collaboration with users
- Acceptance testing framework (automation is essential)
- Ability to simulate hazards
- Unit testing tool
- Compatible development environment

What about boards / devices?

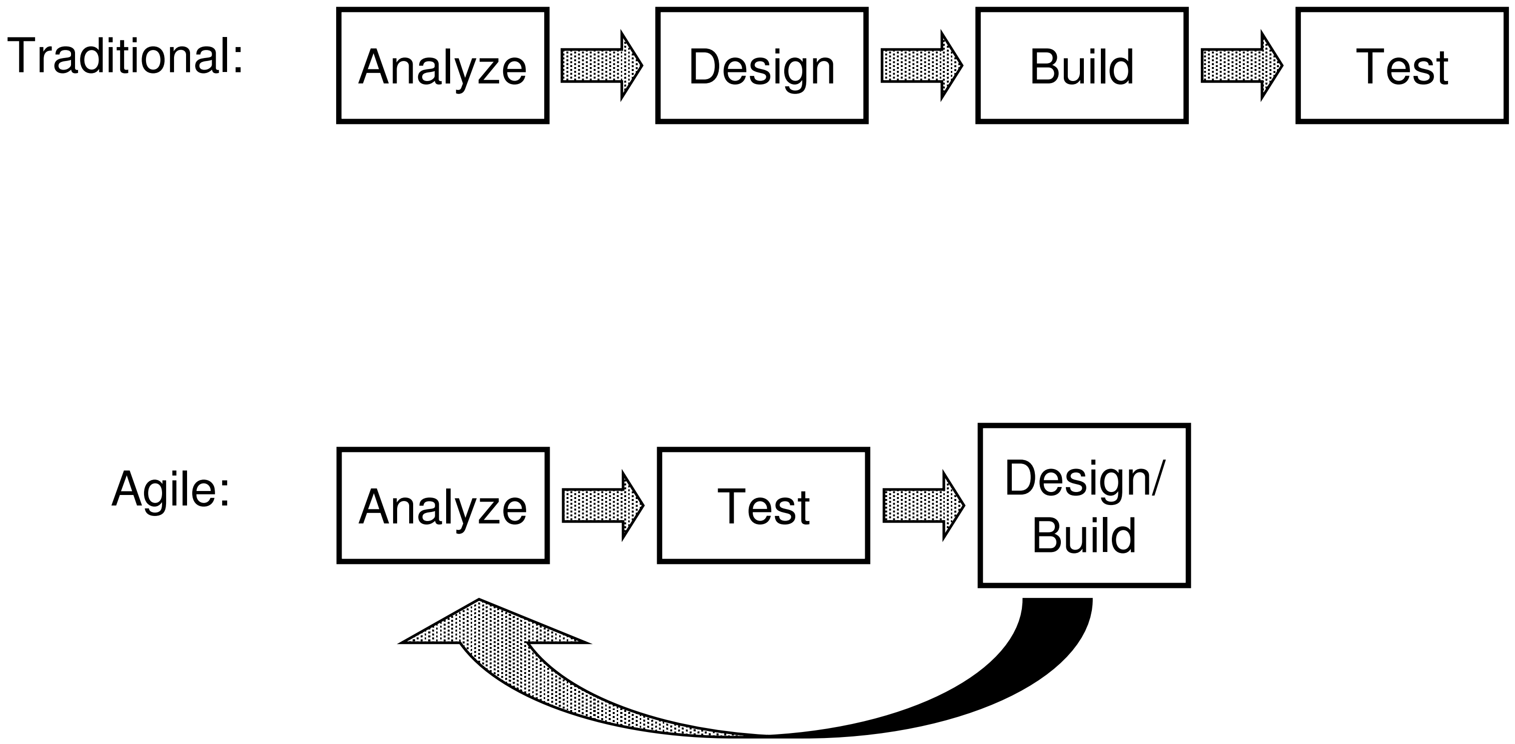
- Divide software into conceptual levels
  - Model (where feature's logic is implemented)
  - Hardware (where actions such as speed control actually happen)
  - Conductor (to relay events and data between model and hardware)
- Use hardware harnesses to:
  - Introduce newest program onto device
  - Generate known input to board / device
  - Read device’s output response
- Complete environment includes unit tests, system tests, mocks as needed, and hardware for automation/communication

Ref: Fletcher et al, 2007. (Agile conference)

Test First Practices for Safety-Critical Development: an Agile Approach

- *Safety doesn’t reside in any distinct function but in the entire system*
- *FDA and international guidances set clear expectations for safety*
- *Hazard mitigation and validation – closely related – benefit from iteration and user feedback*
- *Classical hazard analysis may seem completely “up front”, but isn’t*
- *Consider an interactive environment for meeting regulatory expectations*
- **Payoff for our effort: safety mitigation, like validation, becomes integral to development**

Turn Development almost backward



ShoeBar Associates

Safety, quality, efficiency all benefit

- Requirements → executable specifications enforces clarity, reveals implied requirements
- Traceability is built in
- Evolution of requirements / functional spec / hazard analysis is woven into the development process
- Test framework permits repeating tests frequently (ongoing regression test)
- Testing remains independent: non-developers can add, refine tests as development proceeds
- Requirements, functional spec, hazard analysis / mitigation, tests are all updated concurrently

References – FDA documents

Design Control Guidance For Medical Device Manufacturers (March 11, 1997), http://www.fda.gov/cdrh/comp/designgd.html

General Principles of Software Validation (January 11, 2002), http://www.fda.gov/cdrh/comp/guidance/938.html

Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices (May 11, 2005), http://www.fda.gov/cdrh/ode/guidance/337.html

Off-The-Shelf Software Use in Medical Devices (Sep. 9, 1999), http://www.fda.gov/cdrh/ode/guidance/585.html

Cybersecurity for Networked Medical Devices Containing Off-the-Shelf (OTS) Software (Jan. 14, 2005), http://www.fda.gov/cdrh/comp/guidance/1553.html

Computerized Systems Used in Clinical Trials
(April 1999) http://www.fda.gov/ora/compliance_ref/bimo/ffinalcct.htm;
(May 2007) http://www.fda.gov/cber/gdlns/compclintrial.htm

21 CFR Part 11: Electronic Records and Signatures (Aug. 1997), http://www.fda.gov/ora/compliance_ref/part11/FRs/background/pt11finr.pdf

Other references

EduQuest, Inc., "FDA Auditing of Computerized Systems and Part 11," notes from course given July 2005.

Fletcher, M., Bereza, W., Karlesky, M., and Williams, G., "Evolving into Embedded Development", Proceedings Agile 2007 (13-17 August 2007), IEEE Computer Society, 2007.

Institute of Electrical and Electronics Engineers, "Recommended Practice for Software Requirements Specifications" (IEEE 830-1998), 01-May-1998.

Leveson, N.G., *Safeware - System safety and Computers. 1 ed*. 1995: Addison-Wesley Publishing Company Inc. [Revised version at: http://sunnyday.mit.edu/papers/therac.pdf] *(Therac-25 investigation)*

Royce, Winston W., “Managing the development of large software systems: Concepts and techniques,” in: *Proceedings, IEEE WESCON* (August 1970). *(Waterfall method)*

Robertson, James and Suzanne, *Mastering the Requirements Process*, Harlow (England), Addison-Wesley / ACM Press, 1999.

Schneider, Geri, and Jason P. Winters, *Applying Use Cases: A Practical Guide*, Harlow (England), Addison-Wesley / ACM Press, 1998.

Weyrauch, Kelly, “Safety-Critical. XP Rules.”, *Better Software*, July/August 2004.

ShoeBar Associates

Contact information

Brian Shoemaker, Ph.D.
Principal Consultant, ShoeBar Associates
199 Needham St, Dedham MA   02026 USA
781-929-5927
bshoemaker@shoebarassoc.com
http://www.shoebarassoc.com

Acknowledgement: my collaborator (co-presenter at other conferences) contributed significantly to the ideas and examples offered here.
Ron Morsicato
Managing Partner, Agile Rules
162 Marrett Rd., Lexington, MA 02421 USA
978-973-1603
ronm@agilerules.com
http://www.agilerules.com

ShoeBar Associates