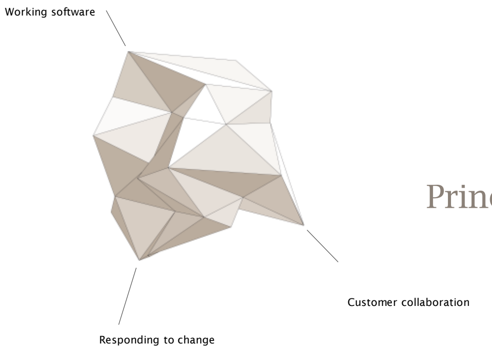
Industrial experience with Agile in high-integrity software development



Roderick Chapman
Principal Engineer, Altran UK

Can we do “High Integrity Agile” ?

- Short Answer

YES!

Can we do "High Integrity Agile" ?

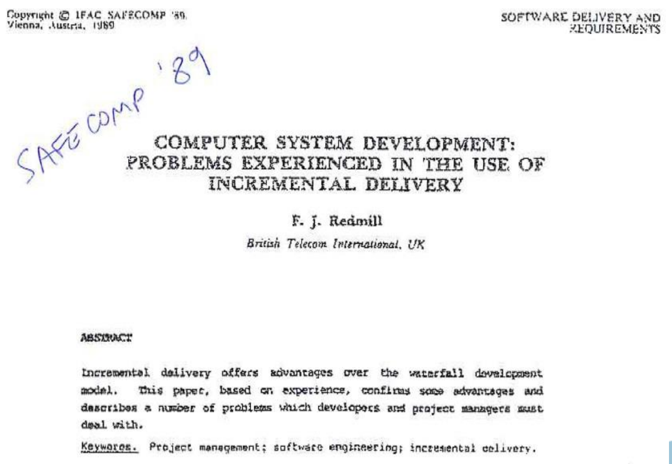
- Long Answer

Yes … but …

CONTENT

CONTENT

Some light reading…



Copyright © IFAC SAFECOMP '89, Vienna, Austria, 1989

SOFTWARE DELIVERY AND REQUIREMENTS

SAFECOMP '89

COMPUTER SYSTEM DEVELOPMENT: PROBLEMS EXPERIENCED IN THE USE OF INCREMENTAL DELIVERY

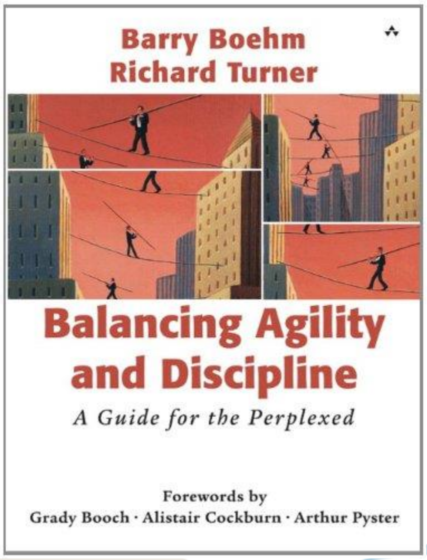
F. J. Redmill

British Telecom International, UK

ABSTRACT

Incremental delivery offers advantages over the waterfall development model. This paper, based on experience, confirms some advantages and describes a number of problems which developers and project managers must deal with.

Keywords. Project management; software engineering; incremental delivery.



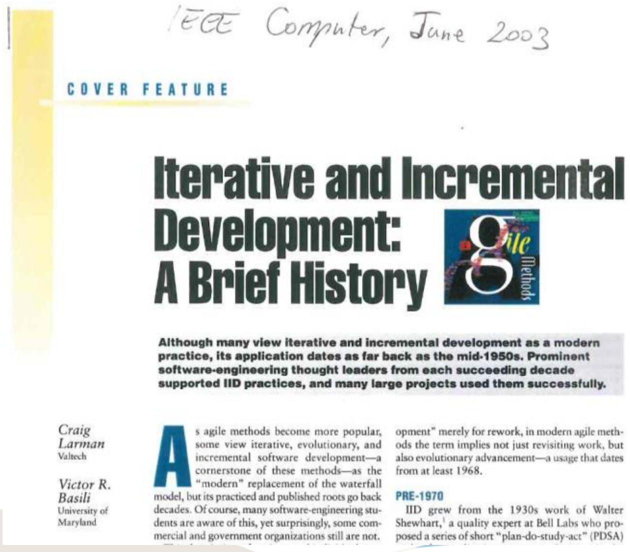
Barry Boehm
Richard Turner

Balancing Agility and Discipline

A Guide for the Perplexed

Forewords by
Grady Booch · Alistair Cockburn · Arthur Pyster

altran



IEEE Computer, June 2003

COVER FEATURE

Iterative and Incremental Development: A Brief History

Although many view iterative and incremental development as a modern practice, its application dates as far back as the mid-1950s. Prominent software-engineering thought leaders from each succeeding decade supported IID practices, and many large projects used them successfully.

Craig Larman
Valtech

Victor R. Basili
University of Maryland

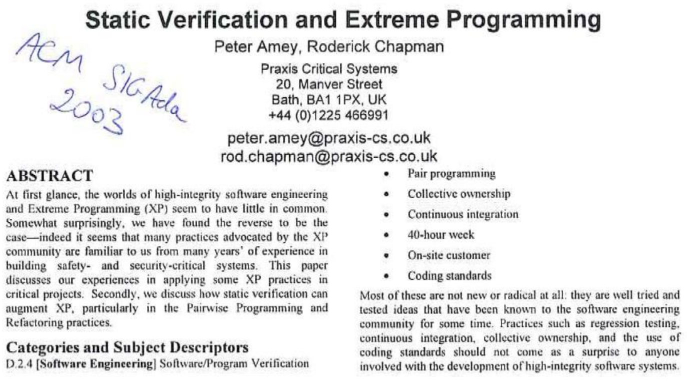
As agile methods become more popular, some view iterative, evolutionary, and incremental software development—a cornerstone of these methods—as the "modern" replacement of the waterfall model, but its practiced and published roots go back decades. Of course, many software-engineering students are aware of this, yet surprisingly, some commercial and government organizations still are not.

opment" merely for rework, in modern agile methods the term implies not just revisiting work, but also evolutionary advancement—a usage that dates from at least 1968.

PRE-1970

IID grew from the 1930s work of Walter Shewhart,[1] a quality expert at Bell Labs who proposed a series of short "plan-do-study-act" (PDSA)

altran



ACM SIGAda 2003

Static Verification and Extreme Programming

Peter Amey, Roderick Chapman

Praxis Critical Systems
20, Manver Street
Bath, BA1 1PX, UK
+44 (0)1225 466991

peter.amey@praxis-cs.co.uk
rod.chapman@praxis-cs.co.uk

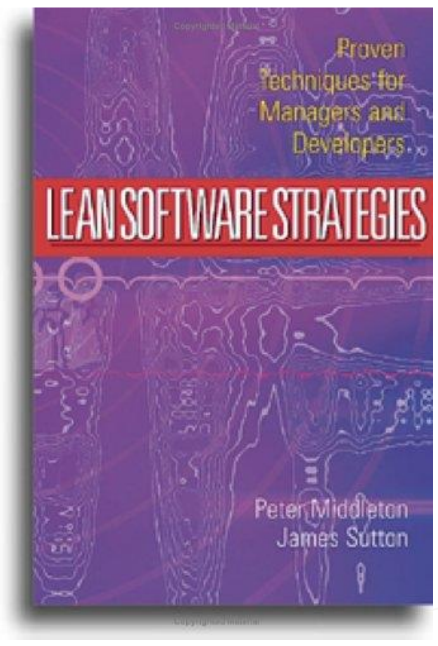
ABSTRACT

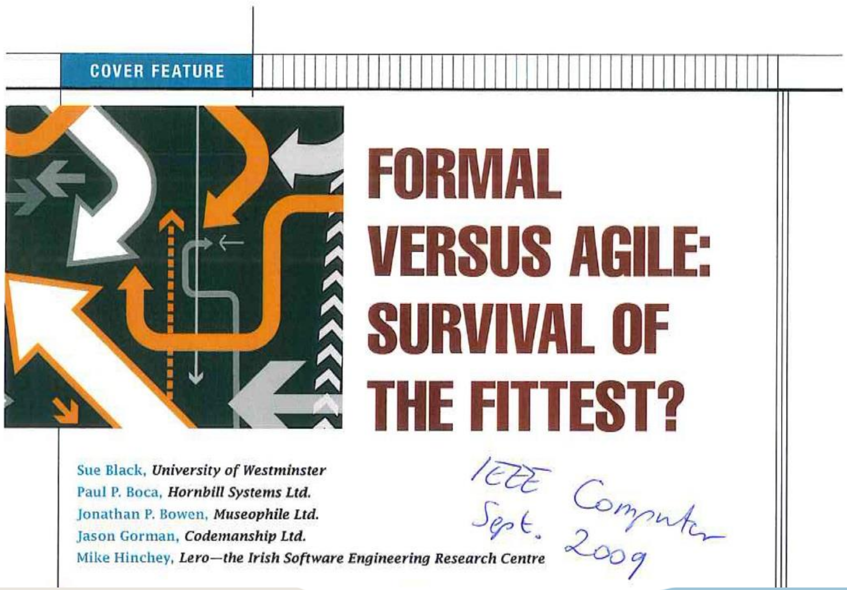
At first glance, the worlds of high-integrity software engineering and Extreme Programming (XP) seem to have little in common. Somewhat surprisingly, we have found the reverse to be the case—indeed it seems that many practices advocated by the XP community are familiar to us from many years' of experience in building safety- and security-critical systems. This paper discusses our experiences in applying some XP practices in critical projects. Secondly, we discuss how static verification can augment XP, particularly in the Pairwise Programming and Refactoring practices.

Categories and Subject Descriptors
D.2.4 [**Software Engineering**] Software/Program Verification

- Pair programming
- Collective ownership
- Continuous integration
- 40-hour week
- On-site customer
- Coding standards

Most of these are not new or radical at all: they are well tried and tested ideas that have been known to the software engineering community for some time. Practices such as regression testing, continuous integration, collective ownership, and the use of coding standards should not come as a surprise to anyone involved with the development of high-integrity software systems.



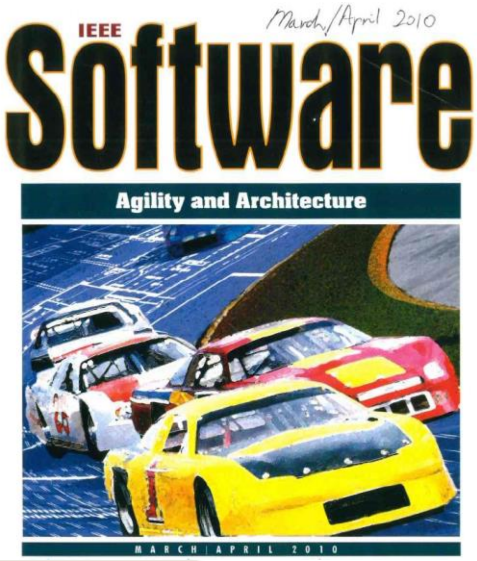


COVER FEATURE

FORMAL VERSUS AGILE: SURVIVAL OF THE FITTEST?

Sue Black, *University of Westminster*
Paul P. Boca, *Hornbill Systems Ltd.*
Jonathan P. Bowen, *Museophile Ltd.*
Jason Gorman, *Codemanship Ltd.*
Mike Hinchey, *Lero—the Irish Software Engineering Research Centre*

IEEE Computer
Sept. 2009

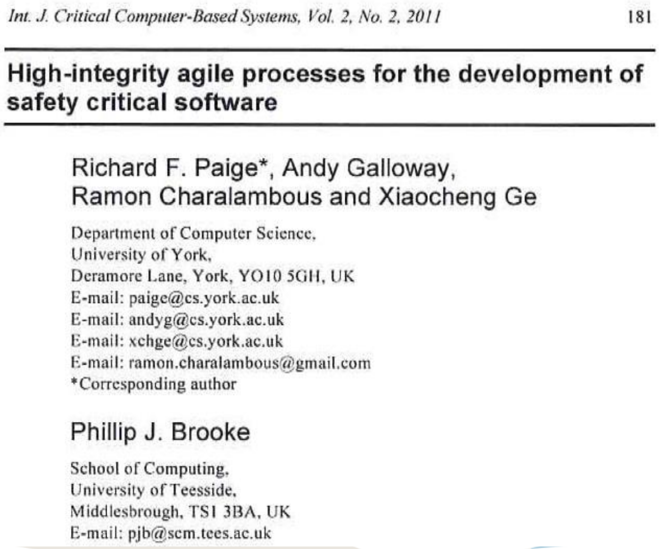


March/April 2010

IEEE Software

Agility and Architecture

MARCH | APRIL 2010


Int. J. Critical Computer-Based Systems, Vol. 2, No. 2, 2011 181

High-integrity agile processes for the development of safety critical software

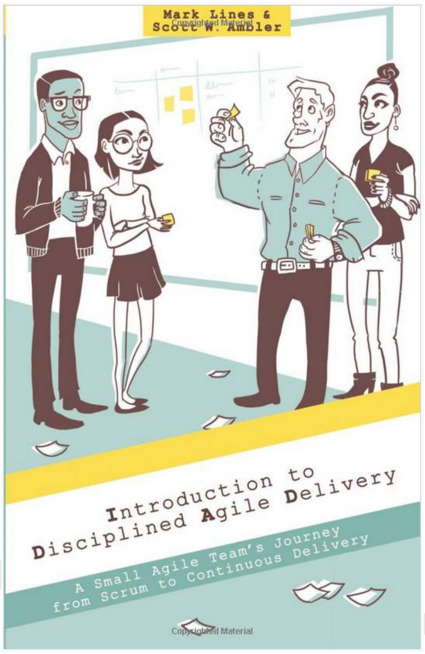
Richard F. Paige*, Andy Galloway,
Ramon Charalambous and Xiaochenq Ge

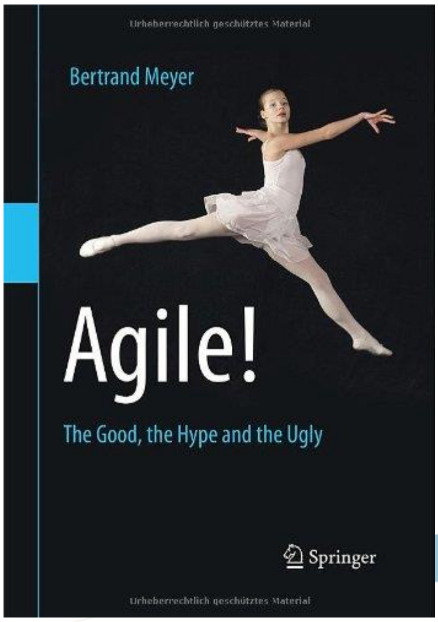
Department of Computer Science,
University of York,
Deramore Lane, York, YO10 5GH, UK
E-mail: paige@cs.york.ac.uk
E-mail: andyg@cs.york.ac.uk
E-mail: xchge@cs.york.ac.uk
E-mail: ramon.charalambous@gmail.com
*Corresponding author

Phillip J. Brooke

School of Computing,
University of Teesside,
Middlesbrough, TS1 3BA, UK
E-mail: pjb@scm.tees.ac.uk

altran


Mark Lines & Scott W. Ambler
Introduction to Disciplined Agile Delivery
A Small Agile Team's Journey from Scrum to Continuous Delivery
Copyrighted Material

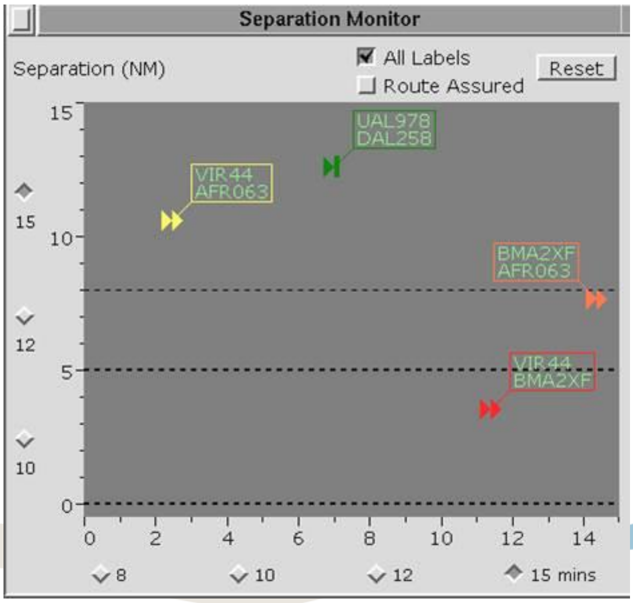

Urheberrechtlich geschütztes Material
Bertrand Meyer
Agile!
The Good, the Hype and the Ugly
Springer
Urheberrechtlich geschütztes Material

altran

…and a couple of projects…

altran

…and a couple of projects…



altran

…and reports from industry…

- Many reports of Agile being used in medical devices, under FDA regulatory regime,
- Thales Avionics (Valence, France) report use of Agile in development of avionic systems,
- And many more…

altran

CONTENT

altran

Single customer?

- Scrum viewpoint: Single “customer”, represented by “Product Owner” role…
- Really?
- What about
  - Multiple classes of “User”
  - Procurer
  - Regulator (and standard-setting body)
  - Project ISA

Regression Test and Verification

- Agile view: “Regression Test” is principal (only?) verification activity, and is *fast* and amenable to *automation*.
- “All tests pass” defines
  - When a refactoring is done
  - When a product is “good enough” to close a sprint and ship to customer.

Regression Test and Verification

- High Integrity View - No chance!

- We know “test” is utterly insufficient to claim ultra-reliability, safety or security properties.
  - Butler/Finelli and Littlewood papers from 20 years ago…
  - *Security* will *always* defy test anyway…
    - *Programming Satan’s Computer…*

altran

Regression Test and Verification

- Many more forms of verification are required by standards, for example:
  - Personal and Peer Review
  - Automated static analysis
  - Structural coverage (on target?)
  - Traceability analysis
  - Performance test
  - Penetration test etc. etc…

- We know we can do much better anyway – for example, aggressive use of *sound* static analysis.

altran

Upfront and Architecture

- Observation 1: High-Integrity systems have demanding non-functional requirements for safety, security, performance, reliability etc. etc.
- Observation 2: Our main weapon to achieve these goals is *architecture*.
- Observation 3: *You can't afford to "refactor in" these properties into a system late in the day!*

altran

Upfront and Architecture

- Conclusion: we need *just enough* upfront architecture and design to be *certain* that
  - Non-functional requirements will be met.
  - Change can be accommodated later without horrendous pain and expense.
  - We can estimate the size (and therefore price) of the first N development iteration(s).

altran

Upfront and Architecture

- But how do we know what non-functional properties are required of the architecture?
- Errm…by doing proper (Up Front) requirements engineering for safety and security properties…

User Stories and Non-Functional

- Agile-style "User Stories" provide a *sampling* of the "D, S, R space"
- There will be "gaps" between the stories…
- Guess where the safety and security problems will lie…
- Aside: how much of the MULTOS CA formal specification is devoted to *error handling??*

Agile “Simple sprint pipeline”

- Agile presumes a two-stage pipeline: one system being used by the customer and one system being developed in current sprint.
  - Delivery and deployment is assumed to be “instant”…
- Real world: no chance!
- Example: iFACTS 4-stage pipeline
  - Build N: in live operation
  - Build N+1: in NATS’ test lab
  - Build N+2: in development/test at Altran
  - Build N+3: Requirements and formal specification

altran

Iteration rate…

- How fast can we iterate?
  - Only as fast as the slowest pipeline stage…
  - Full-blown evidence (e.g. safety case production) and customer acceptance test might be *way* too slow for a standard “Agile” model…
- Idea: multiple iteration rates and deliveries:
  - Fast “minor” iteration with reduced evidence package and limited deployment.
  - Slower “major” iteration with full evidence, suitable for operational deployment.

altran

Embedded Systems Issues

- Agile depends on *plentiful* availability of "target environment" to drive a *fast* build/integration/test process.
- *Not True* for embedded systems.
  - Many projects have *no* target hardware for the majority of the time…
- Some verification activities (e.g. on-target structural coverage) are painful and slow.

Embedded Systems Issues

- Availability of target hardware for "test" can be a massive bottleneck.
- Idea: don't depend on "target hardware" and "test" so much…
- Idea: Virtualize the "deployment environment" (i.e. the target machine).
  - See the next presentation…

CONTENT

Turning the dials up…

- We've been building high-integrity software for more than 20 years…
- What have we learned that could improve an Agile approach?
- What about
  - Team and Personal Software Process (TSP/PSP)?
  - Formal Methods?
  - Correctness-by-Construction approach?
  - Lean Engineering?
  - Programming Language Design and Static Verication like SPARK?

Static Verification

- Strong Static Verification can complement “test”
  - Faster
  - “Sounder” – potentially covers *all* input data and system states.
  - Deeper – prevents and finds bugs that “test” simply cannot reach.
- So…precede “Regression Test” with “Regression Proof”
- *All* developers run SV tools *all the time,* and is not dependent on availability of target hardware, so scales well.
- Performance? iFACTS regression proof now takes *15 minutes*.

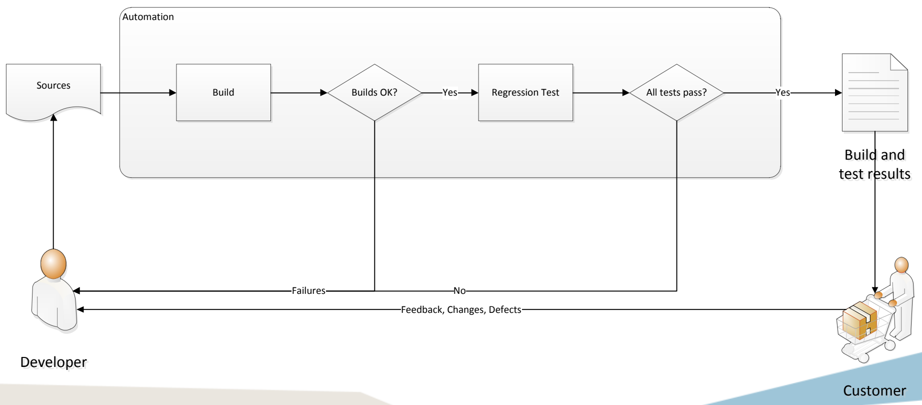
Reviewing vs pair programming

- Jury is still out on whether XP-style“pair programming” is really better…
- Conjecture:
  - Developer +
  - Strong Static Verification +
  - PSP Personal Review +
  - TSP Peer Review
- …is *much* better.
- No control experiment to confirm this…sorry!

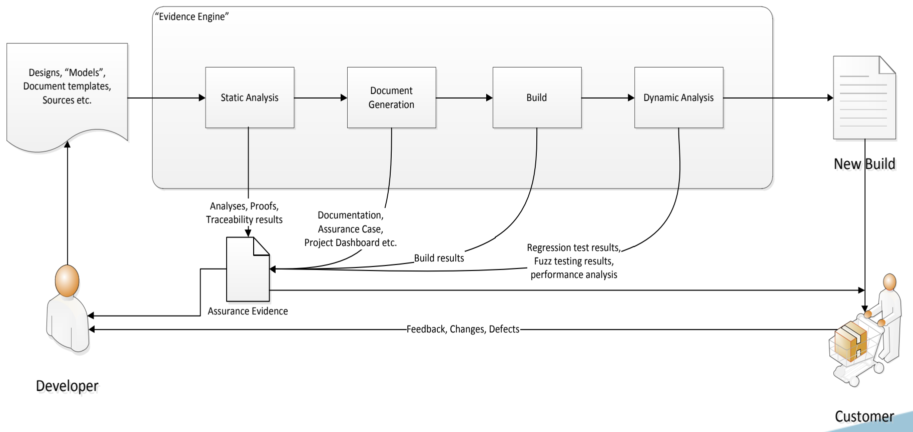
Automation, automation, automation…

- Can we automate production of other verification evidence?
  - Structural coverage
  - Traceability analysis
  - Other artefacts required by your standard or regulator?
- Yes…of course…
- So…right-to-left plan it. Work out which artefacts can be auto-generated and *plan* approach, disciplines and languages to do this in your minor or major iteration.

A naïve Agile "build/integration" system



An Agile "Evidence Engine"…


"Evidence Engine"
Designs, "Models", Document templates, Sources etc.
Static Analysis
Document Generation
Build
Dynamic Analysis
New Build
Analyses, Proofs, Traceability results
Documentation, Assurance Case, Project Dashboard etc.
Build results
Regression test results, Fuzz testing results, performance analysis
Assurance Evidence
Feedback, Changes, Defects
Developer
Customer

CONTENT

The $64M question…

- So...how much “Upfront” is “Just Right” ???
- It depends…
- …but inform this decision with solid Requirements Engineering, especially for non-functional properties.

The $64M question…

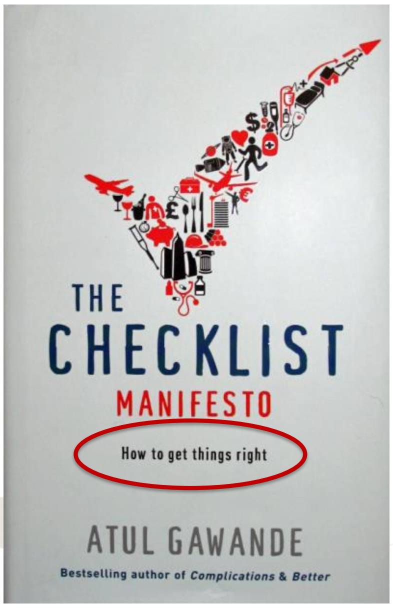
- Proposal: two-stage project
- Stage 1: Upfront work, resulting in requirements, specification (complete enough to estimate from), and enough architecture to verify NFRs and foreseeable change.
- Stage 2: Incremental/Agile build with multiple iteration rates.
- Critical: Completely different contractual and financial terms for Stages 1 and 2. (Discuss with your procurer… ☺)

CONTENT

Next Steps…

- For us: report on next project – Scrum with SPARK!
- For us: Publish…watch this space… ☺
- For you: please publish your experiences.

Homework…



THE CHECKLIST MANIFESTO

How to get things right

ATUL GAWANDE

Bestselling author of *Complications* & *Better*

altran

Questions?

altran