Innovate2011
The Rational Software Conference
11th and 12th of October
Let's build a smarter planet.

Are we there yet? IBM Software Groups agile journey

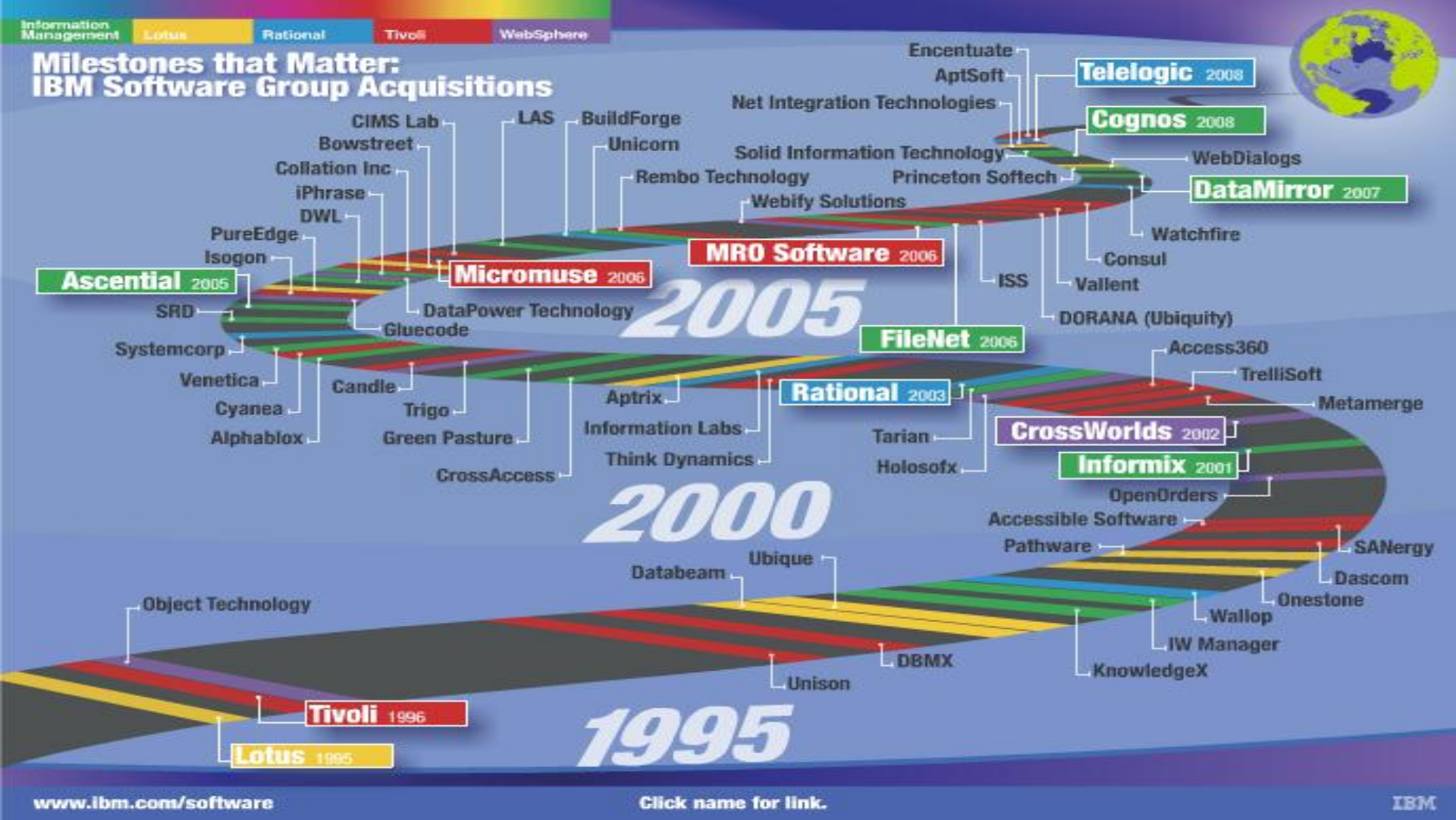
Jon Tilt / Tony Grout
Role: Solution Delivery Transformation Engineer / World Wide Rational Tiger

Problem

We need to get more effective

- Deliver what our customers will buy – not more or less
- Protect scarce development resources – stop doing things that don't add value to our customers
- Reduce rework and waste in the development process.

Big Bang Just Doesn't Work

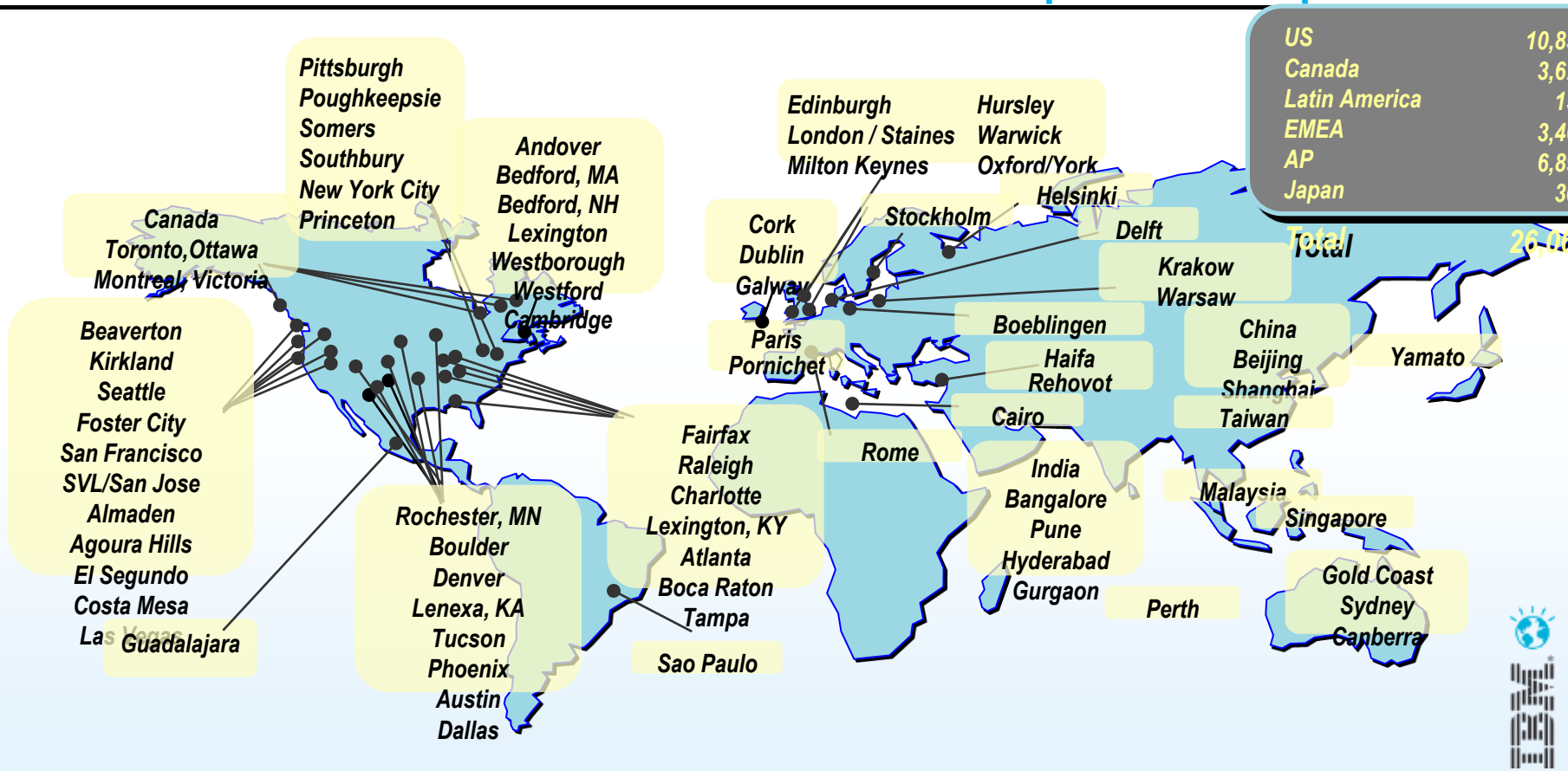


www.ibm.com/software

Click name for link.

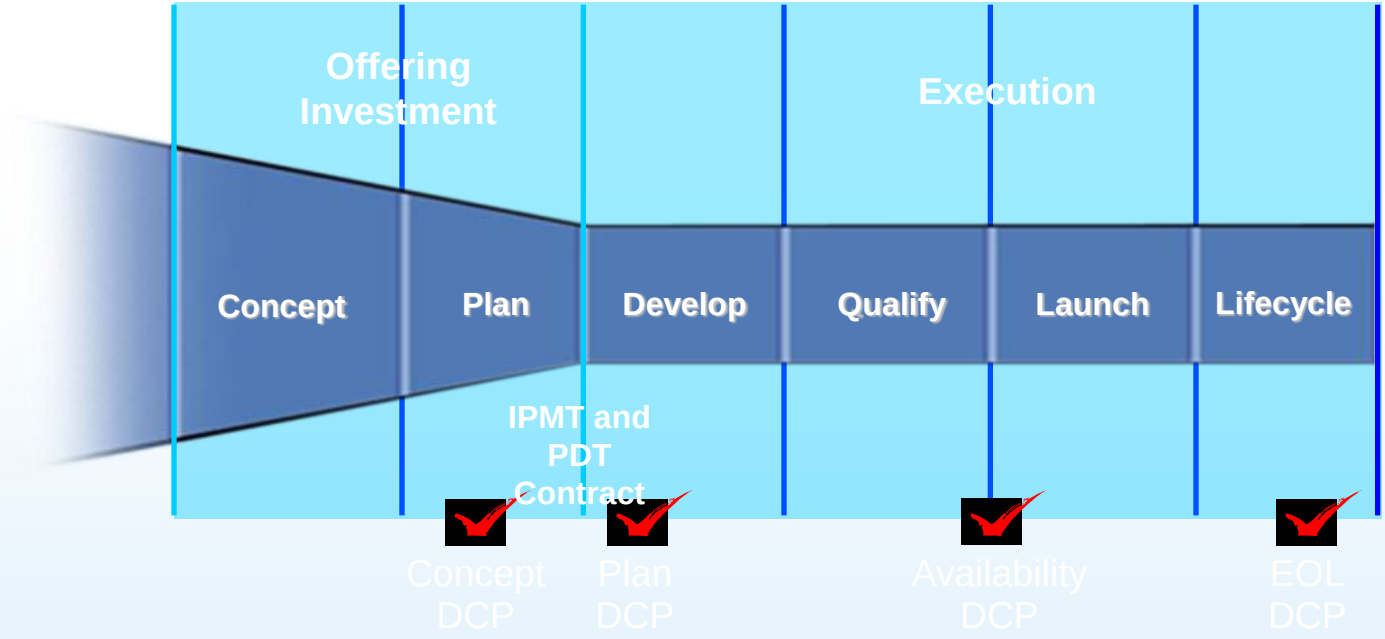
IBM

A Global Team of IBM Software Group Developers



IPD Process: Formal Decision Check Points

Event Driven Formal Decision Check Points (DCPs)



…. and yes we can exploit agile practices within this framework

Some keys to today's transformation efforts….

Collaboration through communities

- Employ collaboration across communities for everything from the SWG Architecture Board to Development Best Practices to Test Automation ("None of us are as smart as all of us")

Encouraging a culture of reuse

- Continue to expand the reuse program to drive development efficiencies, consistent component behavior and improved portfolio quality.

Agile/Lean enablement

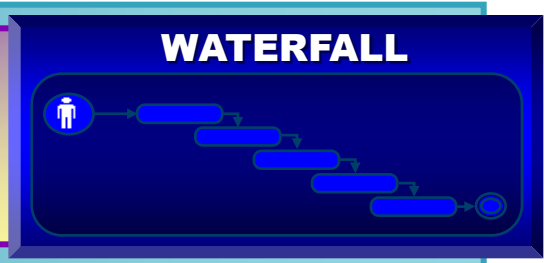
- Provide all SWG development teams with the tools they need to efficiently deploy appropriate agile/lean practices to improve their business performance

Iterative, Agile and Lean Software Development

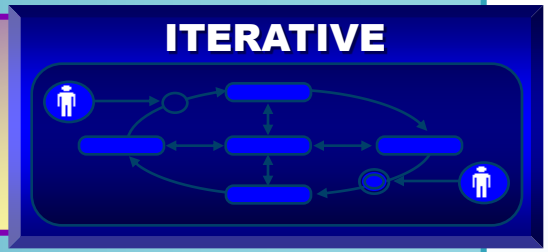
IBM Software Development Transformation

1980's

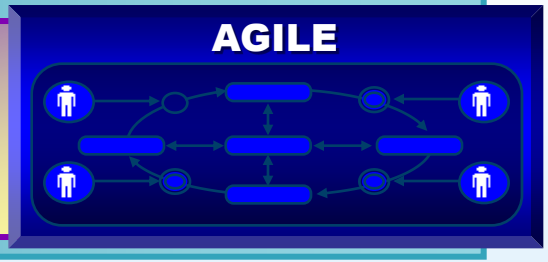
- **Waterfall development**
  - Rigid, late feedback, slow reaction to market changes



- **Iterative development**
  - Customized RUP, community source and component reuse, emphasis on consumability



- **Agile / Lean development**
  - Global reach, agile practices, outside-in development, tools and not rules



Present

Continuous Learning and Adaptive Planning

Agile and Lean on One Slide

Agile

- Individuals and interactions
  over processes and tools
- Working software
  over comprehensive docs
- Customer collaboration
  over contract negotiation
- Responding to change
  over following a plan

Key Practices

- Use Cases now Epics and Stories
- Iterative Development
- Continuous Customer Validation
- Test Driven Development
- Daily Scrum
- Maximum Automation
- Trust the Team

Lean Themes

- Eliminate All Waste (All Rework)
- Build In Quality (Discipline & Defect Prevention)
- Create Knowledge (Tune Product and Process)
- Defer Commitment (Keep your options open)
- Deliver Fast (Iterate and share)
- Respect People (Trusted to make decisions)
- See the Whole (Avoid Sub-optimisation)

Tactics

- Focus on Customer Value
- Reqts = Use Cases = No additional Functions
- Validate often with the customers = Use Iterations
- Just in time artefacts to prevent need for rework
  - Use Cases ➔ Design ➔ Develop ➔ Test = Iterations
- Fast Cycles limit Rework
- Architect for rapid change – be willing to refactor
- Remove every defect at the earliest opportunity
- Don't rely on communicating through Dev Docs
- Institutionalise learning & rapid reflection

Agile Software Engineering

"Uses continuous stakeholder feedback to deliver high-quality, consumable code through use cases (user stories) and a series of short, stable, time-boxed iterations."

- Focused on identifying and reducing risk throughout the cycle
- Adaptive; expects change and reprioritization
- Communication intensive (e.g. daily Scrums)
- Aimed at making incremental progress; **working software is the measure**
- Disciplined, scaleable, collaborative and effective across sites
- Potentially ready to ship **every** iteration

*A good agile project will deliver the most **Business Value** possible, within the project constraints, … improving on the original plan*

Five Levels of Agile Planning

Product vision (2-5 years)

- **Desired future “state”**
- **Elevator statements**

Product roadmap (1-2 years)

- **Plan to implement product vision through multiple releases**
- **Prioritized product backlog of epic user stories that describe release themes**

Release plan (3-12 months)

- **“Next step” in delivering the roadmap**
- **Pull top Epics from the product backlog to create the release backlog.**
- **Break Epics down into Stories that fit into iterations**

Sprint plan (2-4 weeks)

- **Next "Step" in delivering highest priority stories from the release backlog**
- **User stories broken down into tasks**

Daily work (hours)

- **Daily 15-minute Scrum Meetings to plan work and make impediments visible**
- **Daily work to complete the user stories**

Think before diving in…..one size does not fit all….

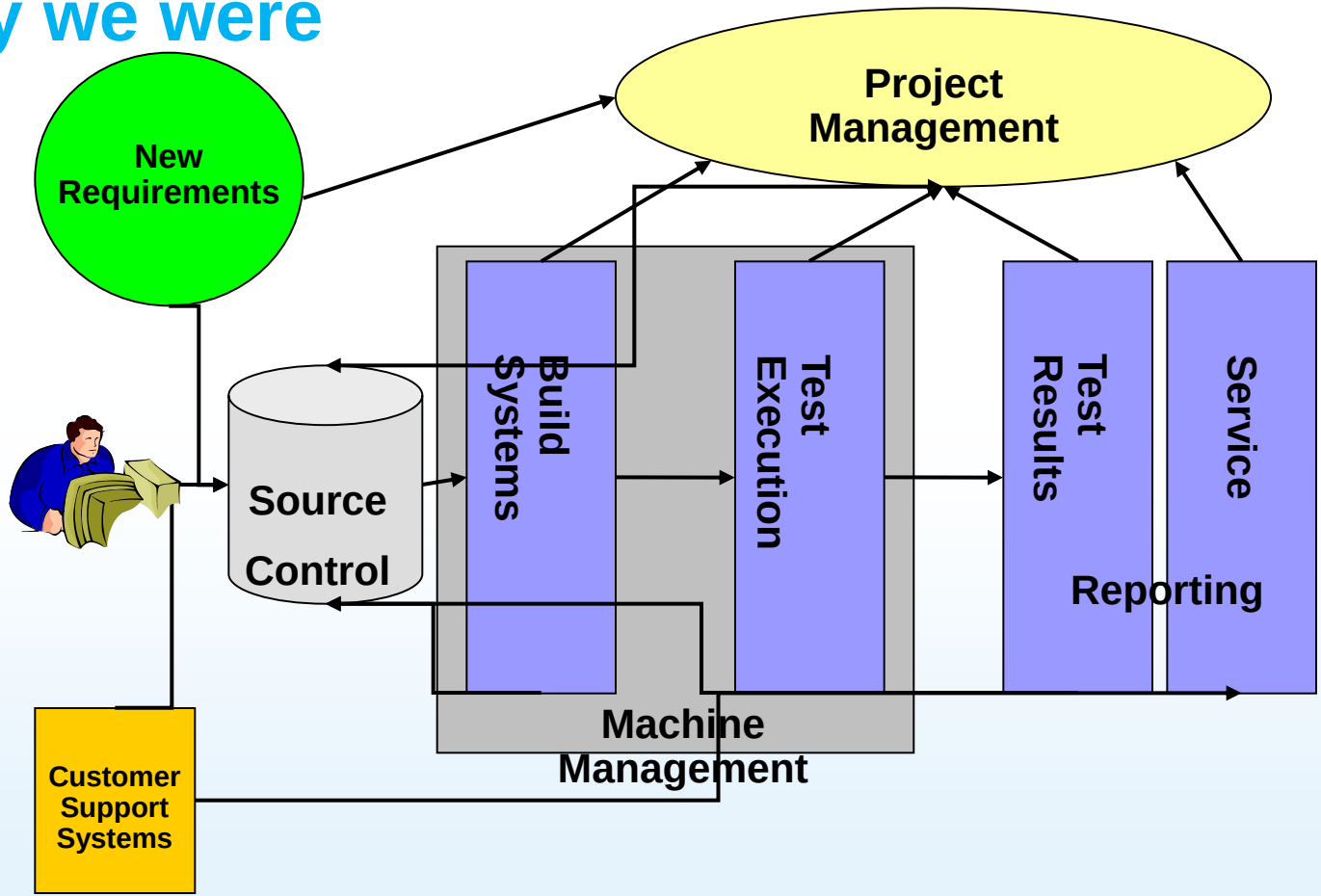
SWG composed of diversity of project profiles

- From:New small (20 HC) teams across a couple of sites looking for 6 month product releases
- To: Mature large 600+ teams with WW sites looking for 2 year product releases and incremental feature packs in between

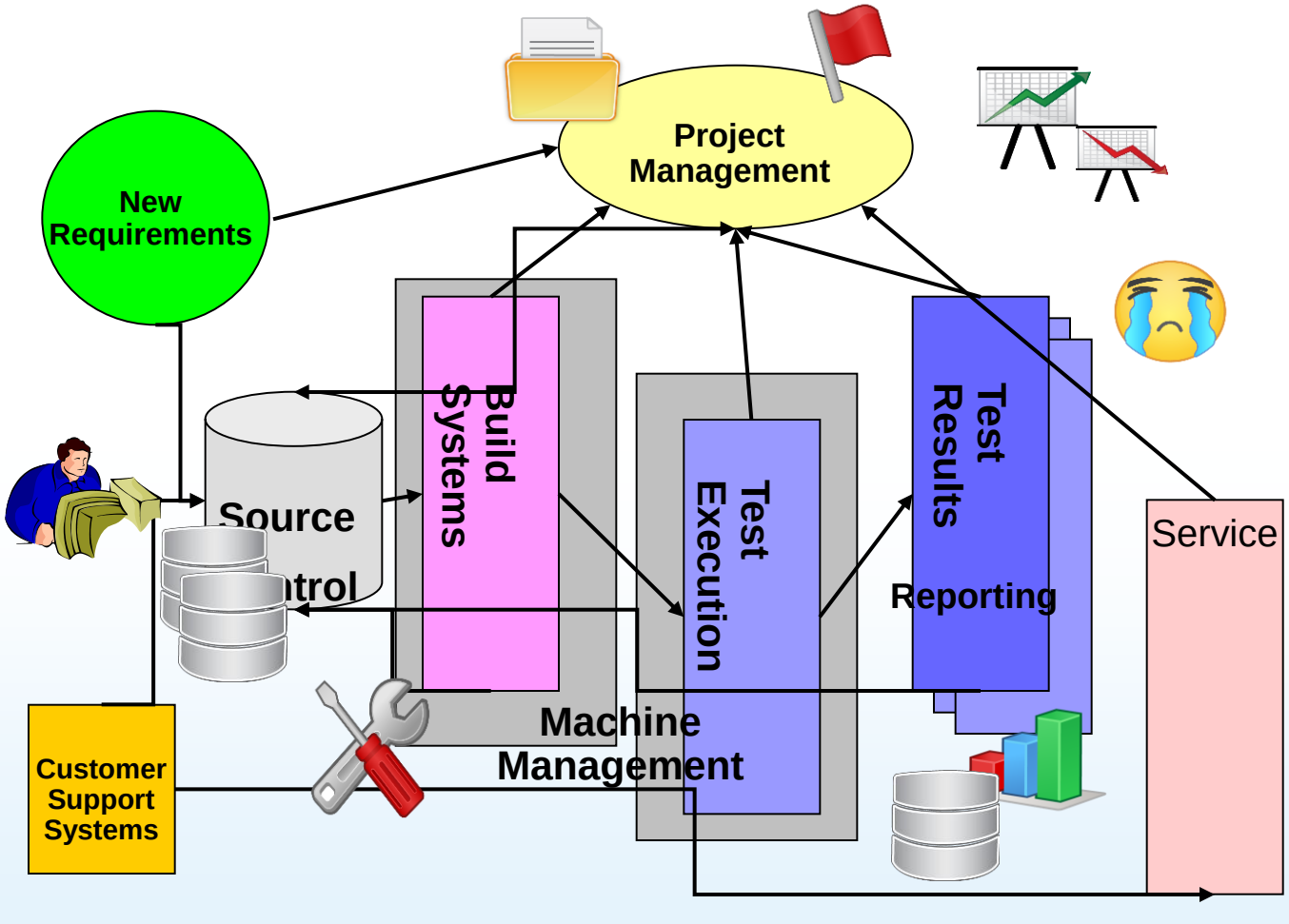
Practice adoption and pace should fit team goals and risk management strategy

- Learn from others
- Make incremental, achievable changes focused on goals
- Go for early wins
- Failures will occur…..learn and move on without disillusionment

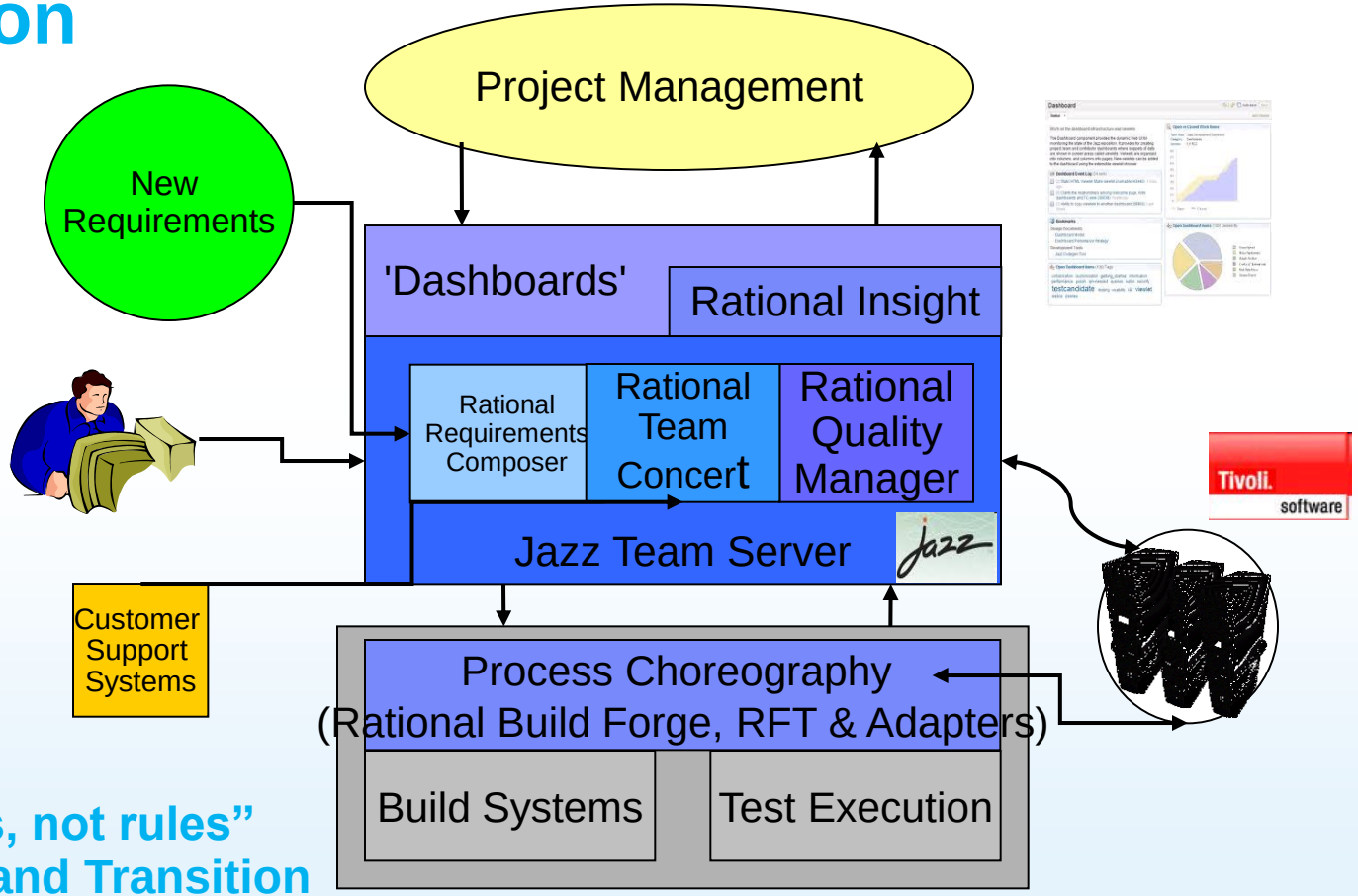
The way we were



Reality?


New Requirements
Project Management
Source
trol
Build Systems
Test Execution
Test Results
Reporting
Service
Machine Management
Customer Support Systems

Our Vision



IBM SWG "Tools, not rules"
Rational Vision and Transition

WEBSPHERE MQ EXAMPLE

Introduction – A brief history of WMQ

History:

- 16 years old this year
- Over 10,000 known customers
- Supports just about any platform you can think of
- Several million lines of code written in a combination of Assembler, C/C++, Java, PL/x

Why change to agile?

- 3 year delivery cycle too long
- Significant competition appearing over the last 4-5 years
- Market moving more rapidly than in the past
- Evolution too slow, needed revolution

What the observer saw

Retrospective Findings

- General observations
  - **Calmness and control**
  - **Continuous integration test up and running**
  - **Willingness of management to accept change**
  - **End of iteration reviews**
    - **Demonstration lead – even for middleware products**
    - **Dashboard driven**
  - **Uncomfortable celebrating success**
  - **Tools helped drive the change to agile**

Where are we?

Teams wanted more information on where the release was overall

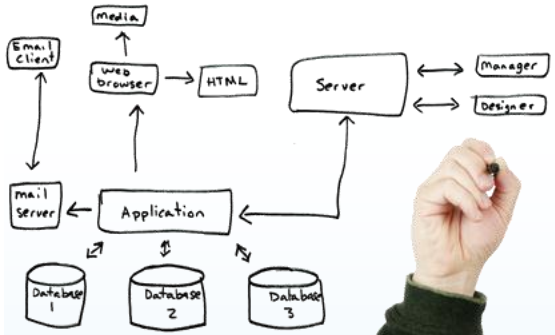
They could help balance the work and have ownership

Real-time information radiators have already been installed - monitors showing dashboards to you and me

We know who's best to help and when

Keep people in their teams

They're then able to complete work for a week in to the next iteration (for removing defects and better forward planning)



I like being part of a team

Teams wanted to remain in teams

Resource pools had been tried but weren’t working even for specialist skills

This was already being changed by new management

How big is it?

Teams needed help in estimation

We’re already planning in facilitated calibration and estimation workshops

Which ball do we drop?

Teams needed support with triaging when there was too much to do and not enough time

The team were used to being told not asked

Workshops planned for techniques and tooling for prioritising

Some people just don’t like agile

They’d rather be told what to do and work in a silo

The continuing pace is something they’re not comfortable with

The level of interaction is too much

Management Retrospective

More	Less
More definition around what we mean by done	Less delivery teams starting in parallel
More sharing of vision and gaining buy-in	Less specialists
More meaningful and early communication with customers	
More checking rather than waiting (go see)	
More devolution of control (with coaching support)	
More standardisation of process	
More flexibility around iteration lengths between teams	
More contingency	
More planning and definition of requirements	

Summary of achievements

Zero tolerance of regressions and general technical debt reduction:

- 90% reduction in deferred defects

Calendar monthly iterations:

- Much clearer focus on short term objectives

Collocation of delivery teams:

- Improved communications and flexibility within teams.
- Off-site teams gain greater autonomy and more interesting work
- Greater understanding of the perspectives of the different disciplines.

Summary of achievements (Contd.)

Test infrastructure and measurement:

- Regressions typically spotted within 24 hours, compared with 2-3 weeks.
- Average defect turnaround cut from 5-6 weeks, down to approx. 1 week

LID process & user stories

- Improving communications between our Strategists and Architects
- Creating a wider team understanding of expected use of features and their business value.

Customer program

- Helped to instil culture of maintaining build stability.
- Time to stabilise and ship down from approx. 3 weeks to 2-3 days.

Rational Team Concert

- Helped establish user stories as our currency for change.

Websphere MQ Conclusions

WebSphere MQ has made big steps forward in becoming more agile

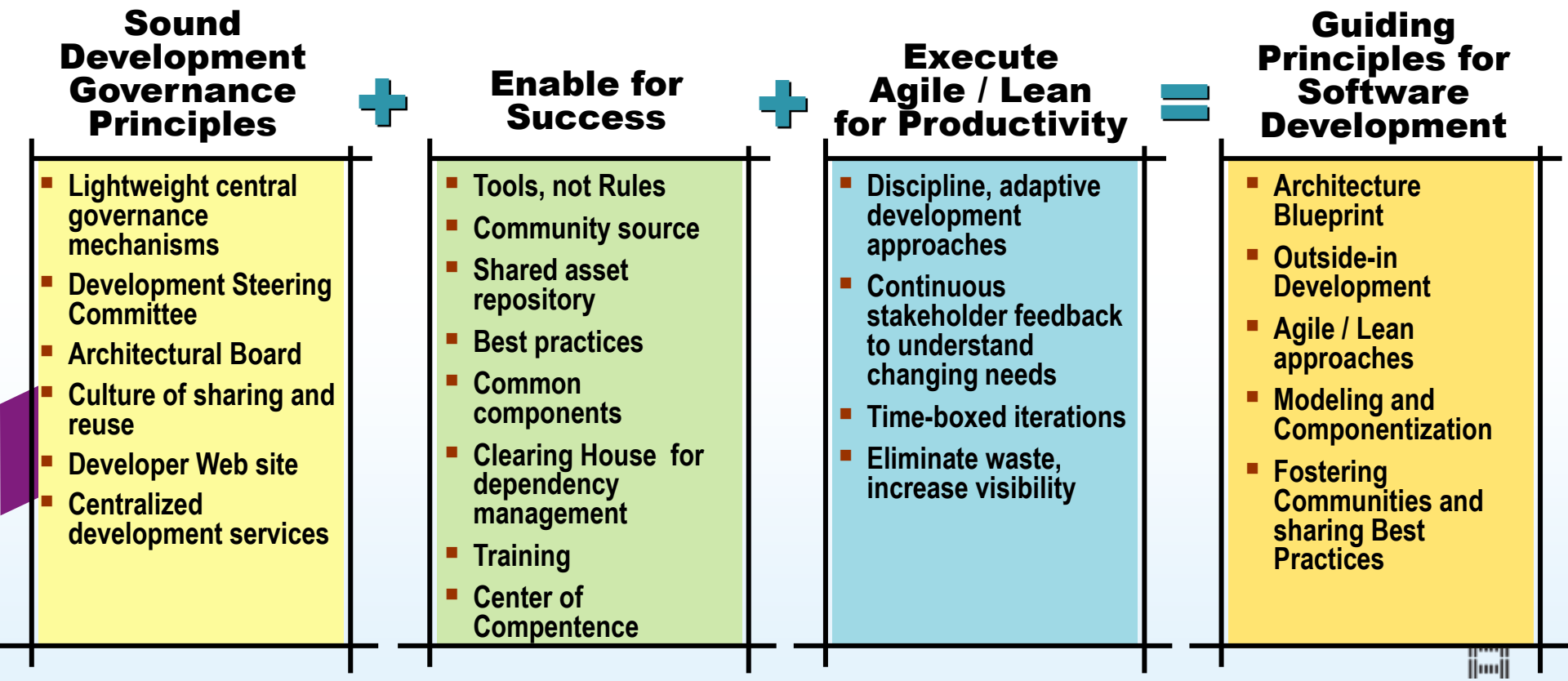
- A range of actions have already been taken
- Positive impact has already been seen in productivity, focus on customer and market needs, and shipped product quality
- WebSphere MQ v7.1 is the first major delivery in which these benefits will reach the market

This move to increased agility is still a work in progress, with evolution of processes through the coming releases based on lessons learned

We are keen to share our experience with other teams and with customers

Summary

Best Practices for Distributed Development



In Summary

These are naturally effective software development approaches

Agile and Lean are Very disciplined

- This isn't an excuse for code and fix

Use a Learning Approach in your teams

- Big Bang Doesn't Work

Transformational capabilities are within the organization

Educate, enable and empower the teams

Tools and not rules