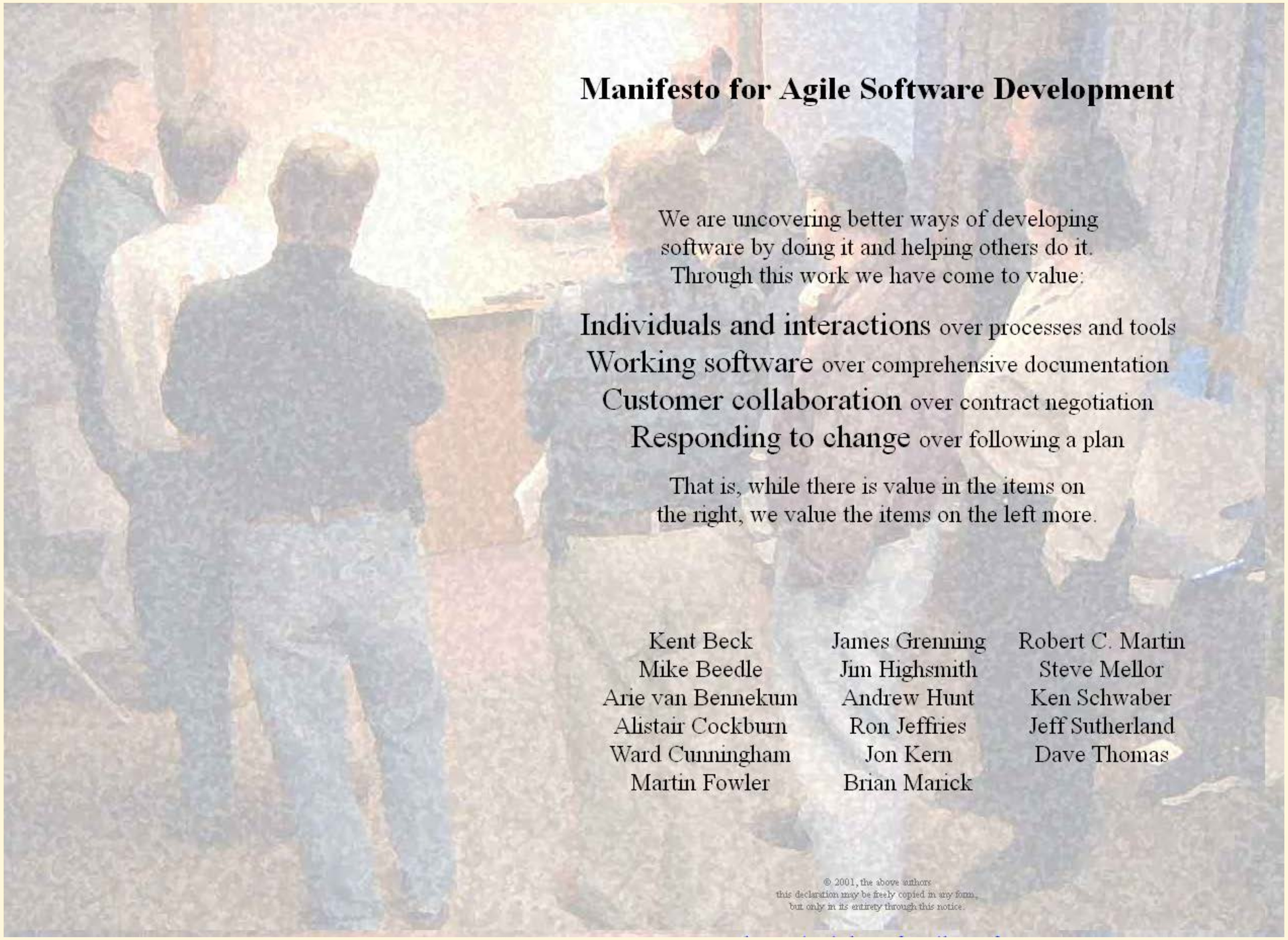
Agile Methods: The Good, the Hype and the Ugly

ACM Webinar,18 February 2015
Bertrand Meyer

Based in part on the book *Agile! The Good, the Hype and the Ugly* by Bertrand Meyer, Springer, 2014

Where it came from



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.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

ACM Highlights

- Learning Center tools for professional development: http://learning.acm.org
  - 1,400+ trusted technical books and videos by O'Reilly, Morgan Kaufmann, etc.
  - Online training toward top vendor certifications (CEH, Cisco, CISSP, CompTIA, PMI, etc
  - Learning Webinars from thought leaders and top practitioner
  - ACM Tech Packs (annotated bibliographies compiled by subject experts
  - Podcast interviews with innovators and award winners

- Popular publications:
  - Flagship *Communications of the ACM* magazine: http://cacm.acm.org/
  - *ACM Queue* magazine for practitioners: http://queue.acm.org/

- ACM Digital Library, the world's most comprehensive database of computing literature: http://dl.acm.org.

- International conferences that draw leading experts on a broad spectrum of computing topics: http://www.acm.org/conferences.

- Prestigious awards, including the ACM A.M. Turing and Infosys: http://awards.acm.org/

- And much more…http://www.acm.org.

Talk Back

- Use Twitter widget to Tweet your favorite quotes from today's presentation with hashtag **#ACMWebinarAgile**
- Submit questions and comments via Twitter to @acmeducation – we're reading them!
- Use the Facebook and other sharing toolsin the bottom panel to share this presentation with friends and colleagues

Eiffel Software

ETH zürich

Chair of
Software Engineering

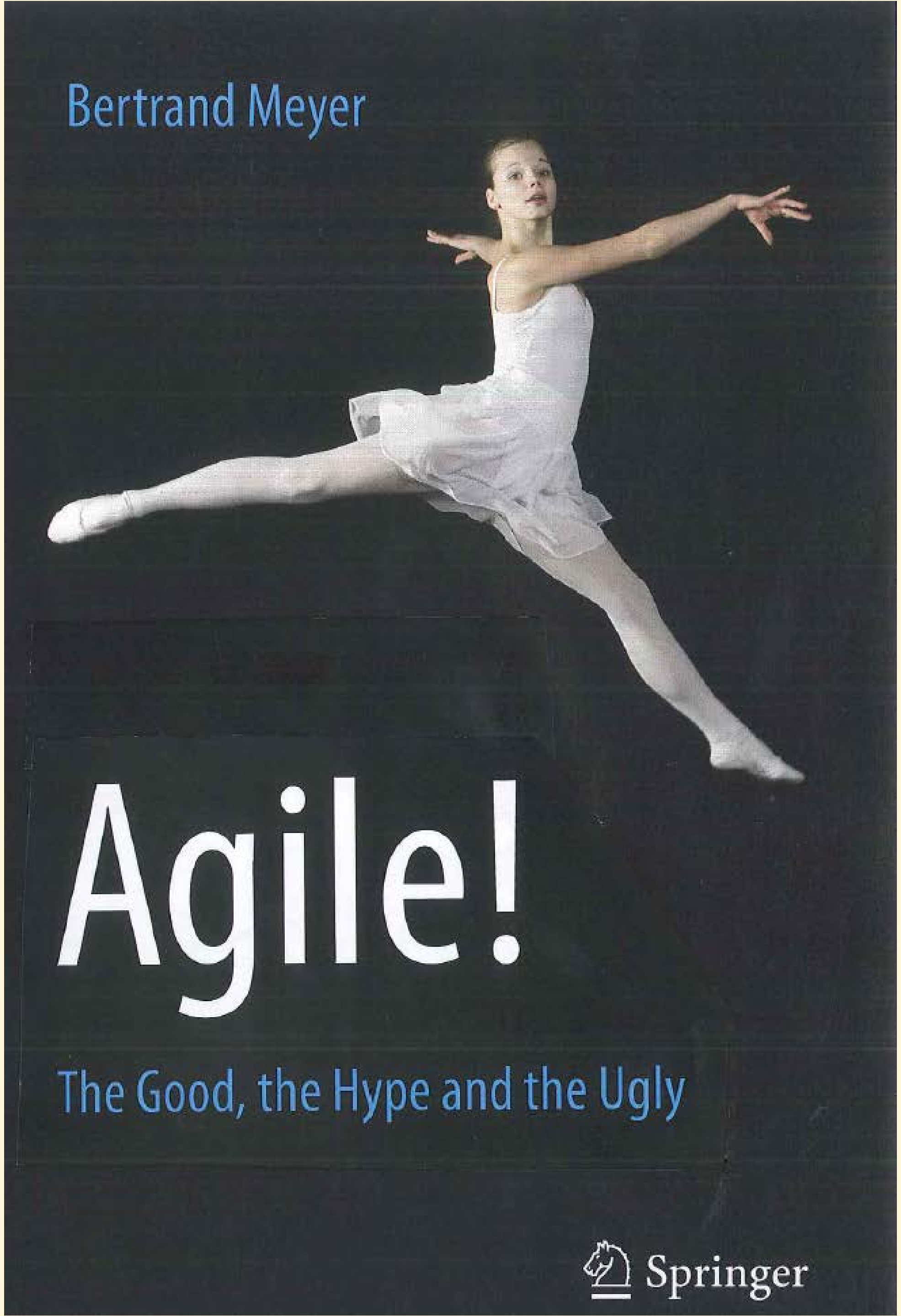
Agile Methods:
The Good, the Hype and the Ugly

ACM Webinar,18 February 2015

Bertrand Meyer

Based in part on the book *Agile! The Good, the Hype and the Ugly*
by Bertrand Meyer, Springer, 2014

innopolis



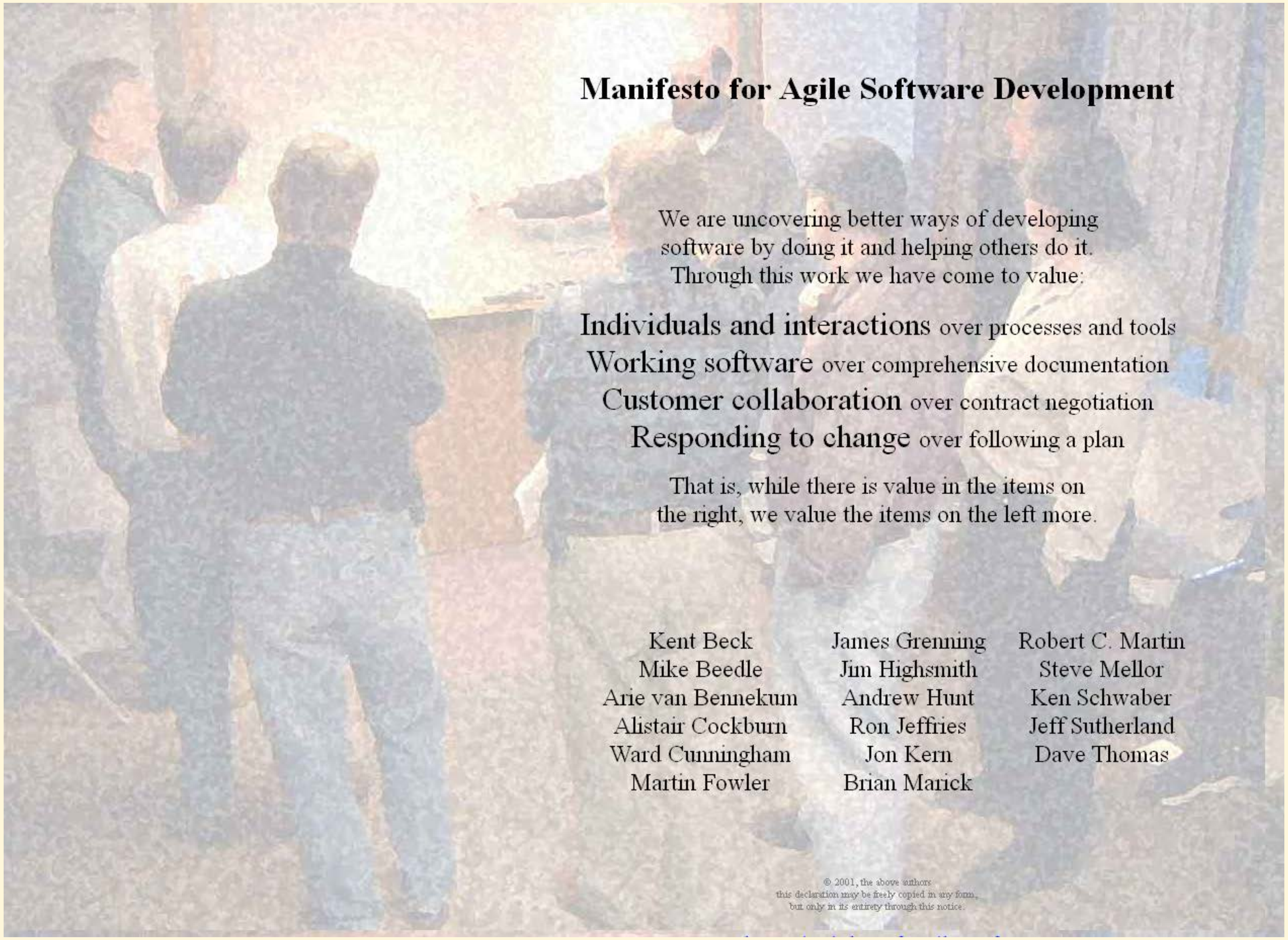
Bertrand Meyer

Agile!

The Good, the Hype and the Ugly

Springer

Agile manifesto



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.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Agile methods

XP	Kent Beck
Lean	Mary Poppendieck
Crystal	Alistair Cockburn
Scrum	Schwaber & Sutherland

Topics

1 Key agile concepts

2 Assessment

Supplementary material: pitfalls in assessing agile methods

1

Key agile concepts

Twelve principles

Source: Agile manifesto

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity — the art of maximizing the amount of work not done — is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Redundancy

Redundancy

What about testing?

Practice

Assertion

Assertion

Assertion

Wrong

Practice

Finishing a design

It seems that the sole purpose of the work of engineers, designers, and calculators is to polish and smooth out, lighten this seam, balance that wing until it is no longer noticed, until it is no longer a wing attached to a fuselage, but a form fully unfolded, finally freed from the ore, a sort of mysteriously joined whole, and of the same quality as a poem.

It seems that perfection is reached, not when there is nothing more to add, but when there is no longer anything to remove.

(Antoine de Saint-Exupéry,
Terre des Hommes, 1937)

Steve Jobs, 1998

That's been one of my mantras — focus and simplicity. Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple.

But it's worth it in the end because once you get there, you can move mountains.

Towards a better definition

Values

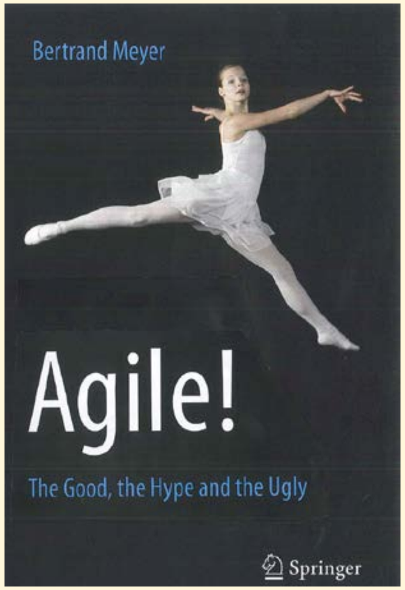
Principles

Practices:

- Managerial
- Technical

Artifacts

Agile methods



My view: agile values

- **A** New, reduced role for manager
- **B** No “Big Upfront” steps
- **C** Iterative development
- **D** Limited, negotiated scope
- **E** Focus on quality, achieved through testing

My view: agile principles

Organizational

- 1 Put the customer at the center
- 2 Accept change
- 3 Let the team self-organize
- 4 Maintain a sustainable pace
- 5 Produce minimal software:
  - 5.1 Produce minimal functionality
  - 5.2 Produce only the product requested
  - 5.3 Develop only code and tests

Technical

- 6 Develop iteratively
  - 6.1 Produce frequent working iterations
  - 6.2 Freeze requirements during iterations
- 7 Treat tests as a key resource:
  - 7.1 Do not start any new development until all tests pass
  - 7.2 Test first
- 8 Express requirements through scenarios

Standard form for user stories

Scrum

"As a <*user_or_role*>
I want <*business_functionality*>
so that <*business_justification*>"

Example:

*"**As a** customer,*
***I want to** see a list of my recent orders,*
***so that** I can track my purchases with a company."*

User stories (my view)

User stories requirement elicitation but not a fundamental requirement technique. They cannot define the requirements:

- Not abstract enough
- Too specific
- Describe current processes
- Do not support evolution

User stories are to requirements what tests are to software specification

Major application: for validating requirements

Additive and multiplicative complexity

Adding features

Source: Zave*

Historically, developers of telecommunication software have had trouble managing feature interactions, causing runaway complexity, bugs, cost and schedule overruns, and unfortunate user experiences. Other areas are also facing the problem.

Consider "busy treatments" in telephony, such as call forwarding, callee interruption, delayed retry and voice mail. Suppose that we have a tool for specifying and composing such features. Features can fail to come into action, even if their individual description says they should, when they start interacting with other features :

- *Bob has enabled the "call-forwarding" feature, to Carol. Carol has "do-not-disturb". Alice calls Bob: the call is forwarded to Carol; her phone rings.*
- *Alice calls a sales group. A sales-group feature forwards the call to the salesperson on duty, Bob. His cellphone is off, so the caller gets Bob's personal Voice Mail message. It would be better to reactivate the sales-group feature to find another salesperson.*

*Abridged, see full citation in my book or original at http://public.research.att.com/~pamela/faq.html

User stories (imagined)

(#1) **As an** executive, **I want** a redirection option **so that** if my phone is busy the call is redirected to my assistant

…

(#5) As a system configurator, I want to be able to specify various priorities for "busy" actions

..

(#12) As a salesperson, I want to make sure that if a prospect calls while I am in a conversation, the conversation is interrupted **so that** I can take the call immediately

…

(#25) As a considerate correspondent, I want to make sure that if a call comes while my phone is busy I get to the option of calling back as soon as the current call is over

6.2 Iterativeness: freeze requirements during iteration

Scrum

The **closed-window rule**: during a sprint, no one may add functionality

(or: the sprint is cancelled)

Dual development

Early on: build infrastructure (horizontal, lasagne)

Later: produce releases

Negotiated scope contract

XP

Source: Beck 05

"Write contracts for software development that fix time, costs, and quality but call for an ongoing negotiation of the precise scope of the system. Reduce risk by signing a sequence of short contracts instead of one long one.

You can move in the direction of negotiated scope. Big, long contracts can be split in half or thirds, with the optional part to be exercised only if both parties agree. Contracts with high costs for change requests can be written with less scope fixed up front and lower costs for changes"

2
The Ugly,
The Hype,
The Good
& The Brilliant:
An Assessment

The ugly

- Rejection of upfront tasks
- Particularly: no upfront requirements
- User stories as a replacement for abstract requirements
- Tests as a replacement for specifications
- Feature-based development & ignorance of dependencies
- Embedded customer
- Coach & method keeper (e.g. Scrum Master) as a separate role
- Test-driven development
- Dismissal of traditional manager tasks
- Dismissal of auxiliary products and non-shippable artifacts
- Dismissal of a priori concern for extendibility
- Dismissal of a priori concern for reusability
- Dismissal of a priori architecture work

The indifferent

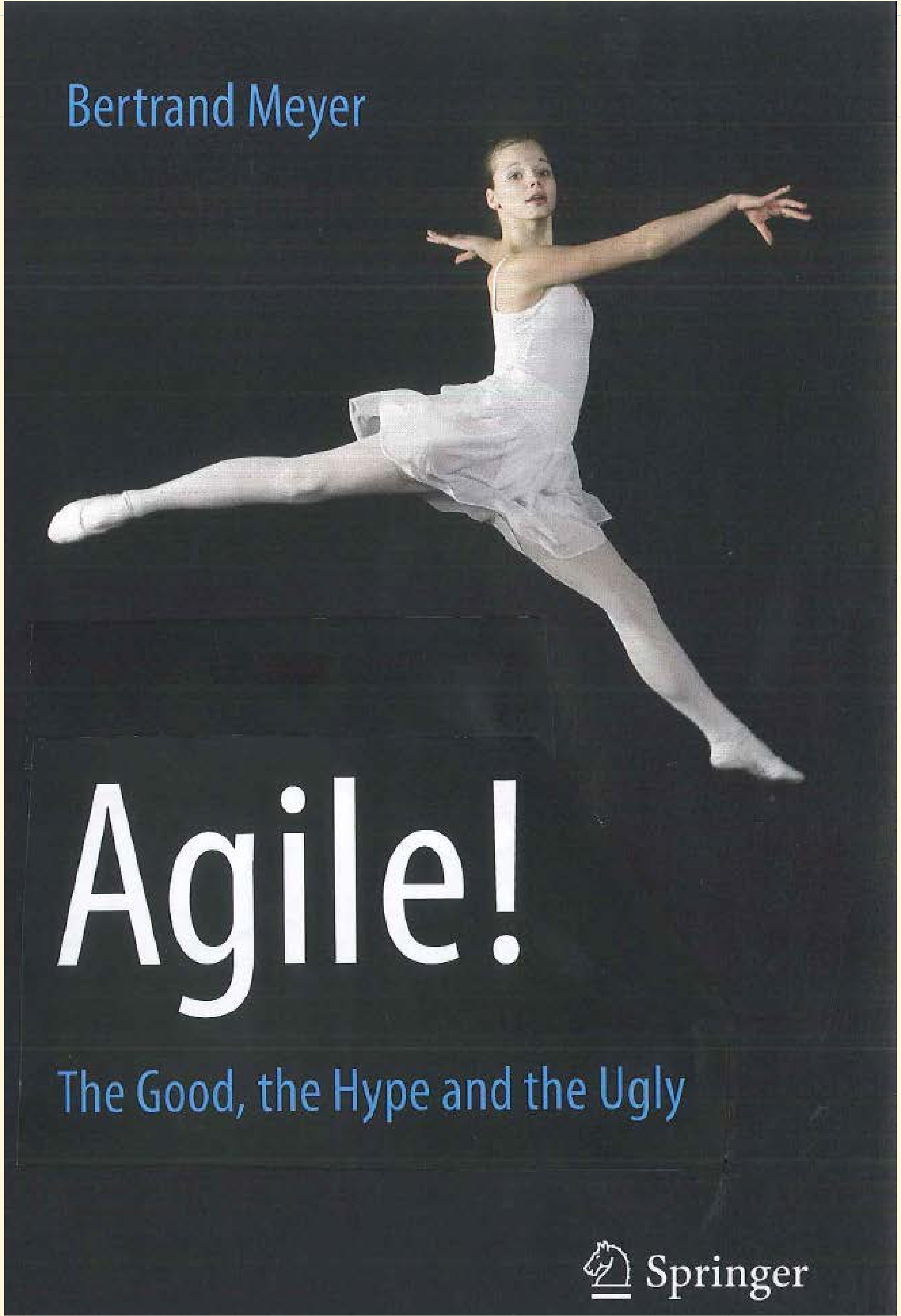
- Pair programming
- Open-space working arrangements
- Self-organizing teams
- Maintaining a sustainable pace
- Producing minimal functionality
- Planning game, planning poker
- Cross-functional teams

The good

- Acceptance of change
- Frequent iterations
- Emphasis on working code
- Tests as one of the key resources of the project
- Constant test regression analysis
- No branching
- Product (but not user stories!) burndown chart
- Daily meeting

The brilliant

- Short iterations
- Closed-window rule
- Refactoring (but not as a substitute for design)
- Associating a test with every piece of functionality
- Continuous integration


Bertrand Meyer

Agile!

The Good, the Hype and the Ugly

Springer

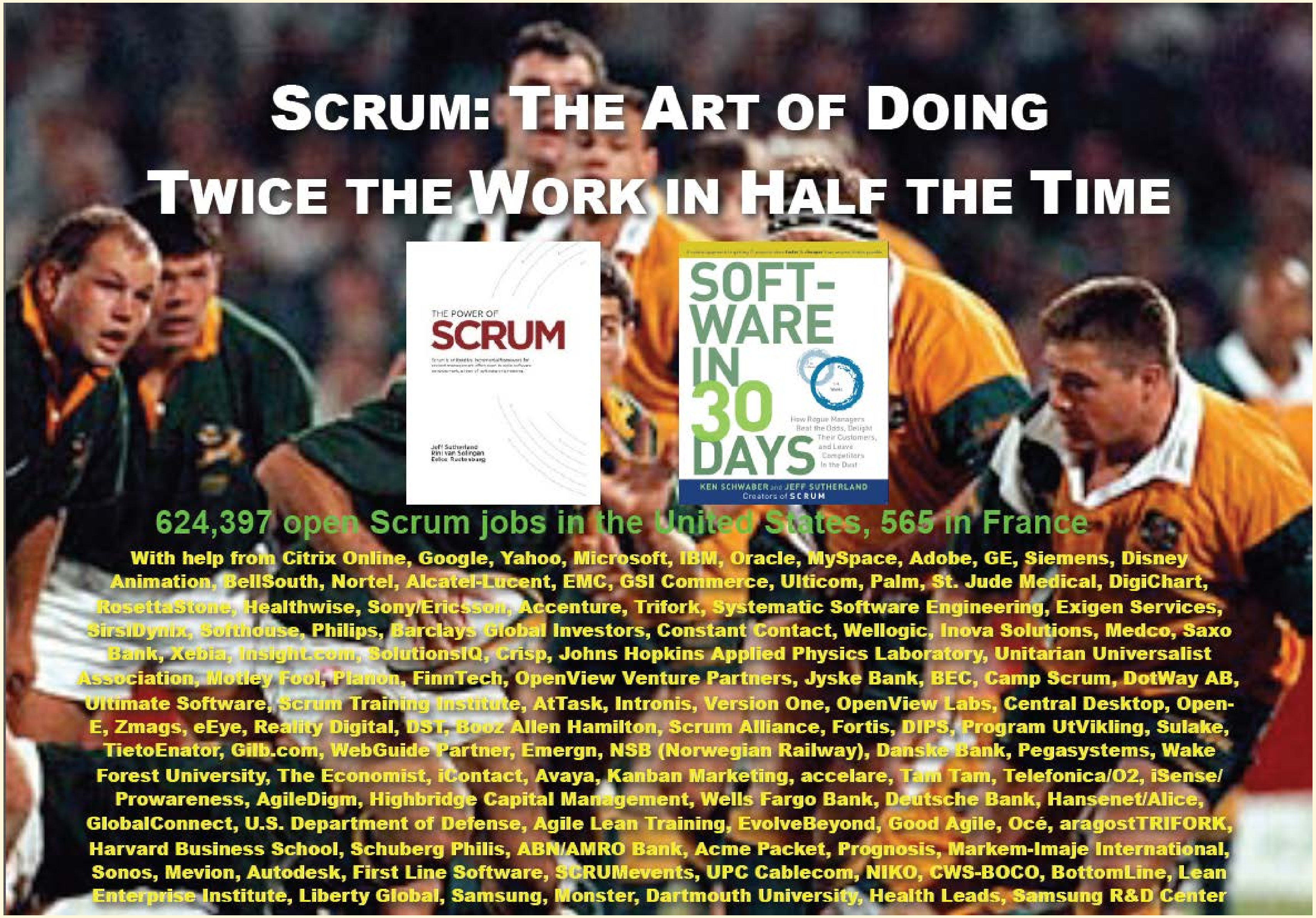
3

Assessment pitfalls
(supplementary material)

Rhetorical devices

- **Unverifiable claims**
- **Proof by anecdote**
- **Slander by association**
- **Intimidation**
- **All-or-nothing**
- **Cover-your-behind**

Unverifiable claims



SCRUM: THE ART OF DOING TWICE THE WORK IN HALF THE TIME

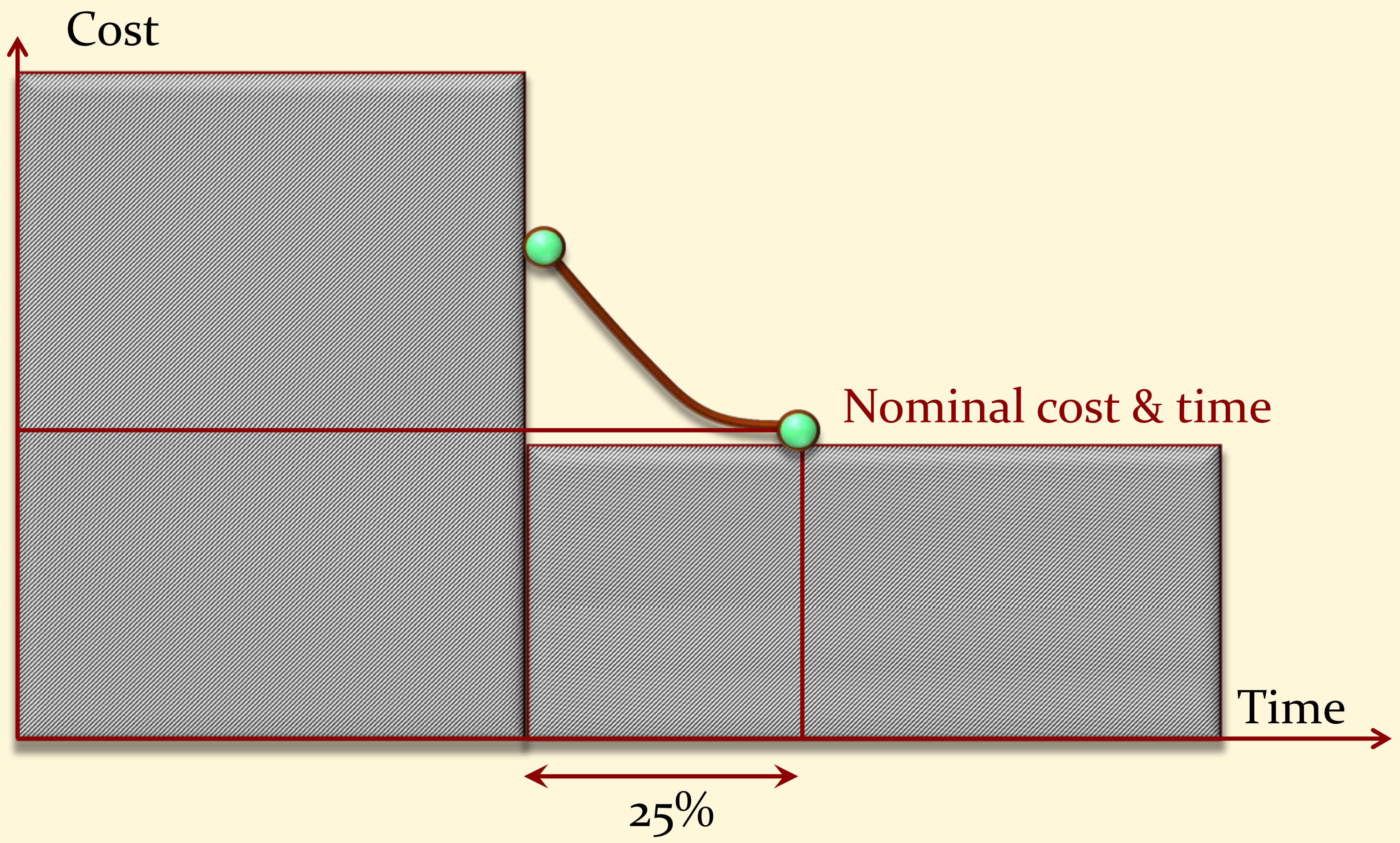
THE POWER OF SCRUM
Jeff Sutherland
Rini van Solingen
Eelco Rustenburg
SOFT-WARE IN 30 DAYS
How Rogue Managers Beat the Odds, Delight Their Customers, and Leave Competitors in the Dust
KEN SCHWABER and JEFF SUTHERLAND
Creators of SCRUM

624,397 open Scrum jobs in the United States, 565 in France

With help from Citrix Online, Google, Yahoo, Microsoft, IBM, Oracle, MySpace, Adobe, GE, Siemens, Disney Animation, BellSouth, Nortel, Alcatel-Lucent, EMC, GSI Commerce, Ulticom, Palm, St. Jude Medical, DigiChart, RosettaStone, Healthwise, Sony/Ericsson, Accenture, Trifork, Systematic Software Engineering, Exigen Services, SirsiDynix, Softhouse, Philips, Barclays Global Investors, Constant Contact, Wellogic, Inova Solutions, Medco, Saxo Bank, Xebia, Insight.com, SolutionsIQ, Crisp, Johns Hopkins Applied Physics Laboratory, Unitarian Universalist Association, Motley Fool, Planon, FinnTech, OpenView Venture Partners, Jyske Bank, BEC, Camp Scrum, DotWay AB, Ultimate Software, Scrum Training Institute, AtTask, Intronis, Version One, OpenView Labs, Central Desktop, Open-E, Zmags, eEye, Reality Digital, DST, Booz Allen Hamilton, Scrum Alliance, Fortis, DIPS, Program UtVikling, Sulake, TietoEnator, Gilb.com, WebGuide Partner, Emergn, NSB (Norwegian Railway), Danske Bank, Pegasystems, Wake Forest University, The Economist, iContact, Avaya, Kanban Marketing, accelare, Tam Tam, Telefonica/O2, iSense/Prowareness, AgileDigm, Highbridge Capital Management, Wells Fargo Bank, Deutsche Bank, Hansenet/Alice, GlobalConnect, U.S. Department of Defense, Agile Lean Training, EvolveBeyond, Good Agile, Océ, aragostTRIFORK, Harvard Business School, Schuberg Philis, ABN/AMRO Bank, Acme Packet, Prognosis, Markem-Imaje International, Sonos, Mevion, Autodesk, First Line Software, SCRUMevents, UPC Cablecom, NIKO, CWS-BOCO, BottomLine, Lean Enterprise Institute, Liberty Global, Samsung, Monster, Dartmouth University, Health Leads, Samsung R&D Center

Reminder: software engineering has laws

Example: Boehm, McConnell, Putnam, Capers Jones…



Slander by association: Schwaber & Sutherland

- *Although the predictive, or waterfall, process is in trouble, many people and organizations continue to try to make it work.*

and later in the same paragraph:

- *[A customer was using] services from PricewaterhouseCoopers (PWC). The PWC approach was predictive, or waterfall.*

The book's index entry for "*Predictive process*" reads "*See Waterfall* "

Catastrophism: Schwaber & Sutherland

"You have been ill served by the software industry for 40 years—not purposely, but inextricably. We want to restore the partnership."

Also: every agile author cites the Standish report

CYA: the "although" style of agile explanations

Schwaber: ***Although*** *project development teams are on their own, they are* ***not*** *uncontrolled.*

Cohn: *Self-organizing teams* ***are not*** *free from management control. Management chooses what product to build or often who will work on their project,* ***but*** *the teams are* ***nonetheless*** *self-organizing.* ***Neither*** *are they* ***free from*** *influence. …* ***That being said****, the fewer constraints or controls put on a team, the better.*

A ***common misconception*** *about agile project management approaches is that because of this reliance on self-organizing teams, there is little or no role for leaders of agile teams.* ***Nothing could be further from the truth.*** *In The Biology of Business, Philip Anderson refutes this* ***mistaken assumption****:*

"Self-organization ***does not mean*** *that workers instead of managers engineer an organization design. It* ***does not mean*** *letting people do whatever they want to do. It means that management commits to guiding the evolution of behaviors that emerge from the interaction of independent agents* ***instead of*** *specifying in advance what effective behavior is."*

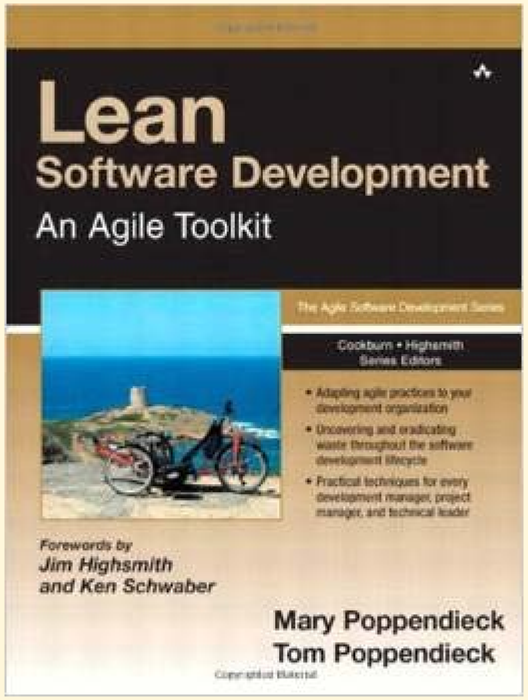
Self-organizing teams ***are not free*** *from management control. Management chooses for them what product to build or often chooses who will work on their project, but they are* ***nonetheless*** *self-organizing.* ***Neither*** *are they free from influence. […]* ***That being said****, the fewer constraints or controls put on a team, the better.*

CYA: Poppendieck

Final chapter, "Instructions and Warranty" (chapter 8, pages 179-186)

Look for the balance point of the lean principles:

- ***Eliminate waste*** [chapter 3] *does not mean throw away all documentation*
- ***Amplify learning*** [chapter 2] *does not mean keep on changing your mind*
- ***Decide as late as possible*** [chapter 3] *does not mean procrastinate*
- (etc.)


Lean
Software Development
An Agile Toolkit
The Agile Software Development Series
Cockburn • Highsmith Series Editors
• Adapting agile practices to your development organization
• Uncovering and eradicating waste throughout the software development lifecycle
• Practical techniques for every development manager, project manager, and technical leader
Forewords by Jim Highsmith and Ken Schwaber
Mary Poppendieck
Tom Poppendieck

Beck, first edition

To some folks, XP seems like just good common sense. So why the "extreme" in the name? XP takes commonsense principles and practices to extreme levels:

- *If code reviews are good, we'll review code all the time (pair programming)*
- *If testing is good, everybody will test all the time (unit testing), even the customers (functional testing)*
- *If design is good, we'll make it part of everybody's daily business (refactoring)*
- …

CYA: Beck, second edition

There are better ways and worse ways to develop software. Good teams are more alike than they are different. No matter how good or bad your team you can always improve.

ACM: The Learning Continues…

Questions about this webcast? learning@acm.org

ACM Learning Webinars (on-demand archive):
http://learning.acm.org/webinar

ACM Learning Center: http://learning.acm.org

ACM SIGSOFT: http://www.sigsoft.org/

ACM Queue: http://queue.acm.org/