Onward to Approval:

Documenting Agile Development for Regulatory Compliance

Brian Shoemaker, Ph.D.
Principal Consultant, ShoeBar Associates

© 2012 ShoeBar Associates
All Rights Reserved

ShoeBar Associates

1

My Background

- Originally an analytical chemist
- 15 y in clinical diagnostics (immunoassay):
  analytical support → assay development → instrument software validation
- 6 y as SW quality manager (5 in clinical trial related SW)
- 7 y as independent validation consultant to FDA-regulated companies – mostly medical device
- Active in: software validation, Part 11 evaluation, software quality systems, auditing, training

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

2

Onward to Approval – Documenting Agile Development

- **Agile vs IEC 62304: apparent contradiction?**
- *Quality – avoid bad news late*
- *Risk Management fits in well*
- *Documentation can be iterative!*
- *Agile can be clearly superior*

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

3

Why this discussion?

- Traditional doc-heavy SW development is ***expensive***, ***slow***, and ***error prone***
- Regulatory bodies rightly concerned with product software vs safety (OSEL report: 24% of 2011 medical device recalls were for software!)

- Classic belief: tightly controlled process → better engineering
- Agile is highly productive, but seems the antithesis of tightly controlled process

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

4

Agile Methods for Device SW?

- Simple answer: yes
- Discipline is necessary – but that's always true
- Compare IEC 62304 and the Agile Manifesto: despite contrast, there's common ground

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

5

IEC 62304 – All about *processes*

Key Principles:

- Have a Quality Management System
- Use a risk management approach
- Classify software according to safety
- Have processes for known development steps
- Use maintenance processes
- Manage configuration (versions)!
- Follow a problem resolution process

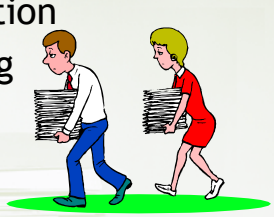
© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

6

Known Development Processes?

- Planning
- Requirements analysis
- Architectural design and detailed design
- Unit Implementation and verification
- Integration and integration testing
- System Testing
- Software release



These processes may sound heavy – but we'll come back to what the standard <u>doesn't</u> say or require!

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

7

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.

http://agilemanifesto.org/

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

8

These seem contradictory . . .

. . . But the common goal is quality and as a corollary, safety!

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

9

Onward to Approval – Documenting Agile Development

- *Agile vs IEC 62304: apparent contradiction?*
- **Quality – avoid bad news late**
- *Risk Management fits in well*
- *Documentation can be iterative!*
- *Agile can be clearly superior*

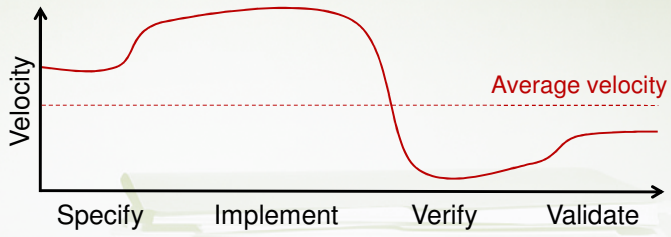
© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

10

Common Scenario . . .

- Project velocity varies greatly
- Much slower at integration time



Solution: Pace yourself! It's a marathon, not a sprint

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Lean Thinking

Lean Thinking

Lean Principles:
Zero Defects
Minimize Work In Progress
Continuous Improvement

Lean Manufacturing
(All kinds)

≠

Lean Development
(S/W, H/W, Services, other)

Classic "best practices"
Agile practices:
- Continuous Integration
- Automated unit tests
- Small co-located teams

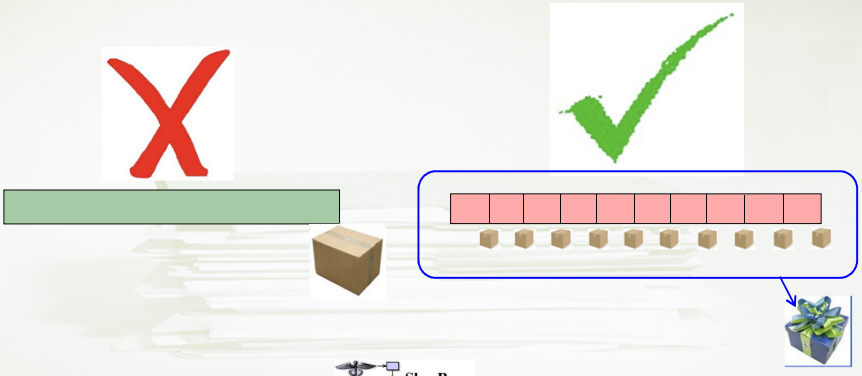
Our "pain points":
Bad news late in projects
Implementation different from spec
Documentation issues

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Avoid Late Integration

- Integrate new work as you go
- Incremental deliveries early & often



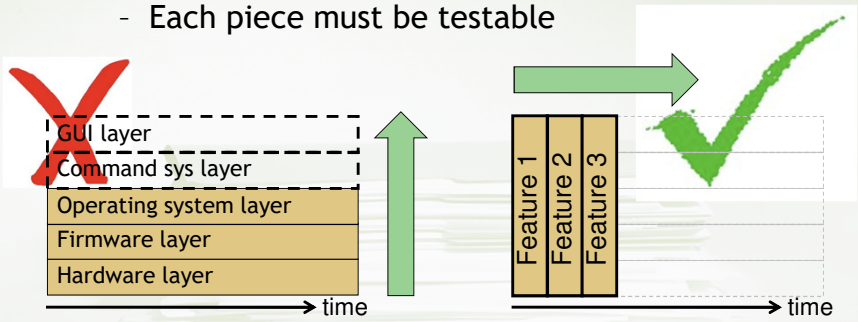
© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

13

Deliver Incrementally

- To deliver incrementally you must:
  - Carve the work into functional pieces
  - Each piece must be small
  - Each piece must be testable



© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

14

Work pieces: user stories

- User stories are similar to use cases
  - Written from customer view point
  - Written using words all understand
- Smaller than use cases
- Estimates are owned by the team
  - *Equally* likely to be too high or too low

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

15

Example User Story

- Story – Card, Conversation, Confirmation –
  headline, narrative, test

Cards have the headline

Story

Verify Sensor Module OS runs on the new Sensor Module Radar

Narrative details captured in documents

Conditions of Satisfaction

Both In and Out values are displayed and out value should equal to 2*In value

CoS becomes the root of story acceptance test

An old idea: If you have a clear goal, you are much more likely to achieve it.

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

16

4 Stages of Story Refinement

Story and CoS (Conditions of satisfaction) defined – from "user needs and intended uses"

Product Owner conducts the team in 'planning poker' story points estimation

At iteration planning meeting, team defines tasks & team-owned task estimates (hours)

During iteration, team pulls further detail that was not needed for estimation. PO validates stories, with Testers' help.

FDA considers software validation to be "**confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled.**" – GPSV p. 6

Automated agile tests

GPSV = General Principles of Software Validation

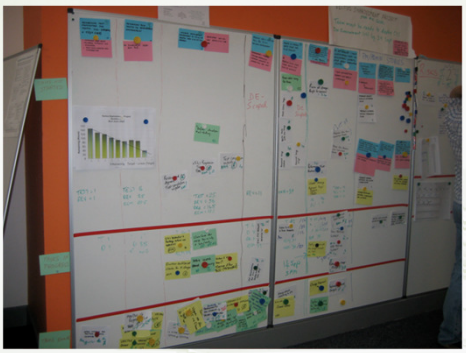
© 2012 ShoeBar Associates All Rights Reserved

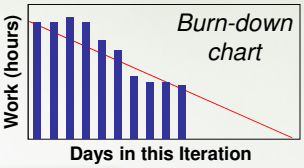
ShoeBar Associates

17

Total Transparency

- Status reporting is not separate from team's own way of tracking their work





Each day:

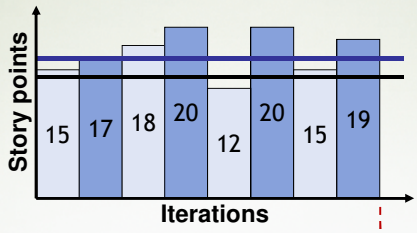
- Team estimates hours remaining for each task
- All remaining hours are summed
- That total is today's data point on burn-down chart

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

18

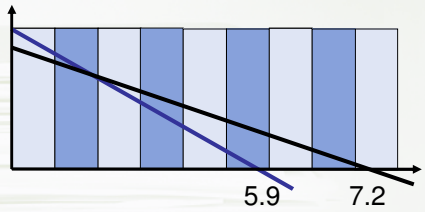
Predictable Project Speed



Mean (Last 8) = 17
Mean (Worst 3) = 14

Q. How long to finish project if 100 story points of work remains in product backlog?

A. If it's 14 points/iter, then it takes 7.2 iterations. If it's 17 points/iter. then it will take 5.9 iterations. You can be conservative or not, as appropriate.



© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

19

Onward to Approval – Documenting Agile Development

- *Agile vs IEC 62304: apparent contradiction?*
- *Quality – avoid bad news late*
- **Risk Management fits in well**
- *Documentation can be iterative!*
- *Agile can be clearly superior*

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

20

Objection → Discipline

Perception: Agile has no formal hazard mitigation process

- Developers may not identify hazards arising from software
- Is zero-defects the same as risk-free?
- Mitigations may not be documented or tested
- Can teams avoid negating a mitigation in later development or refactoring?

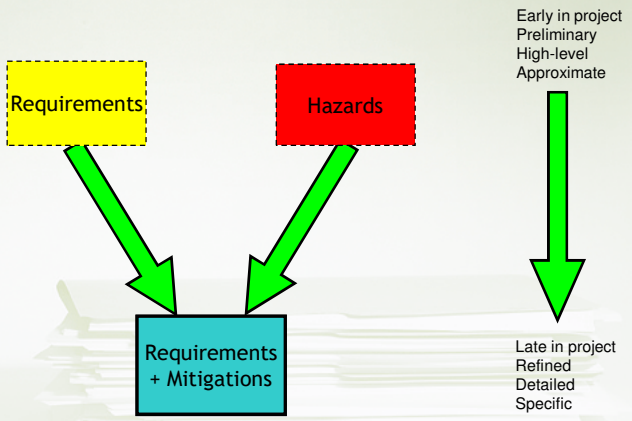
Discipline: Include risk management in each iteration
Evaluate hazards and update risk management file
Capture mitigations in requirements
Plan to include review of risk management docs

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

21

Requirements / Hazards: Converging Analyses



© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

22

Hazards: analyze early and often

- Systematic methods (FMEA / FMECA, FTA) help analyze potential hazards
- Evaluate hazards repeatedly throughout project
- Just as requirements (aka User Stories) become more refined as design evolves -
- So identifying hazard mitigations is changing or adding to requirements
- Think of a hazard as a negative user story

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

23

Hazards: Often Caught in Context

- Direct failure
  Software flaw in normal, correct use of system causes or permits incorrect dosage or energy to be delivered to patient.
- Permitted misuse
  Software does not reject or prevent entry of data in a way that (a) is incorrect according to user instructions, and (b) can result in incorrect calculation or logic, and consequent life-threatening or damaging therapeutic action.
- User Complacency
  Although software or system clearly notes that users must verify results, common use leads to over-reliance on software output and failure to cross-check calculations or results.

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

24

Hazards: Often Caught in Context

- User Interface confusion
  Software instructions, prompts, input labels, or other information is frequently confusing or misleading, and can result in incorrect user actions with potentially harmful or fatal outcome.
- Security vulnerability
  Attack by malicious code causes device to transmit incorrect information, control therapy incorrectly, or cease operating. No examples in medical-device software known at this time, but experience in personal computers and "smart" cellular phones suggests this is a serious possibility.

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

25

Lean-Agile adapts well to hazard mitigation

- Early analysis not static – review & revise as iterations proceed
- Users / product owner have multiple chances to uncover hazard situations
- Hazards can be simulated via “mock objects” in test suite
- Flexible, adaptive method can react to hazards learned during development (considered “negative user stories”)

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

26

Onward to Approval – Documenting Agile Development

- *Agile vs IEC 62304: apparent contradiction?*
- *Quality – avoid bad news late*
- *Risk Management fits in well*
- **Documentation can be iterative!**
- *Agile can be clearly superior*

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

27

Iterative Docs – Objections, Answers

Points to counter:

- Lack of defined requirements
- Lack of structured review/release cycles
- Lack of documentation

Advantages to offer:

- Ability to resolve incomplete / conflicting requirements
- Ability to reprioritize requirements (mitigations) as system takes shape
- Many chances to identify hazards (controls not frozen too soon)

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

28

Objection → Discipline

Perception: Agile methods lack formal requirements

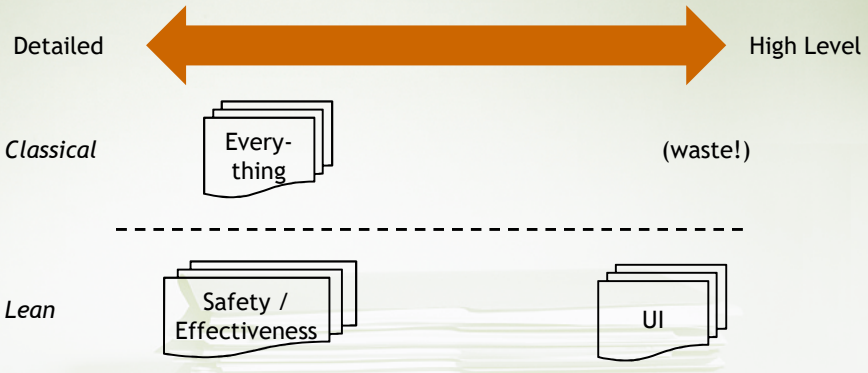
- "User Stories" are usually vague – do they need to under doc control?
- How and when does a complete requirement document get assembled?
- Developers focus on implementation – what about fundamental requirements?
- Will developers pay attention to issues that affect safety or effectiveness?
- Can we be sure that something implemented in one iteration won't be eliminated in later refactoring?
- Are requirements under configuration management?

Discipline: Capture requirements during the iteration

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Not all Requirements are Equal

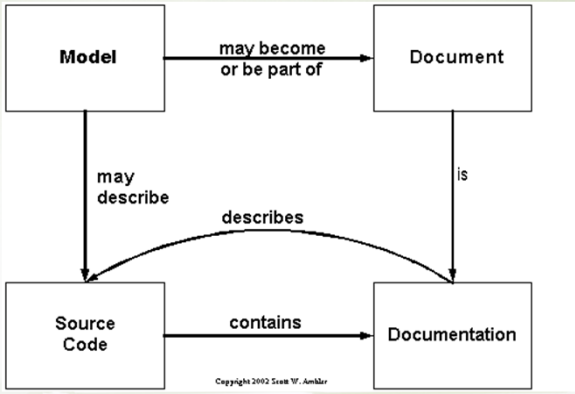


Source: Pate & Russell, 2010

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Models vs. Code vs. Documentation



© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

31

Some of Scott Ambler's Points

- The fundamental issue is communication, not documentation
- Document stable things, not speculative things
- Well-written documentation supports organizational memory effectively, but is a poor way to communicate during a project
- With high quality source code and a test suite to back it up you need a lot less system documentation
- Each system has its own unique documentation needs, one size does not fit all
- The investment in system documentation is a business decision, not a technical one
- Create documentation only when you need it at the appropriate point in the life cycle

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

32

Document Effectively but Flexibly

- SOPs: focus on deliverables
  - Cover all required areas
  - Specify outputs, not strict order of completion
- Development outputs: focus on information
  - Requirements, architecture/design, hazard analysis
  - View as deliverables rather than process support
  - Consider nontraditional form if this makes information capture easier or more automatic

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

33

Use Appropriate Tools

- Initial user stories – may simply be index cards
- Requirements manager as they're elaborated
- Unit test harness
- Consider code-comment document extraction
- User-focused functional / system test engine – best if tied to requirements, e.g. FitNesse

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

34

Don't Forget Communication!

- With customer / product owner – for input and ongoing feedback
  - Iteration end Demo; discussions during iteration
- Among team members – frequent but brief, to build team dynamics
  - Daily stand-up meeting; team room conversations
- With management – to show progress and build trust
  - Information radiators; Iteration end demo

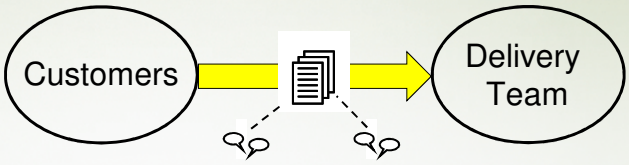
© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

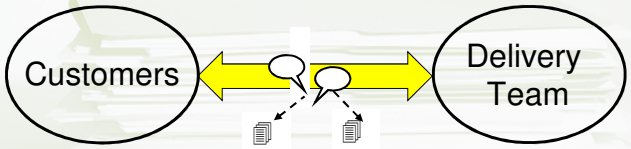
35

Documents as Output

- From Document-centric, supported by Conversation



- To Conversation-centric, supported by documents

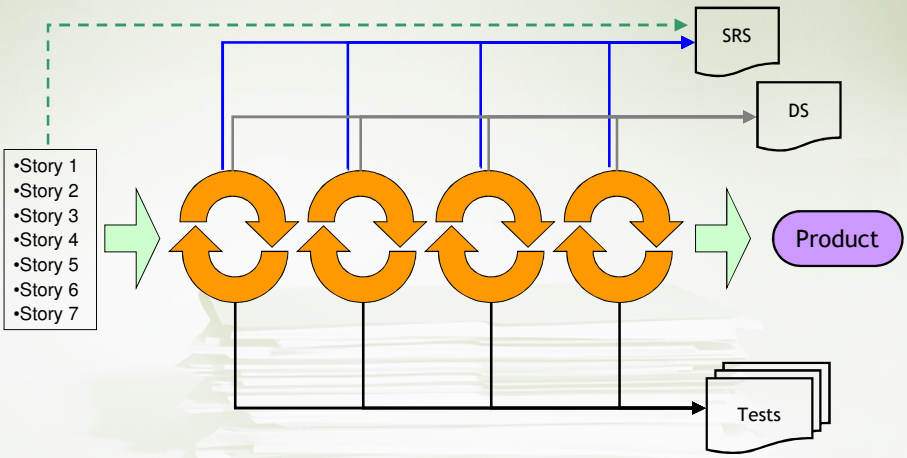


© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

36

Capture knowledge as work proceeds



© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

37

Onward to Approval – Documenting Agile Development

- *Agile vs IEC 62304: apparent contradiction?*
- *Quality – avoid bad news late*
- *Risk Management fits in well*
- *Documentation can be iterative!*
- **Agile can be clearly superior**

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

38

What ISN'T in IEC 62304?

- No prescription for how to accomplish requirements
- No specific required software life cycle
- Particular documents not specified – what to cover, not where to cover

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

39

From IEC 62304

Introduction

This standard does not prescribe a specific life cycle model. The users of this standard are responsible for selecting a life cycle model for the software project and for mapping the processes, activities, and tasks in this standard onto that model.

Annex B (informative)
Guidance on the provisions of this standard
The purpose of this standard is to provide a development process that will consistently produce high quality, safe medical device software. To accomplish this, the standard identifies the minimum activities and tasks that need to be accomplished to provide confidence that the software has been developed in a manner that is likely to produce highly reliable and safe software products. (...)

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

40

From IEC 62304

Annex B (cont.)

This standard does not require a particular software development life cycle model. However, compliance with this standard does imply dependencies between processes, because inputs of a process are generated by another process. For example, the software safety classification of the software system should be completed after the risk analysis process has established what harm could arise from failure of the software system.

Because of such logical dependencies between processes, it is easiest to describe the processes in this standard in a sequence, implying a "waterfall" or "once-through" life cycle model. However, other life cycles can also be used.

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

41

From IEC 62304

5.1.1. Software Development Plan

The manufacturer shall establish a software development plan (or plans) for conducting the activities of the software development process appropriate to the scope, magnitude, and software safety classifications of the software system to be developed. The software development life cycle model shall either be fully defined or referenced in the plan (or plans). (...)

NOTE 1. The software development life cycle model can identify different elements (processes, activities, tasks, and deliverables) for different software items according to the software safety classification of each software item of the software system.

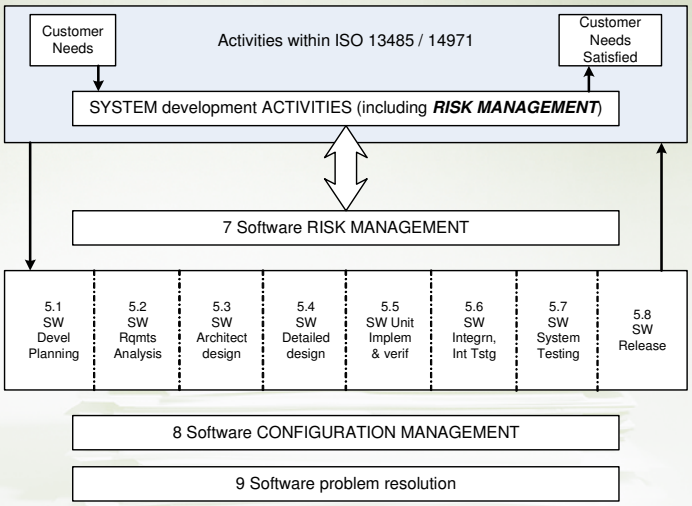
NOTE 2. These activities and tasks can overlap or interact and can be performed iteratively or recursively. **It is not the intent to imply that a specific life cycle model should be used.**

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

42

IEC 62304 Development Lifecycle

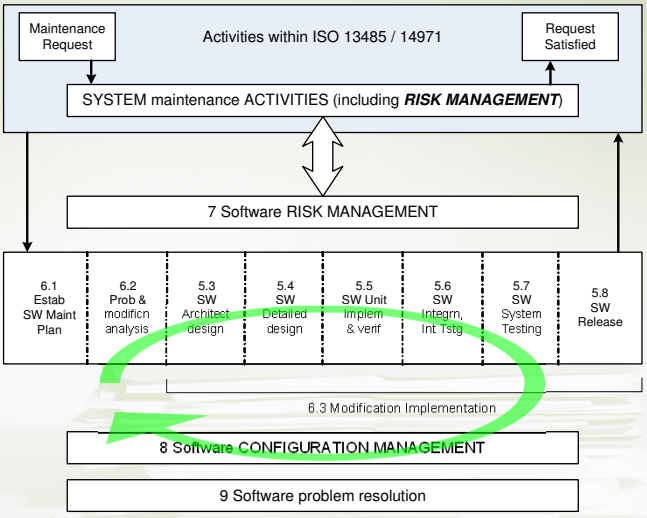


© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

43

IEC 62304 Maintenance Lifecycle

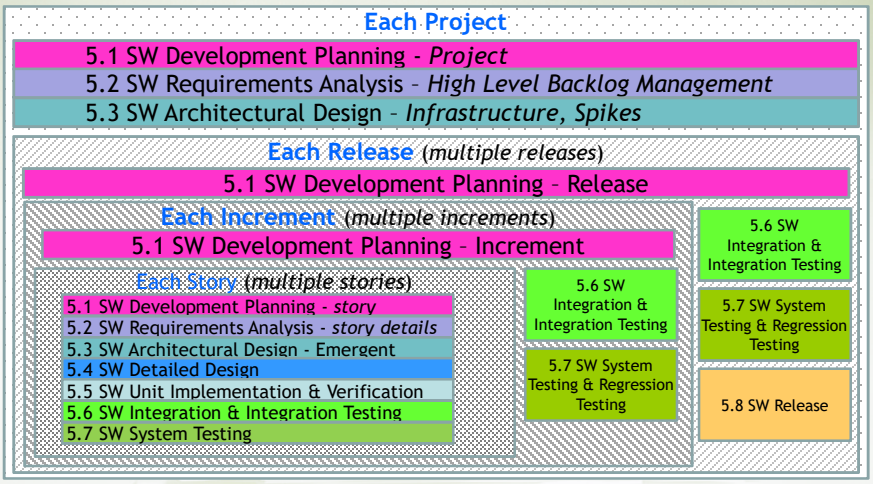


© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

44

AAMI Agile TIR: Map Agile to 62304



© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

45

Device Software Case Study

- Authors compared two projects (one Agile, one not): found that Agile gave lower cost, shorter development time, better accommodation of change, better test cases, and higher quality
- Used FDA's concept of "least burdensome approach" as part of their justification for using the Agile method
- Considered risk as integral part of development
- Iterative approach helped manage scope and limit feature creep

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

46

Device Case: Comments

Developer:
"Control what you know, don't let it control you."

Client:
"At time of commercial launch, a number of features, once thought to be essential, were not included. Some were deferred as long as three years. Nonetheless, the product was considered highly successful and trading off nice to have features for three years of sales is an easy choice."

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

47

Device Example: Reported Results

- High visibility – few surprises, able to manage and control
- Cost / duration: Agile project required 20-30% smaller team ***and*** shorter time, saved 35-50% cost, vs. non-Agile project
- Agile project gave higher quality – fewer overall defects, especially at end of project
- Agile project involved far better work-life balance and team morale (issues surfaced and managed in course of project, not saved for the end)

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

48

Agile Performance: Productivity

	"Biotech" re-implemented, as Agile (1)	"Biotech" original, as Waterfall (1)	SirsiDynix, as Agile (Scrum) (2)
Person Months	54	540	827
Lines of Java	51,000	58,000	671,688
Function Points (FP)	959	900	12,673
FP per Dev/month	17.8	2.0	15.3

1. M. Cohn, User Stories Applied for Agile Development, p. 175. Addison-Wesley, 2004 (Reported without giving company or project name, but it was a life sciences application.)
2. J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in HICSS'40, Hawaii International Conference on Software Systems, Big Island, Hawaii, describing SirsiDynix team.

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Agile Performance: Quality

Team	Defects/FP	Process
Follett Software (1)	0.0128	Agile, XP co-located
BMC Software (1)	0.048	Agile, Scrum distrib.
GMS (2)	0.22	Agile, XP for embedded
Industry Best (3)	2.0	traditional
Industry Average (3)	4.5	traditional

Co-located agile XP team achieved 100X the defect performance of the best traditional waterfall teams!

1. M. Mah, "How Agile Projects Measure Up and What This Means to You", Cutter IT Journal vol 9, no. 9, Sep 2008.
2. N. Van Schooenderwoert, "Embedded Agile Project by the Numbers With Newbies", Agile 2006 conference report.
3. Capers Jones, "Software Quality in 2002: A Survey of the State of the Art", presentation to Boston SPIN, Oct 2002

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

References

Slide	Source
4	FDA, Office of Science and Engineering Laboratories Annual Report for 2011.
6-7	AAMI/ANSI/IEC 62304:2006, "Medical Device Software - Software Life Cycle Processes", Association for the Advancement of Medical Instrumentation, July 2006.
8	http://www.agilemanifesto.org/
11-19	Shoemaker, B., and N. Van Schooenderwoert, "Jump Out of the Waterfall: Applying Lean Development Principles in Medical Device Software Development," presented at Software Design for Medical Devices, May 2010.
21	Pate, B. and M. Russell, "Agile methods for medical device software … Can it be compliant? Can it be safe?" SoftwareCPR LLC Presentation, October 2010.
29-30	Pate & Russell
31-32	Ambler, S. "Agile/Lean Documentation: Strategies for Agile Software Development", http://www.agilemodeling.com/essays/agileDocumentation.htm
36	Raymond, T., N. Van Schooenderwoert, and B. Shoemaker, "Software Quality and FDA: The Lean/Agile Way," course presented 12. May 2011.
39	Pate & Russell
40-44	IEC 62304
45	AAMI TIR45:2012 "Technical Information Report: Guidance on the use of AGILE practices in the development of medical device software", Association for the Advancement of Medical Instrumentation, August 2012.
46-48	Rasmussen, R., T. Hughes, J.R. Jenks, J. Skach, Adopting Agile in an FDA Regulated Environment, Agile 2009 Conference Proceedings, IEEE Computer Society, 2009.
49-50	Raymond, Van Schooenderwoert, and Shoemaker

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Additional References

- FDA: General Principles of Software Validation (Jan 11, 2002)
  http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085281.htm
- FDA: Premarket Submissions, Software Contained in Medical Devices (May 11, 2005)
  http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089543.htm
- FDA: Draft Guidance for Industry and Food and Drug Administration Staff - Mobile Medical Applications (July 21, 2011)
  http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm263280.htm
- IEC 62304:2006 Medical Device Software – Software Life Cycle Processes
- IEC TIR80002-1:2009, Medical device software - Part 1: Guidance on the application of ISO 14971 to medical device software
- ISO 13485:2003 (2nd ed) Medical devices – Quality management systems – Requirements for regulatory purposes
- ISO 14971:2007 (2nd ed) Medical devices – Application of risk management to medical devices

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Acknowledgement

A number of these slides were developed by Nancy Van Schooenderwoert, Lean-Agile Partners Inc., and are based on her work in coaching teams in lean methods for high-quality software and hardware development.

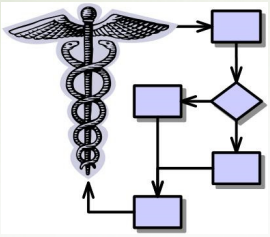
Nancy Van Schooenderwoert
Lean-Agile Partners, Inc.
162 Marrett Rd., Lexington, MA 02421
781-860-0212
NancyV@leanagilepartners.com
http://www.leanagilepartners.com

Lean-Agile Partners

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates

Contact Information



Brian Shoemaker

Principal Consultant

ShoeBar Associates

781-929-5927

bshoemaker@shoebarassoc.com

© 2012 ShoeBar Associates All Rights Reserved

ShoeBar Associates