Los Siete Mejores Hábitos Para Implementar Agile con Éxito

Alan W. Brown
IBM Rational CTO for Europe

alanbrown@es.ibm.com

March 2012

Topics

- Agile in context
- Seven habits of successful agile adoption
  - Be explicit about you agile goals
  - Understand the dimensions of scale up/out
  - Use measures to govern behaviour
  - Focus early on quality as a team issue
  - Re-skill your project/program planners
  - Grow with a clear adoption plan
  - Think globally, act locally!

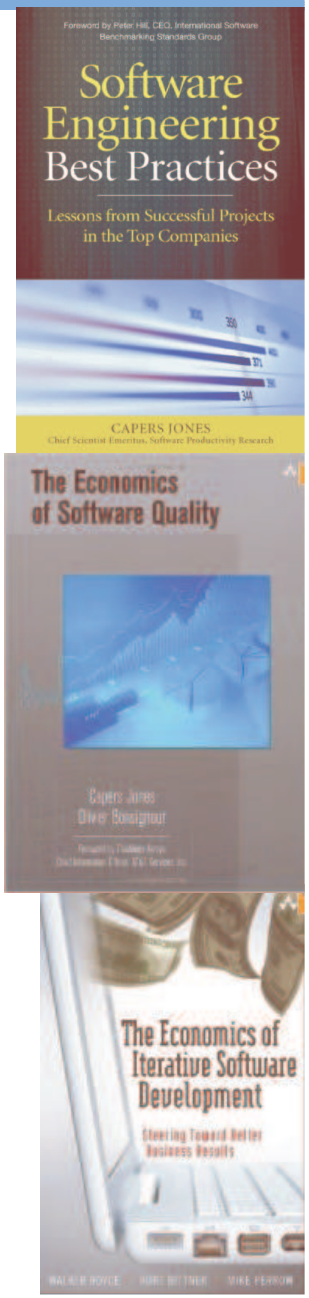
Why Do Software Project Fail?

Understanding the software engineering lifecycle

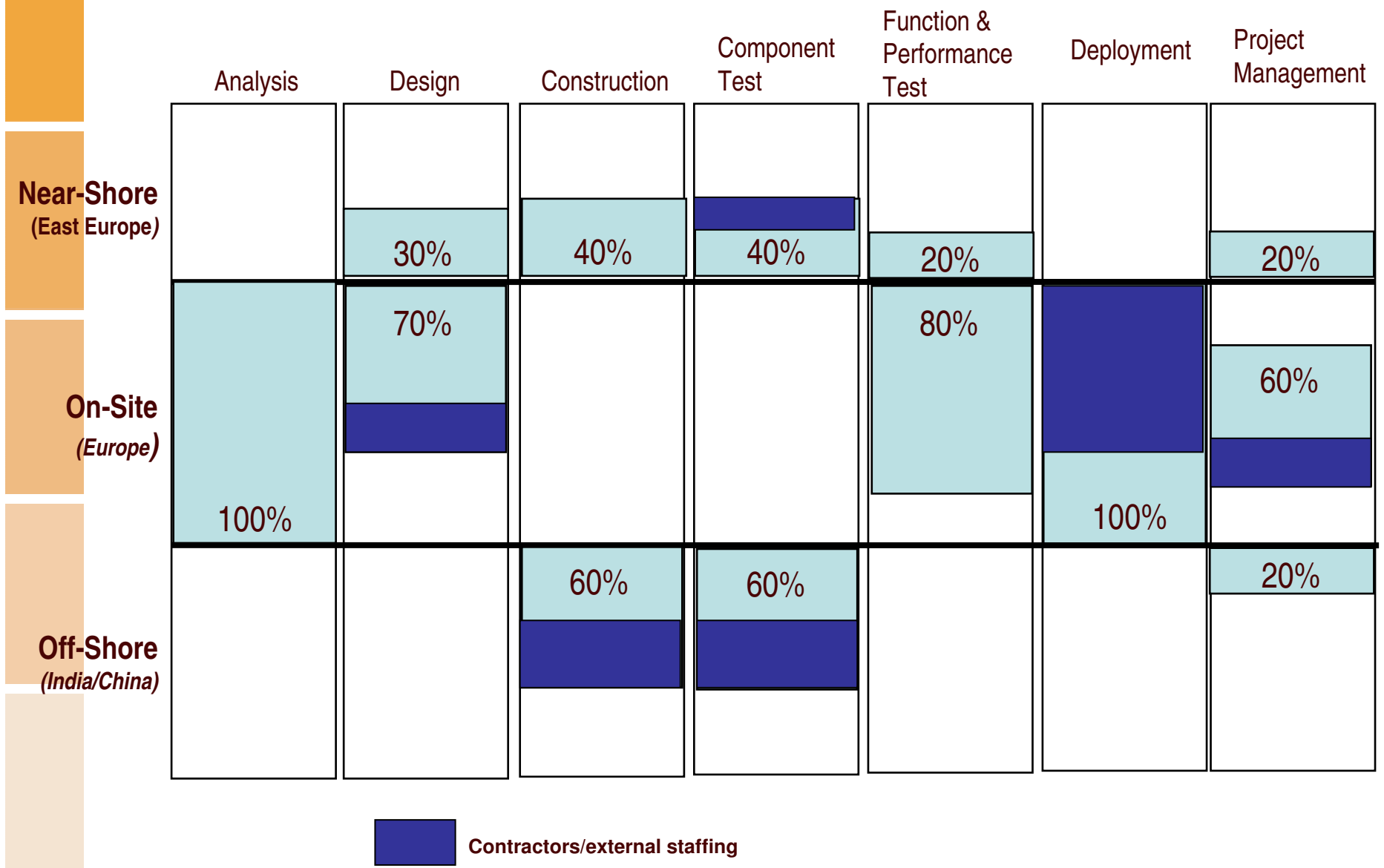
1. Unstable, changing requirements (95%)
2. Inadequate quality control and poor quality measures (90%)
3. Inadequate progress tracking (85%)
4. Inadequate cost and schedule estimating (80%)
5. False promises by marketing and sales personnel (80%)
6. Rejecting good schedule estimates for arbitrary dates (75%)
7. Informal, unstructured development (70%)
8. Inexperienced clients who can't articulate requirements (60%)
9. Inexperienced project managers (50%)
10. Inadequate tools for quality/analysis, lack of inspections (55%)
11. Reusing assets filled with bugs (30%)
12. Inexperienced, unqualified software engineering teams (20%)

From Caper Jones

[Visualization of change and complexity]



Global Delivery of Software: An Example



Summary: Challenges to effective software delivery today

Complexity Challenges

- More granular service functionality in composite business applications
- Large number of projects and assets including custom, outsourced and packaged

Team Challenges

- Geographically dispersed teams that often include business partners
- Effective cross-organizational visibility and synchronization, sharing becomes an imperative

Process Challenges

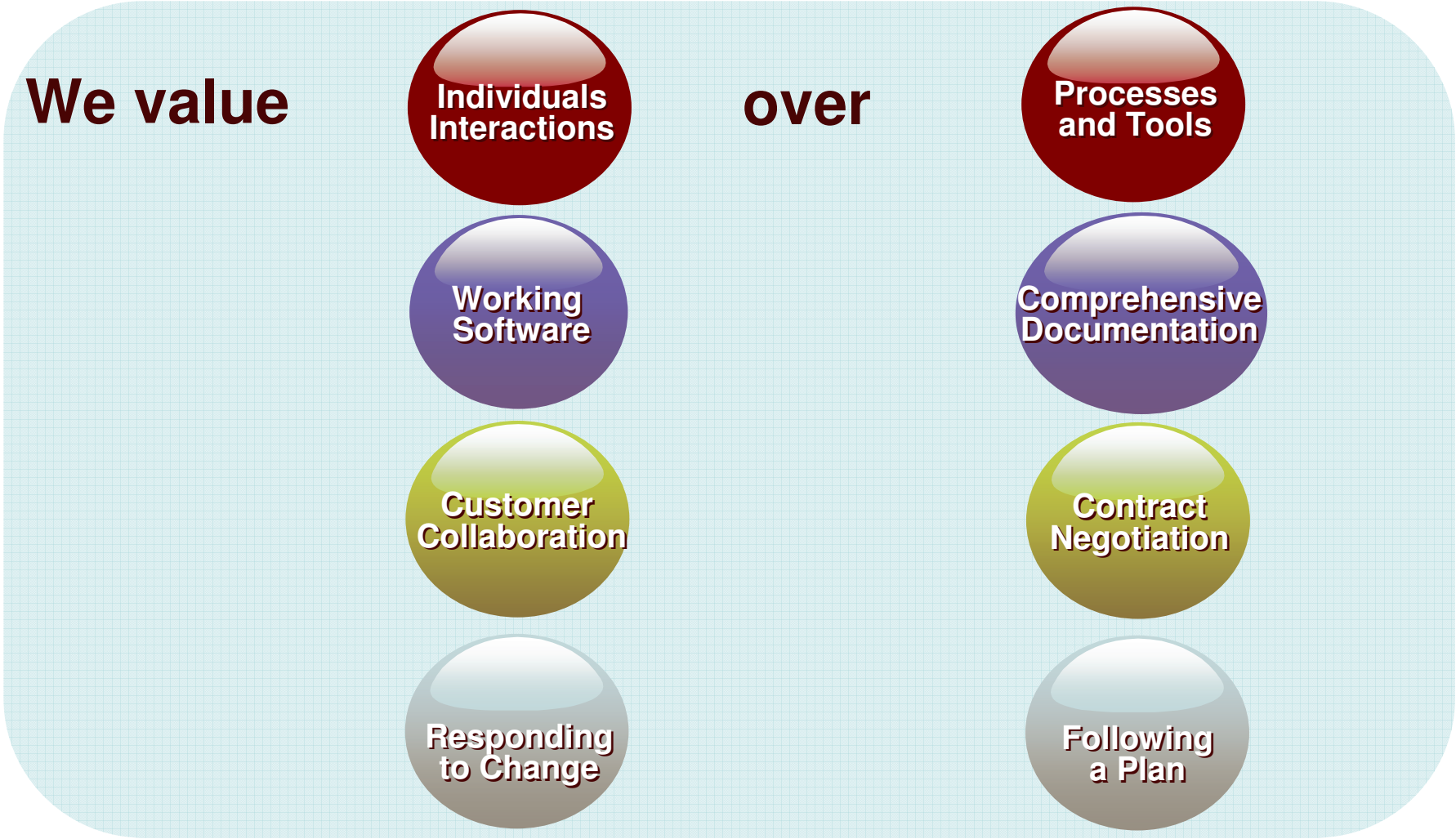
- Need for market experimentation
- Blind adherence to process insensitive to potential business trade-offs
- Need for agility *at scale*

Tools Challenges

- Lack of standards impacts ability to collaborate, automate and report across teams and assumptions
- Frequent asset updates and changing interdependencies

How do I understand this new world to gain advantage?

Agile Software Delivery and Values

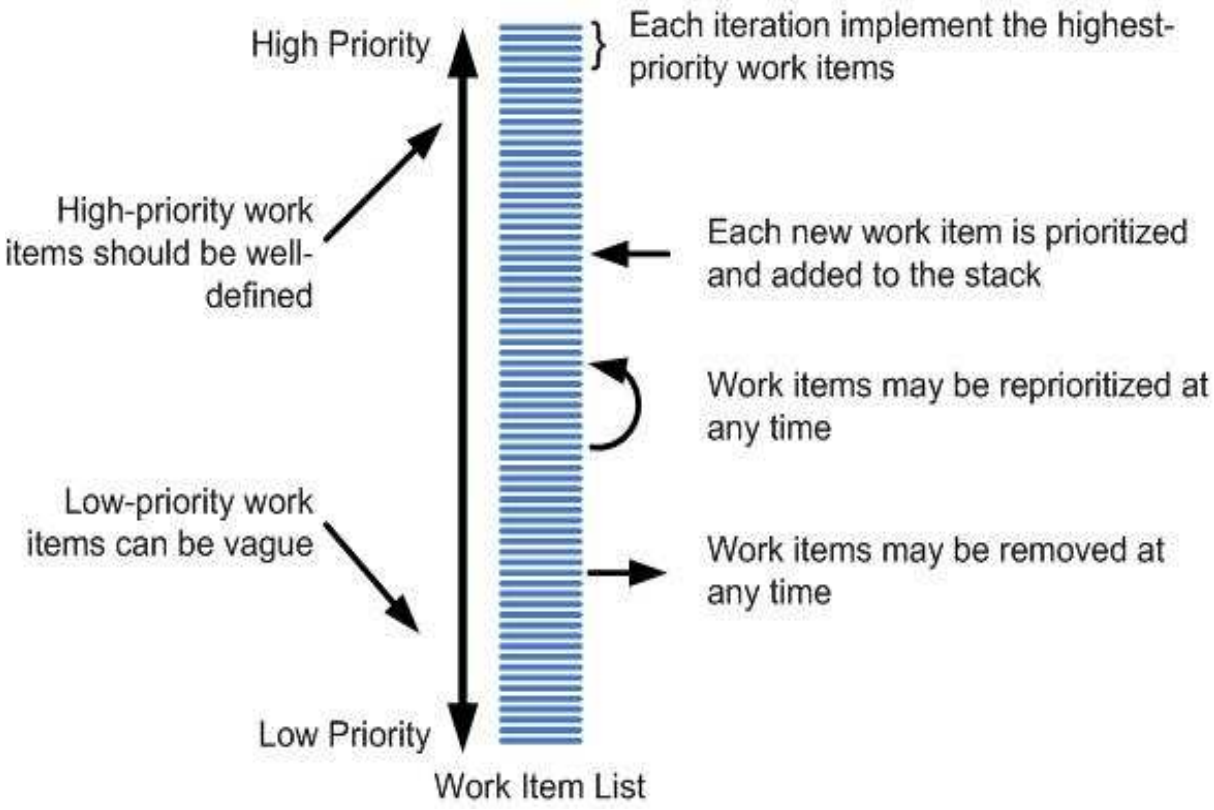


While there is value in the items on the right, we value the items on the left more.

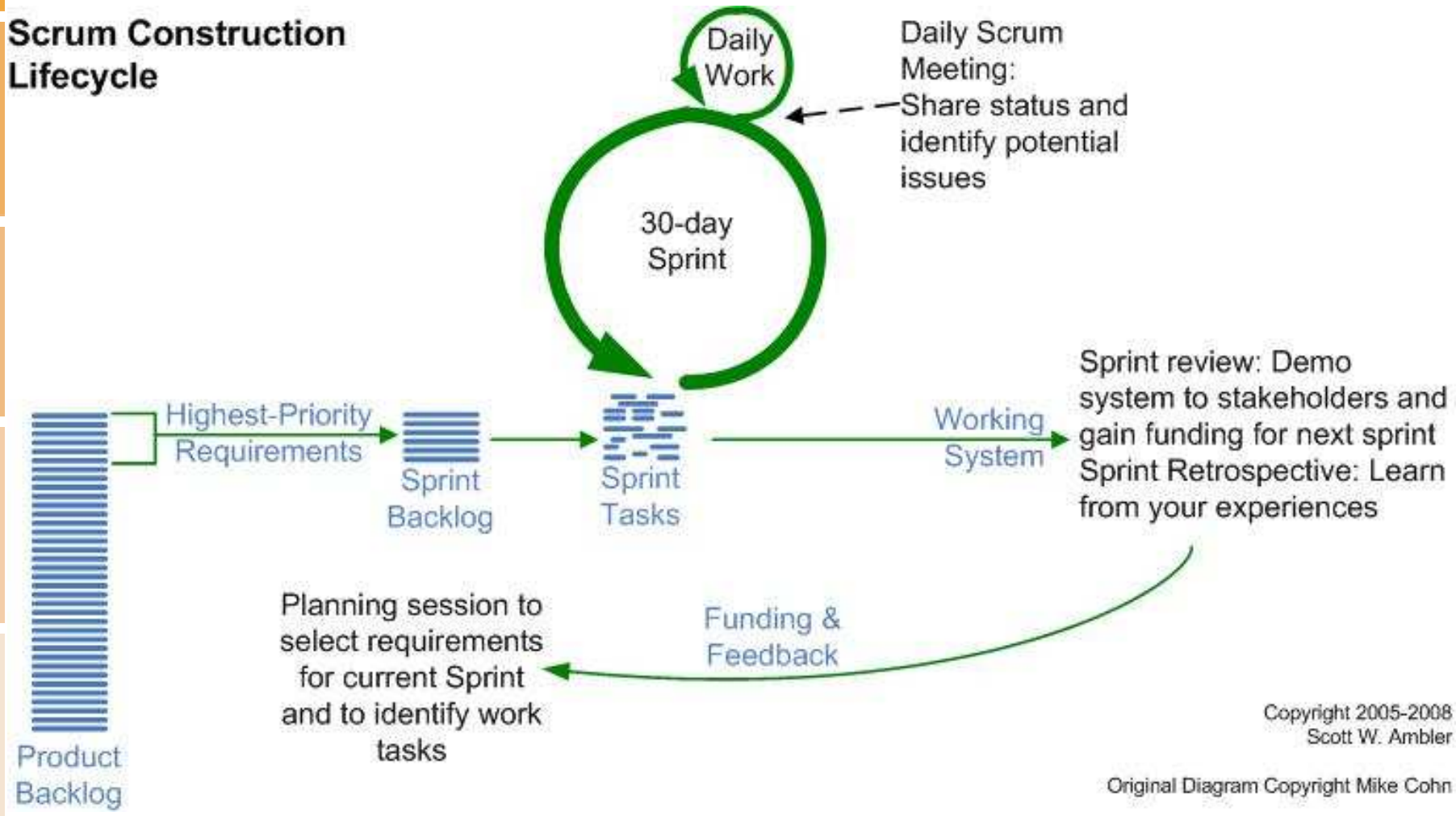
Source: www.agilemanifesto.org

Mainstream Agile Practices

- Regular Deployment of Working Software
- Non-Solo Development
- Refactoring
- Continuous Integration
- Configuration Management
- Test Driven Development (TDD)
- Agile Testing
- Agile Documentation



The Agile Construction Lifecycle



How can I be more agile?

1. Produce working software on a more **regular basis**.
2. Do **continuous** regression testing, and better yet take a Test-Driven Development (TDD) approach.
3. Work **closely** with stakeholders, ideally on a daily basis.
4. Increase your scope for self-organizing, and organize the team within an **appropriate** governance framework.
5. **Regularly** reflect, and **measure** on how the team works together, and act to improve in a **timely** manner.

Topics

- Agile in context
- **Seven habits of successful agile adoption**
  - **Be explicit about you agile goals**
  - **Understand the dimensions of scale up/out**
  - **Use measures to govern behaviour**
  - **Focus early on quality as a team issue**
  - **Re-skill your project/program planners**
  - **Grow with a clear adoption plan**
  - **Think globally, act locally!**

Agile Scaling Factors

Team size

Under 10 developers ⟷ 1000's of developers

Compliance requirement

Low risk ⟷ Critical, audited

Geographical distribution

Co-located ⟷ Global

Domain Complexity

Straight -forward ⟷ Intricate, emerging

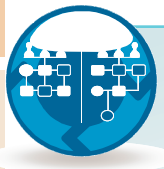
Disciplined Agile Delivery

Enterprise discipline

Project focus ⟷ Enterprise focus

Organization distribution (outsourcing, partnerships)

Collaborative ⟷ Contractual



Organizational complexity

Flexible ⟷ Rigid

Technical complexity

Homogenous ⟷ Heterogeneous, legacy

Measures Govern Behaviour

While many methods of measurement exist at various levels of depth, this forms the minimum necessary and sufficient set of measurement areas to assess the fundamental health and status of a live software project. Other measurements can aid in determining root causes, but do not determine ultimate performance, thus serve a secondary role.

But What Should We Measure?

An Example Set of Candidate Agile Metrics…..lots of possibilities!

Executive Dashboard

Project Health

- Defect Backlog
- Defect Density
- Defect Repair Latency
- Build Health
- Project Velocity
- Staffing Actuals
- Process Timeliness
- Milestone Status
- Severity Analysis
- Security Vulnerabilities
- Static Code Analysis
- Requirements Met
- IPD Timeliness

Customer Quality

- Transactional Survey
- PMR / Call Rates
- Critical Situations
- Cost of Support
- Installability
- Enhancement SLA
- Useability
- Consumability
- Perceived Performance
- Scalability
- Integrations with other products
- User Experience / Doc
- Time to Resolution

Development Quality

- Defect Backlog
- Test Escapes
- Functional Test Trends
- Critical Situations
- System Test Trends
- S-Curve Progress
- Automation Percentage
- Customer Testcases
- Consumability Scorecard
- Defect Latency
- Quality Plan Commitments
- Test Coverage

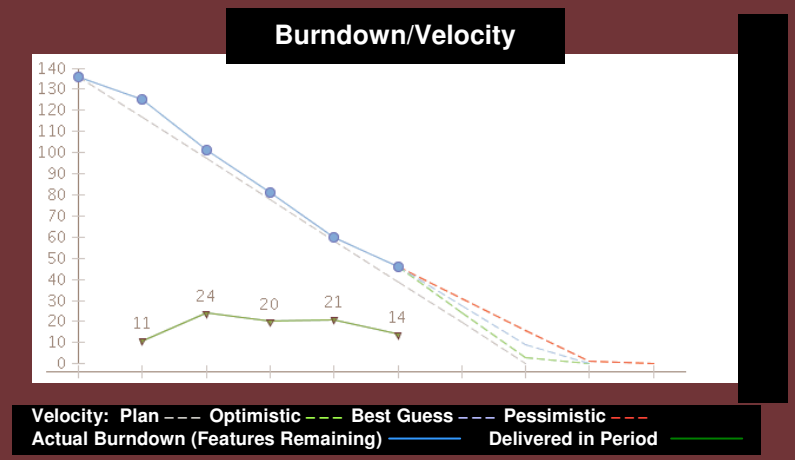
Strategic Health

- Sales Plays
- Partner Enablement
- Support Enablement
- Technical Enablement
- Sales Enablement
- Localization
- MCIF Index
- Competition
- Integrated into Story
- Green Threads
- LCM
- Pipeline / Multiplier
- Revenue

Practices

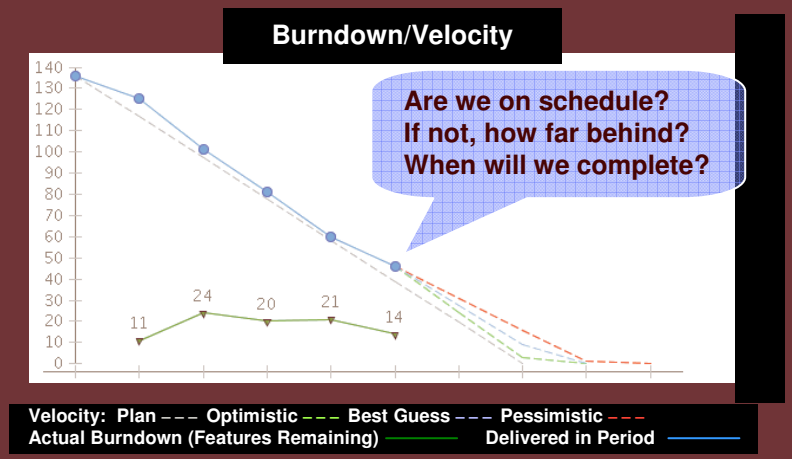
Evolutionary Architecture
Vulnerability Assessment
Concurrent Testing
Test Driven Development
Whole Team
Requirements Management
Team Change Management

Key Project Performance Metrics: Agile View



This is a sample view. Metrics can take different forms. The intent is ensure that the charts address core management concerns and associated questions.

Agile Performance Metrics: Core Answers



Are scope changes overwhelming our ability to deliver?

Are our estimates good?
Do we have enough staff ?
What will it take to complete?

Do we have enough test coverage and execution?
Is the quality of our deliveries good enough?

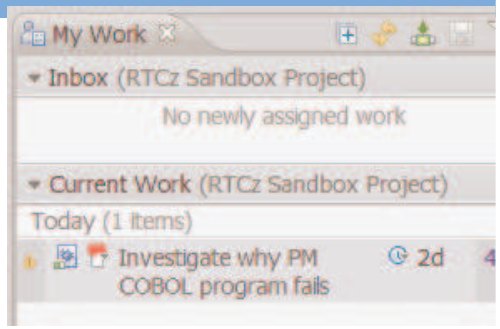
Here are the key management questions answered by each chart. An inability to answer any of these questions serves as a source of fundamental risk.

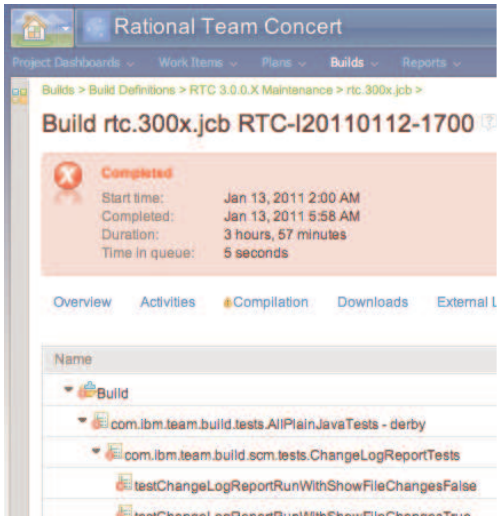
A New Approach to Quality with Agility

- Traditional approaches to quality have a heavy focus on testing at multiple levels, often with separate test teams
- Agile approaches introduce many challenges:
  - Focus on rapid, constant change
  - Time-to-market often dominant
  - Less focus on architecture modeliing
  - Less detailed documentation
  - Assumes collocated teams with direct communication paths
  - Lack of consistency across several small, independent self-organizing teams can have major system test impacts
- Substantial changes may be needed to address software quality in agile projects
- Agile Quality is a team issue, addressed “early and often”

A day in the life... of a 'Pig'

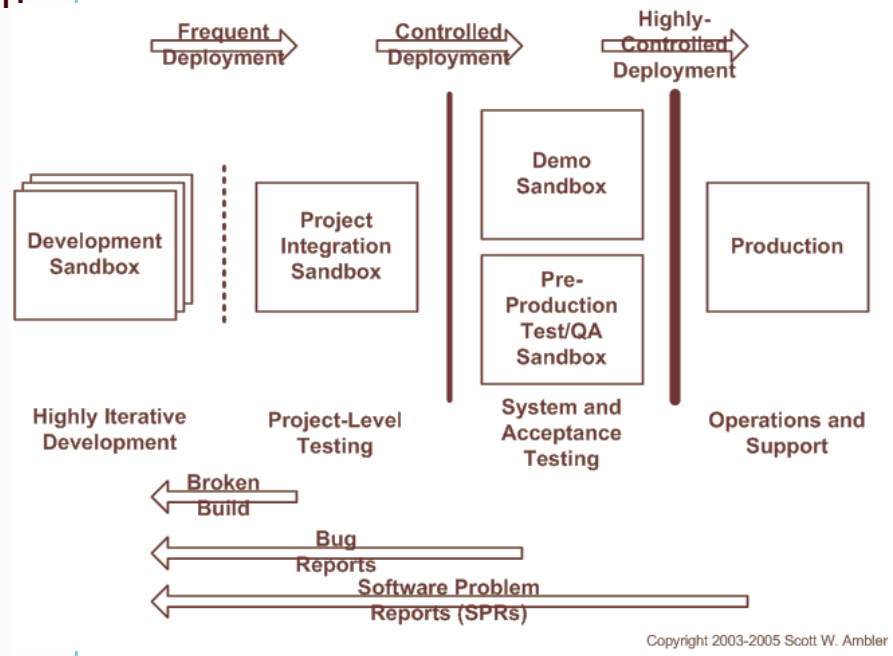
- **Always starts with a daily scrum**
- **Think!... Document ... Write JUnit testcases... Code... Test**
  1. Check *My Work*
  2. API First; *improve the collaboration with your clients*
  3. Test Driven Development (TDD); *solidify your code*
  4. Update work items; *let other members know what you've done*
- **Deliver code to the Team Stream**
  - Test team integration; *now your component is not alone*
- **Deliver code to the Integration Stream**
  - Daily & Weekly builds
  - Test project integration; *we now have a product*
  - Control JUnit testcases execution; *check the overall quality*
- **Recurrent activities**
  - Actively participate in design meetings; *across Scrum teams*
  - Regular JUnit jam sessions; *leverage the know-how within the teams*
  - Scrum of Scrums meetings when appropriate; *keep the rhythm*





Continuous Integration

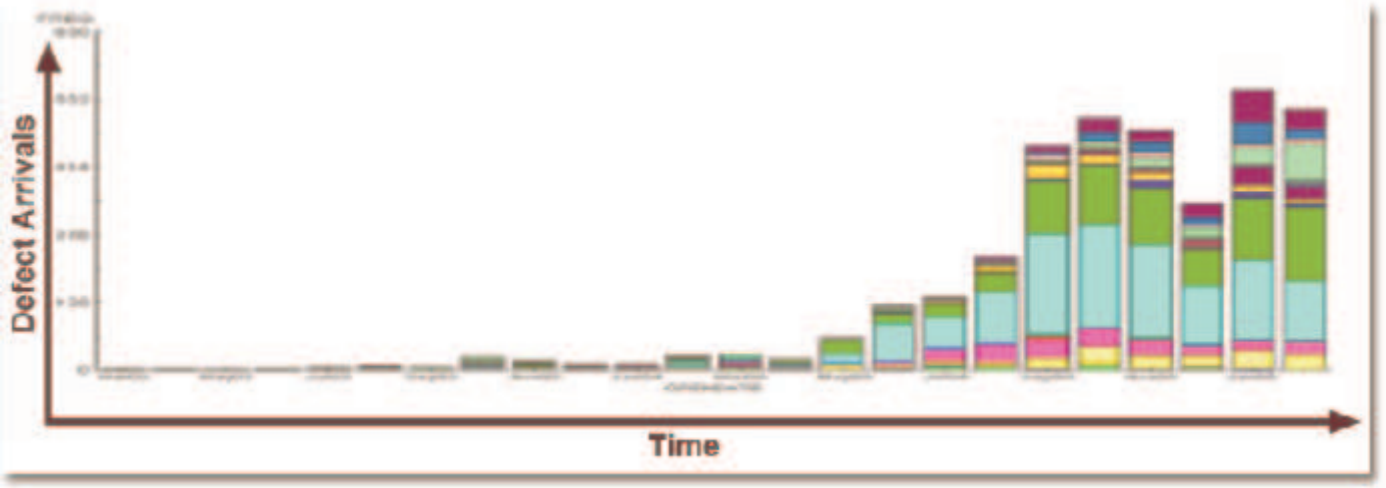
- Daily builds are a good start
- Agilists update and test their code constantly
- Therefore they need to build the system constantly
  - Compile
  - Regression testing
  - Static code analysis
- Critical points:
  - Must be automated
  - Don't forget database integration
  - Need a protocol for automatically deploying builds to higher-level sandboxes
  - Doesn't mean that you're deploying into production every 2 weeks



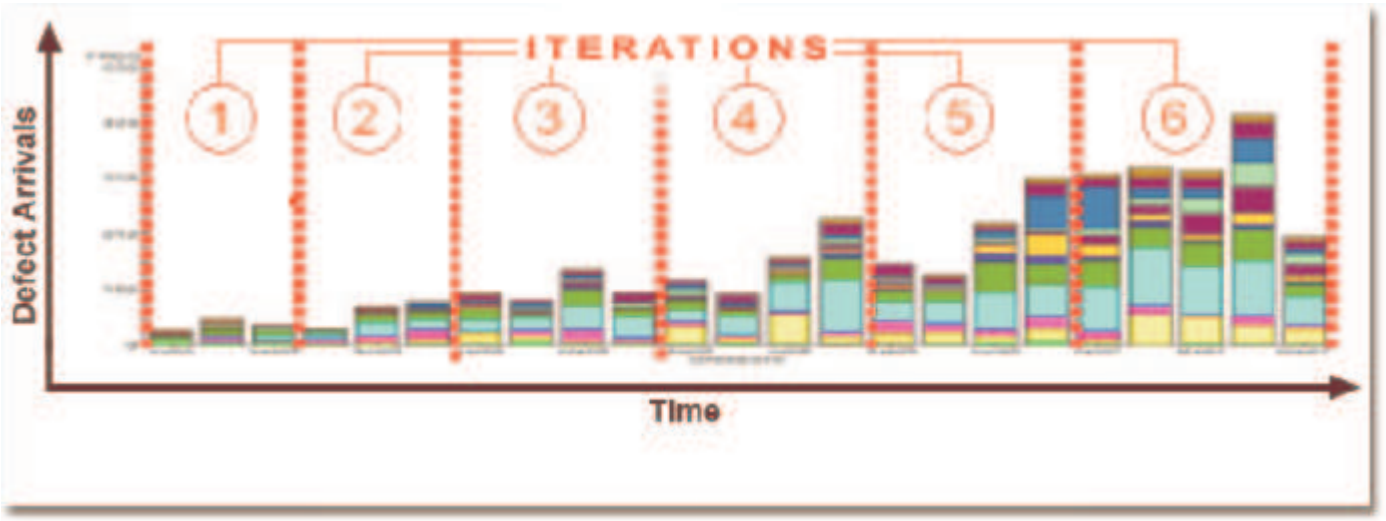
Result: Agile projects have a different test profile

Deliver sooner, fix earlier

Waterfall Profile
Defects found later when they are more expensive to fix

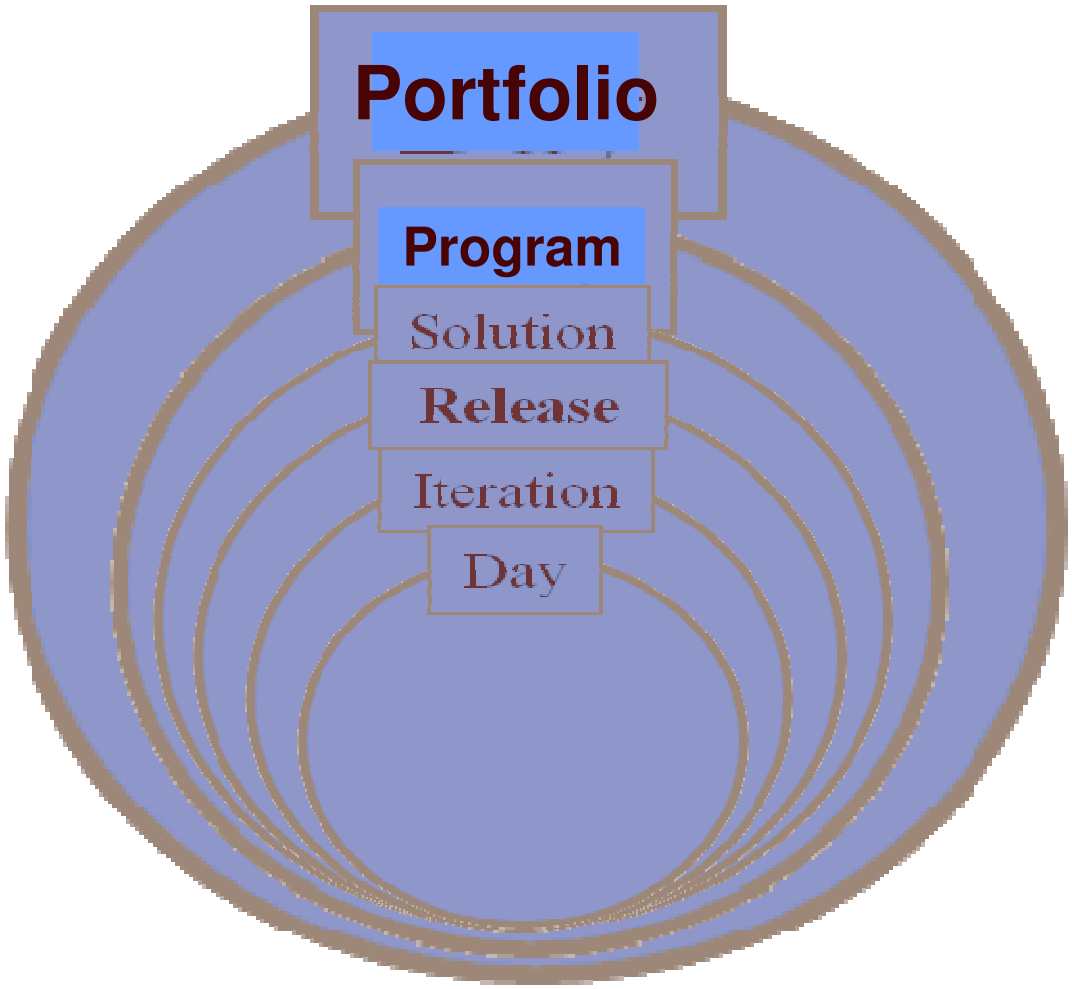


Agile Profile
Defects found early when they are cheaper to fix



Data from IBM Websphere CICS team based on their agile adoption

Agile Project and Program Planning



Most agile teams are concerned only with the three innermost levels of the planning onion…in may situations this is not enough!

Based on the work of Mike Cohn

From Walker Royce, "Improving Software Economics", 2009.,

Evolving View of Software Management -1

Top 10 Management Principles for Waterfall Projects

1. Freeze requirements before design.
2. Forbid coding prior to detailed design review.
3. Use a higher order programming language.
4. Complete unit testing before integration.
5. Maintain a software change activity database.

Most Project Managers Know How To Manage Projects Like This!

6. Thoroughly document each stage of the design.
7. Assess quality with an independent team.
8. Inspect everything.
9. Plan everything early with high fidelity.
10. Control source code baselines rigorously.

From Walker Royce, “Improving Software Economics”, 2009.,

Evolving View of Software Management -2

Top 10 Management Principles for Iterative Projects

1. Base the process on an architecture-first approach.
2. Establish an iterative lifecycle process that confronts risk early.
3. Transition design methods to emphasize component-based development.
4. Establish a change management environment.
5. Enhance change freedom through tools that support round-trip engineering.
6. Capture design artifacts in rigorous, model-based notation.
7. Instrument the process for objective quality control and progress assessment.
8. Use a demonstration-based approach to assess intermediate artifacts.
9. Plan intermediate releases in groups of usage scenarios with evolving levels of detail.
10. Establish a configurable process that is economically scalable.

Some Project Managers Know How To Manage Projects Like This!

From Walker Royce, "Improving Software Economics", 2009.,

Evolving View of Software Management -3

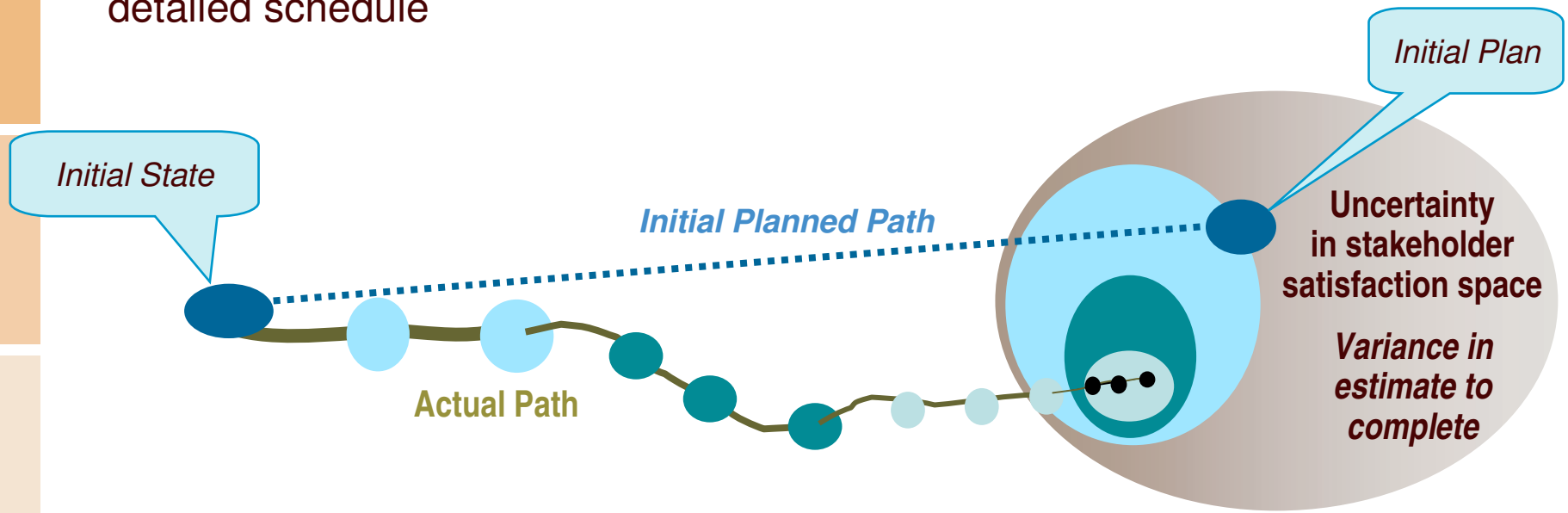
Top 10 Management Principles for Agile Projects

1. Reduce uncertainties by addressing architecturally significant decisions first.
2. Establish an adaptive lifecycle process that accelerates variance reduction.
3. Reduce the amount of custom development through asset reuse and middleware.
4. Instrument the process to measure cost of change, quality trends, and progress.
5. Communicate honest progressions and digressions with all stakeholders
6. Collaborate regularly with stakeholders to renegotiate priorities, scope, resources, and plans.
7. Continuously integrate releases and test usage scenarios with evolving breadth and depth.
8. Establish a collaboration platform that enhances teamwork among potentially distributed teams.
9. Enhance the freedom to change plans, scope and code releases
10. Establish a governance model that guarantees creative freedoms to practitioners through automation.

Few Project Managers Know How To Manage Projects Like This!

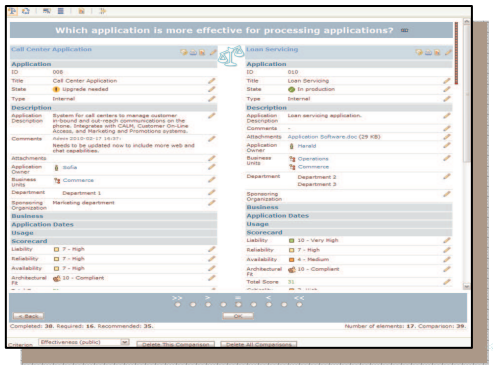
Principles of Agile Planning

- Initially:
  - Make early, high-level predictions about the cost and schedule
- Over time:
  - Improve your prediction from your initial project plan based on actuals
  - After a few iterations, your project plan should be substantially better
- Goal is to get a reasonable, not perfect estimate, and a reasonable, but not detailed schedule

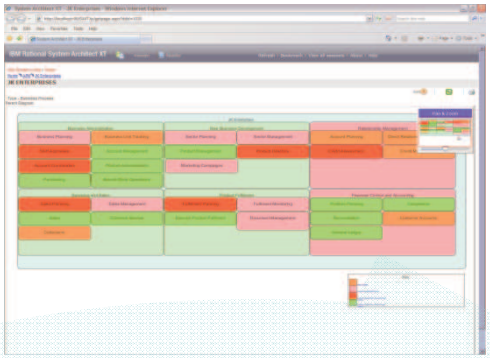


Agile Planning at the Outer Levels

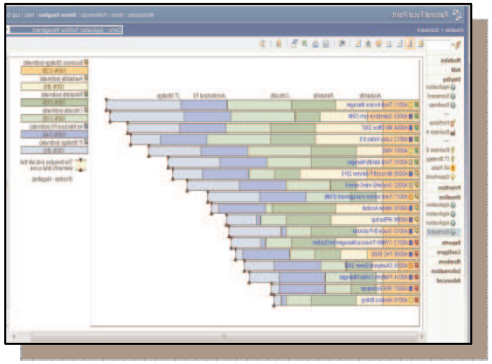
Analyze Business & IT Priorities



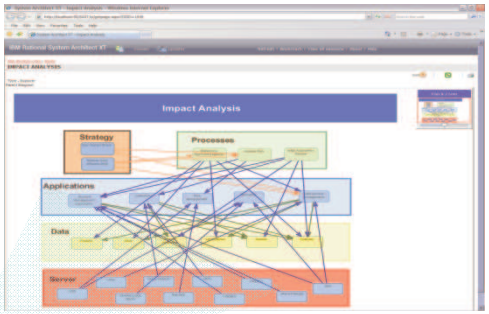
Define Transition Initiatives



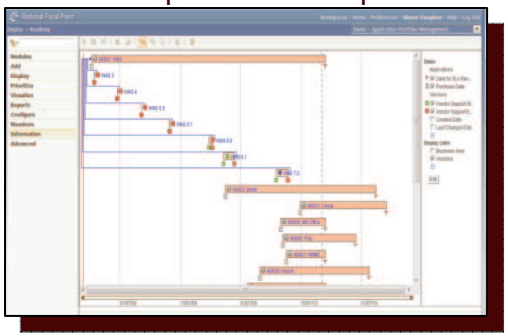
Analyze and Prioritize Initiatives



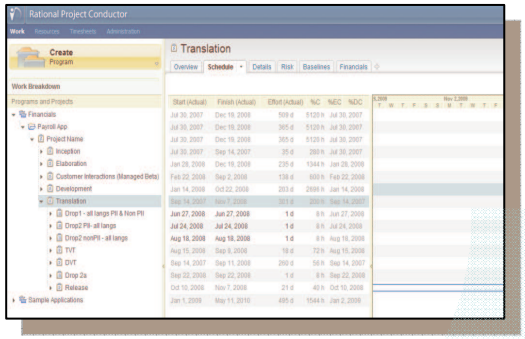
Compare to Current State, Perform Gap Analysis



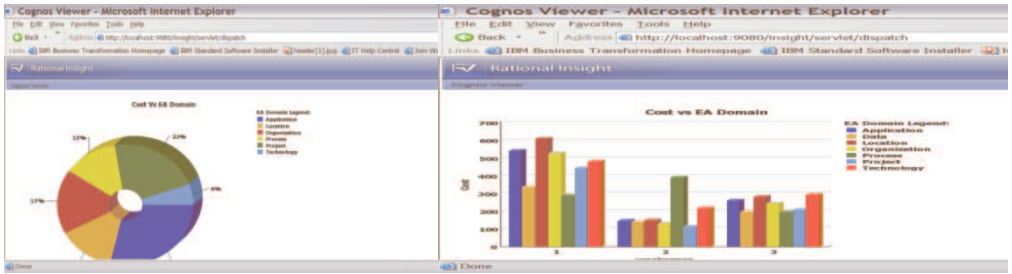
Define Scope & Roadmaps



Plan Projects for Integrated Execution and Feedback



Monitor With Executive Dashboards



Agile Adoption Planning

- Achieve early, relevant and measurable success
- Adopt each of the practices incrementally – no “big bang”
- Create a suite of practices each with a repository of relevant material, guidance, governance to embed excellence
- Map each practice to individual stakeholder challenges
- Align each practice to organisational operational drivers
- Measure adoption of each practice and assess its business value

An Example Practice Prioritization

- Foundation
  - Iterative Development
  - Two-Level Planning
  - Team Change Management
  - Shared Vision
  - Continuous Integration
  - Whole Team
- High
  - Risk-Value Lifecycle
  - Test-driven development
  - Use case-driven development
- Medium
  - Evolutionary Architecture
  - Concurrent Testing
- Low
  - Business Process Sketching
  - Evolutionary Design
- Ultra Low
  - Process authoring and Tailoring
  - Requirements Management
  - Formal Change Management
  - Component Based Software Architecture
  - Design Driven Implementation
  - Test Management
  - Independent Testing
  - Application Vulnerability Assessment
  - Performance Testing

Summary

- Agile in context
- Thinking agile…acting agile…living agile
- Seven habits of successful agile adoption
  - Be explicit about you agile goals
  - Understand the dimensions of scale up/out
  - Use measures to govern behaviour
  - Focus early on quality as a team issue
  - Re-skill your project/program planners
  - Grow with a clear adoption plan
  - Think globally, act locally!


Thank
You