Disciplined Agility

Discipline is the foundation.
Agility is the next level

Date: **January 2006**
Presented By: **Rich McCabe**

Systems and Software Consortium | *2214 Rock Hill Road, Herndon, VA 20170-4227*
Phone: (703)742-8877 | FAX: (703)742-7200
www.systemsandsoftware.org

About the Consortium

Systems and Software Engineering Practices

Realizing value from process improvement
- Value-driven process improvement
- Quantifiable business performance measures
- CMM®, CMMI® appraisals

Implementing integrated engineering
- Requirements analysis & automated testing
- Architecture and design
- Security
- Measurement & analysis
- Verification and validation/Mission assurance

Life cycle strategies for complex systems
- Project management
- Agile development
- Distributed development approaches
- Systematic reuse / Product lines

Applied to Member Needs

As a Consortium
- Shared challenges/co-funded development
- Practitioner-led training
- Technology transfer

As a Teammate
- Subject matter experts
- Process consulting
- Technology consulting

As an Industry Association
- Voice of Industry
- Influence govt. agencies
- Best practices/guidelines
- Neutral ground/honest broker

Learn more at
www.systemsandsoftware.org
with a For Members Only account

Past, Current, and Future Challenges

- Increasing project complexity
- Demand for quicker delivery of useful systems
- Increasingly vague, volatile requirements
- Greater uncertainty/risk from limited knowledge of
  - Underlying technologies
  - Off-the-shelf (OTS) components used

Are your projects troubled by these challenges?
What % of projects in your organization fail?

Why Do Projects Go Bad?

- Integration and test takes "too long"
- Lots of defects and rework is needed
- Technologies or off-the-shelf components don't work as expected
- Interfaces are naively defined
- Architecture has flaws that are realized late in the development cycle
- Requirements are poorly understood
- Requirements change throughout the development cycle

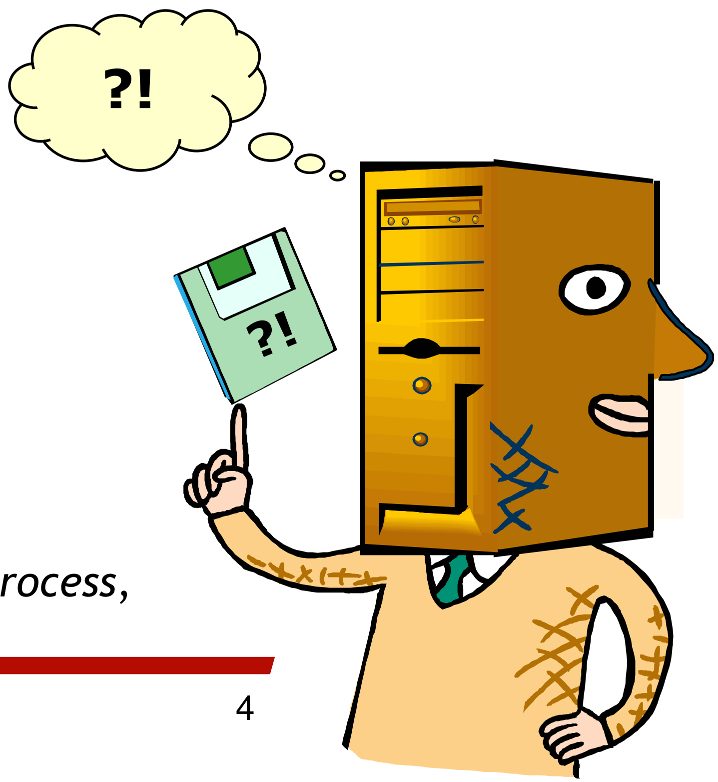
Key knowledge is learned at the end of the development cycle when it is too late to adjust!

Development Is A Learning Enterprise

- System development is knowledge-based
  - Discovering, generating, encoding knowledge
  - Finding out what you know, and what you don't know

"When we build software…the product is not the software; it is the knowledge contained in the software."*

"… for the most part, engineers do not know how to build the systems they are trying to build; it is their job to find out how to build such systems."*



* Phillip Armour, *The Laws of Software Process*, ISBN 0849314895, 2004

SYSTEMS AND SOFTWARE CONSORTIUM
BUILDING BETTER SOLUTIONS TOGETHER

Problems With Current Practice

- Waterfall is a poor learning process
- Too much emphasis on compliance
  - Focuses on following a pre-determined plan
  - Tries to prevent change, control variance from plan
- Low predictability (!)
- Too much speculative documentation
  - High cost
  - Not well-informed—rapidly obsolete
  - Not maintained as more knowledge is discovered

"The 'waterfall model' may be unrealistic, and dangerous to the primary objectives of any software project"*

* Tom Gilb, "Evolutionary Delivery versus the 'Waterfall Model'" *ACM Sigsoft Software Engineering*, July 1985

Positive Trends Begin to Emerge

- 1990s saw a movement toward "greater discipline and governance"—better quality but slower response
- Standish Group annual Chaos Report on IT projects
  - 1994: 31% cancelled, 53% go 89% over estimate
  - 2004: only 15% cancelled, but 51% still "challenged"
  - Improvement attributed to "…smaller…projects with iterative processing as opposed to the waterfall method" — *Standish Chairman Jim Johnson*
- Iterative approach has a history of successes
- Agile development is iterative development++

Foundation of Agile Development

- Development is essentially learning and exploration
- Learning is accelerated by frequent feedback
- Use/review of a working system provides the most effective and credible feedback
- Rapid, evolutionary development/delivery demands
  - Efficient team communication
  - Robust system design
  - Low defect rate (process maturity)
  - Effective decisions from empowered teams

A complex system that works is invariably found to have evolved from a simple system that worked ... A complex system designed from scratch never works and can not be patched up to make it work. You have to start over, beginning with a [simple] system that works.

Gall, J. (1986). *Systemantics: How Systems Really Work and How They Fail*

Agile "Brand Name" Methodologies

- eXtreme Programming (XP) *[Beck]*
  - Widest known, developer-focused for small teams
- Crystal methodolgies *[Coburn]*
  - Set of methodologies conditional on circumstances—Only 2 defined: Crystal Clear, Crystal Orange
- Feature-Driven Development (FDD) *[Palmer]*
  - Agile approach closest to conventional development
- Scrum *[Schwaber]*
  - Focused on management practices
- Lean Software Development *[Poppendieck]*
  - Inspired by Toyota Production System, particularly its product **design** practices

What Defines Agile Development?

- Evolving systems in short iterations
  - Each release is a working system
  - Design for change
  - Focus on value
  - Actively guide to convergence

2-13 weeks!

- Communicating efficiently

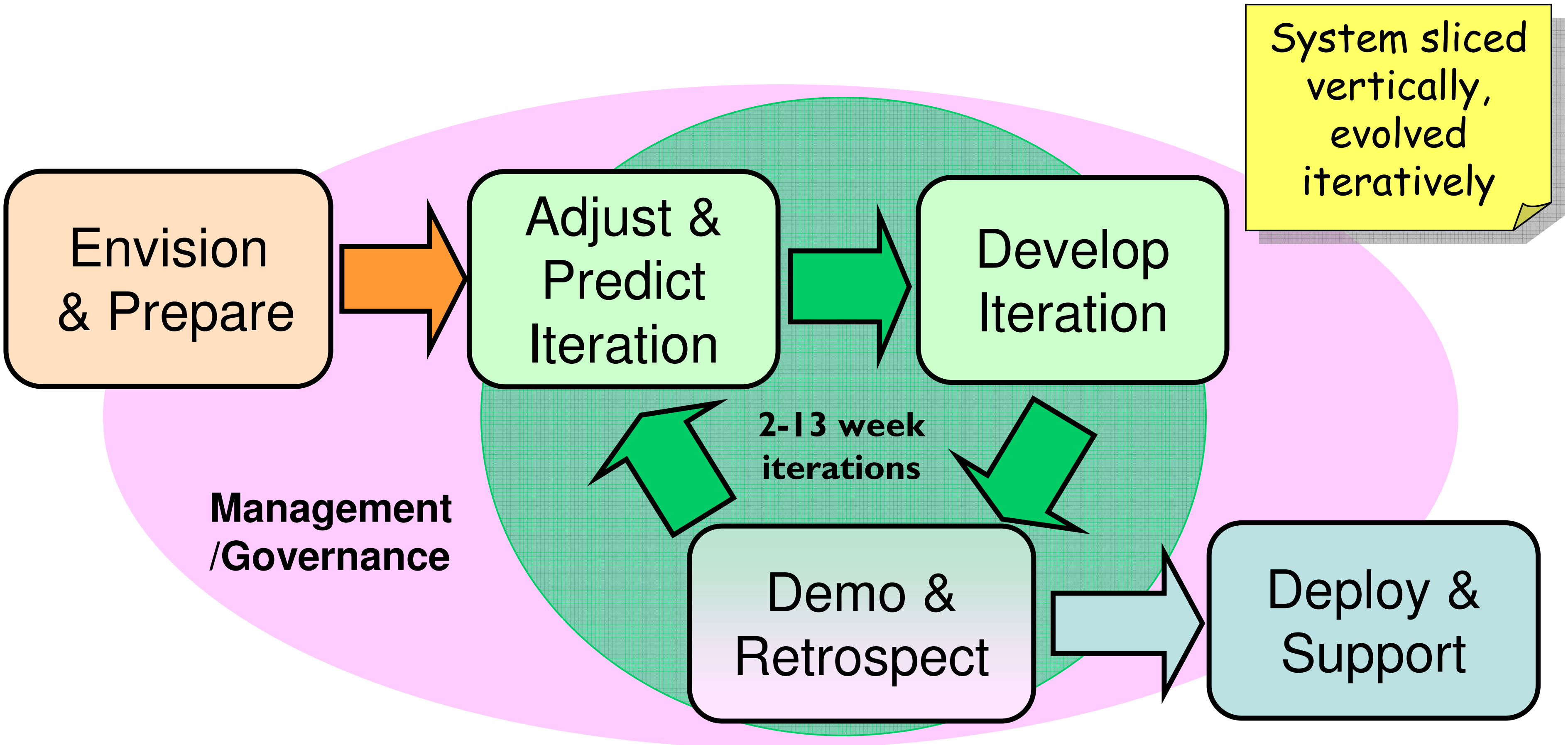
- Leveraging human strengths
  - Engage, align, and empower the team
  - Get power from each member

Comparing various interpretations of agile development, these themes seem to be common and essential

Typical Agile Development

- Applications evolve in multiple short iterations
  - Iterations are constant length, in range of 2-13 weeks
  - Release a working application at end of each iteration
  - Add as many of customer's highest priority features to each new release as can fit in an iteration
  - Requirements and design elaborated each release to support features in that release
  - Extensively test features in each iteration
- Customer (or customer surrogate) reviews each release—can redirect priorities for next iteration
- Track project progress by features completed
- Never slip a release date, instead slip features

Typical Agile Process



SYSTEMS AND SOFTWARE CONSORTIUM
BUILDING BETTER SOLUTIONS TOGETHER

Where's the Discipline?

- Focus and status discipline
- Test discipline
- Design discipline
- Continuous integration discipline
- Process discipline
- Role discipline
- Internal versus external discipline

Focus and Status Discipline

Iterative, *feature-oriented* development ensures

- Customers guide effort risk to align with value priorities
- Project status is credible
  - Features are completed or not; no “plausible deniability”
  - Productivity estimates are substantiated by data
  - Estimates-to-complete are increasingly reliable
- Problems signaled by early warning indicators
  - Tests begin to fail early in an iteration
  - Productivity drops across iterations
- Can we lie with agile metrics? *Eventually, but with difficulty*
- Can customers thrash? *Validation strategy is key; judicious analysis and trial deployment combat customer naivety*

Test Discipline

Extensive, continuous testing drives agile dev

- Developer tests
  - Focus design/implementation
  - Verify developer intentions
  - (As regression tests) Verify team integration
- Acceptance tests
  - Verify customer requirements
  - Measure iteration progress, inform key decisions
- Demonstration validates customer expectations
- How much testing is enough?
  What is your current test policy?—Do you focus on review instead?

Design Discipline

- Often, group design—the whole team buys in
- Refuse to even estimate features when technical risk is too high—resolve with time-boxed analysis or prototypes
- Refactoring is a necessary “overhead”—maintains design quality for continued productivity
- What is the role of enterprise architect/reviewer?
  - Same as before but review in an iterative context
    - First: Big picture, key decisions (feasibility)
    - Early on: Greatest risks on priority features laid to rest
    - Details evolved iteratively over time
  - Less (speculative) documentation and models, more results of implementation and test

Continuous Integration Discipline

- Continuous integration practices reinforce
  - Team synchronization and communication
  - System integration and test discipline
- Typical CM policies:
  - No new code accepted to baseline unless submitted with tests and all new and regression tests pass
  - No code checked out longer than ½ day
- Other typical practices
  - Daily standup
  - Group design
  - Visible status display
  - Pair programming
  - Inspections

Process Discipline

Product results reflect process discipline

- Tests, demonstrations, and productivity realistically reflect effectiveness of process
- Team collectively sets, polices, and improves its practices
  - Various practices help identify problems
    - Pair programming and inspections
    - Continuous integration
    - Team health monitoring by manager or coach
  - Retrospectives are the focal point for analysis and improvement
- What about independent QA and governance?
  This question deserves its own slide

Role Discipline

Every role has authority to match responsibility

- Customers
  - Guide the project toward highest value
  - Prioritize features by value; make cost/value tradeoffs; judge deliveries; terminate projects
- Developers
  - Meet delivery commitments, preserve production capability, minimize overhead
  - Estimate cost and risk, determine how they work as a team
- Managers
  - Represent status realistically, remove obstacles, maintain project viability
  - Set constraints, monitor results, terminate projects,

Internal Versus External Discipline

- Internal discipline is the goal
  - Team takes responsibility for its results and its habits
  - Peer pressure encourages keeping commitments
  - Team is actively engaged in improvement, iterative experimentation encourages effective innovation
- External policing can undermine discipline!
  - Dissipates responsibility, distances developers from understanding the rationale
  - Can disconnect costs from benefit
  - Too easily devolves to adversarial box checking
- However, some larger interests (e.g., government audits) must be served
- Ultimately, audits may necessarily compromise agility

Benefits of Disciplined Agility

For Customers

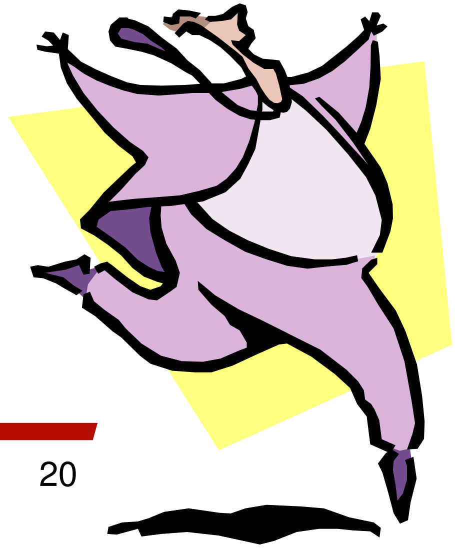
- Timely results
- Best value
- Responsiveness
- Confidence

For Managers

- Reliable results
- 10-50% improvements in productivity, defect reduction
- Easier new hire induction
- Improved customer relations
- Ongoing, rapid cycles of feedback on team performance and process improvement

For Engineers

- Feedback and affirmation
- Community
- Quality
- Progress and momentum
- Empowerment
- Focus
- A life



For More Information

- Disciplined Agility website
  http://www.systemsandsoftware.org/pub/agile/
- "Should You Be More Agile?"
  Rich McCabe and Mike Polen, *Crosstalk*, June 2002
  http://www.stsc.hill.af.mil/crosstalk/2002/10/mccabe.html