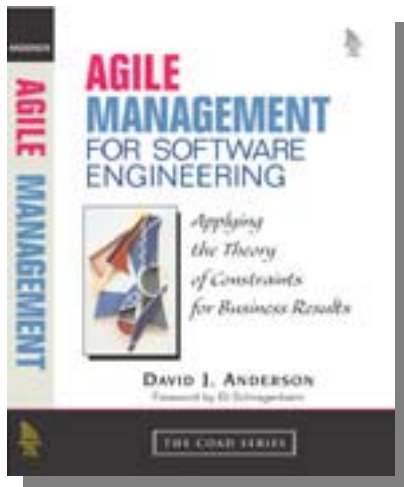


Variation in Software Engineering



David J. Anderson
American Society for Quality
Seattle, October 2004

Agenda

- Development Methods
- Project Management
- Requirements, Architecture, Components, and Distributed Development Programs



Managing Variation in Development

- Tracking the flow of value with cumulative flow diagrams (CFDs)
- Using CFD data with Control Charts
- Correlating Control Chart data with special cause events and appropriate management intervention
 - Avoiding mistake #1 and mistake #2
- Interpreting CFDs for root cause analysis
- Using TOC with software engineering



Coad Method Feature Definition

- Tiny piece of client-valued functionality
 - Typically 4 to 64 man hours of effort
 - Never more than 2 man weeks
- Business Logic Feature
 - <action> <result> [of | to | from | for] <object>
 - e.g. list availability of conference venues for given dates and attendee numbers

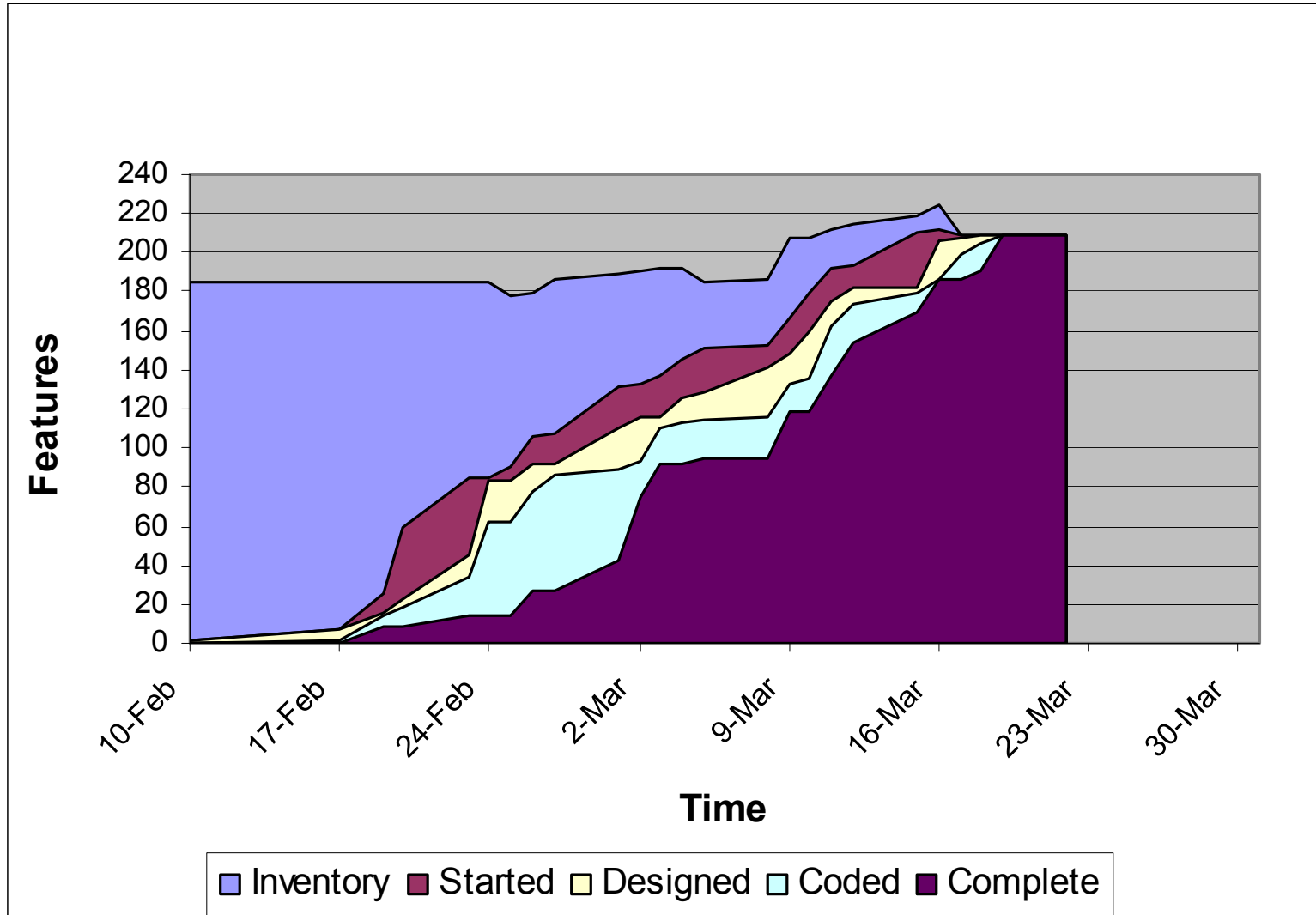


Managing The Design Factory

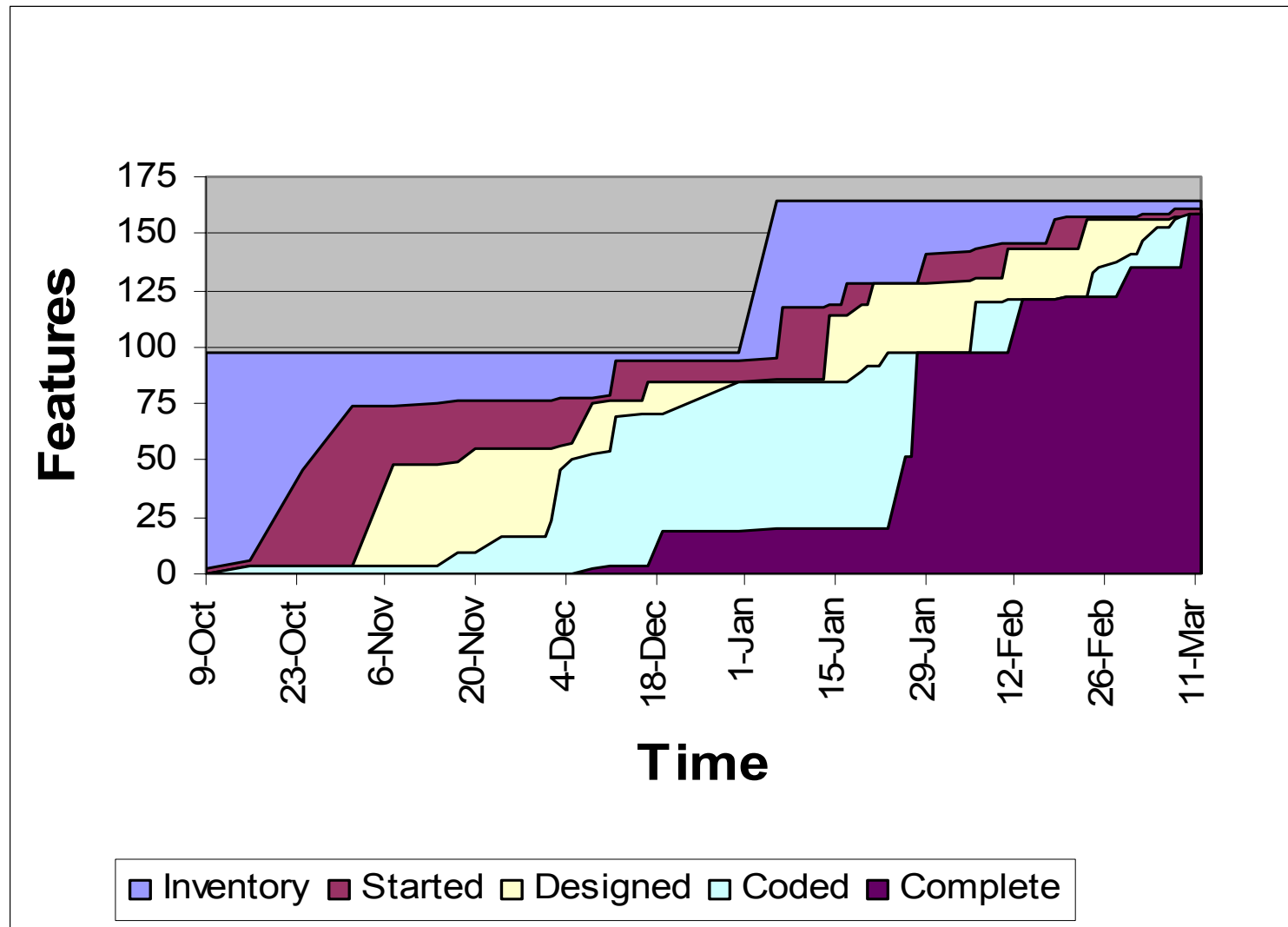
- **Marvin Patterson (1993)**
 - design is a process of discovery of information
 - Flow of value is achieved by increasing the certainty of information being discovered
- **Donald Reinertsen (1997)**
 - design processes can track the flow of information discovery, design in process (DIP) is analogous to “inventory” in production processes
 - Track with Cumulative Flow Diagram from Lean Production
 - Design is perishable – inventory depreciates over time



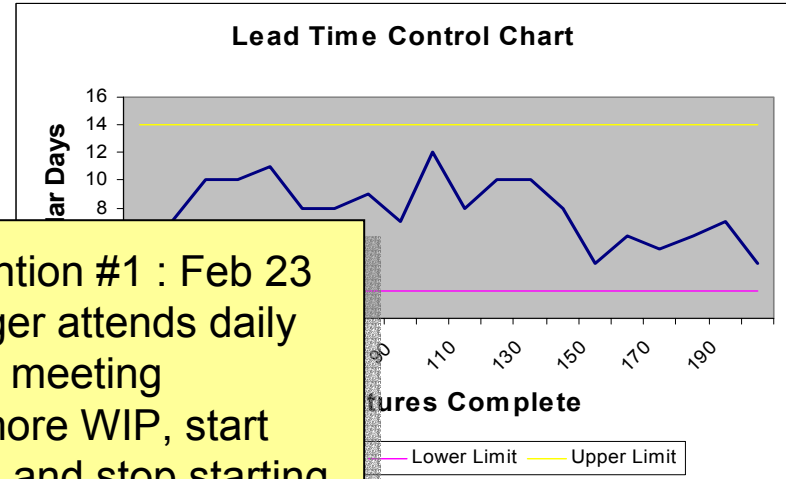
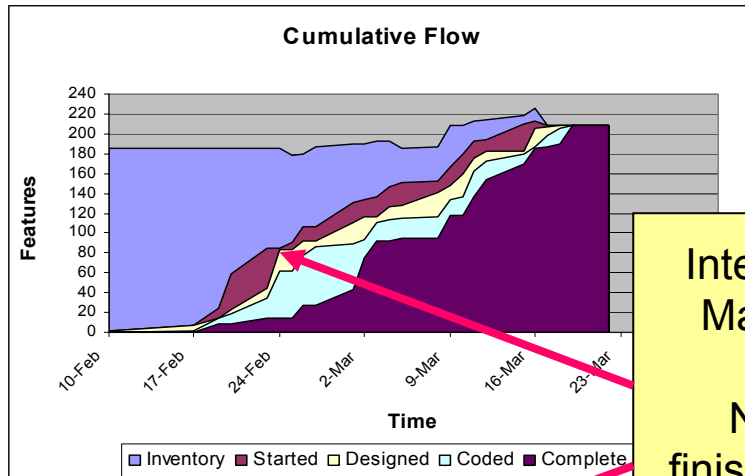
Achieving Smooth Flow



Ragged Flow

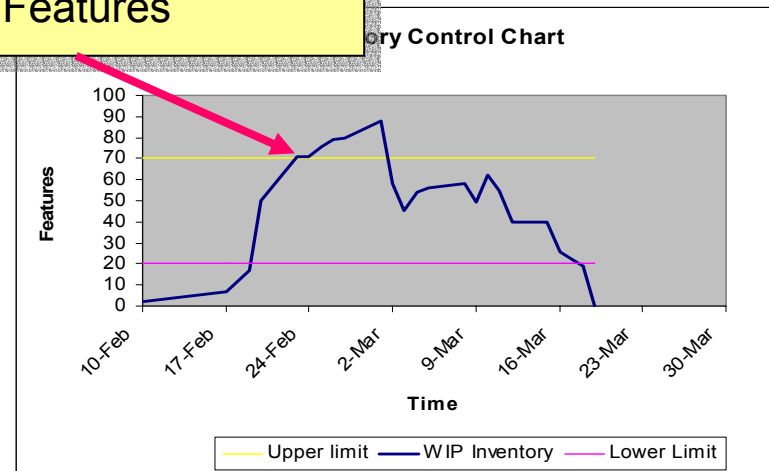
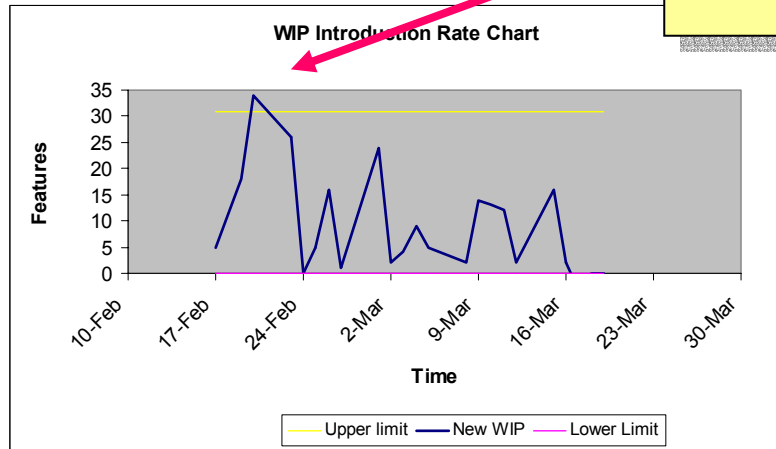


Using Control Charts with CFDs

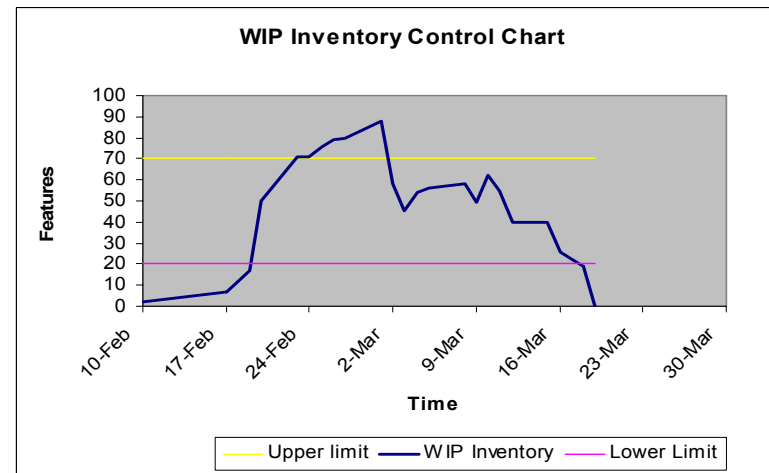
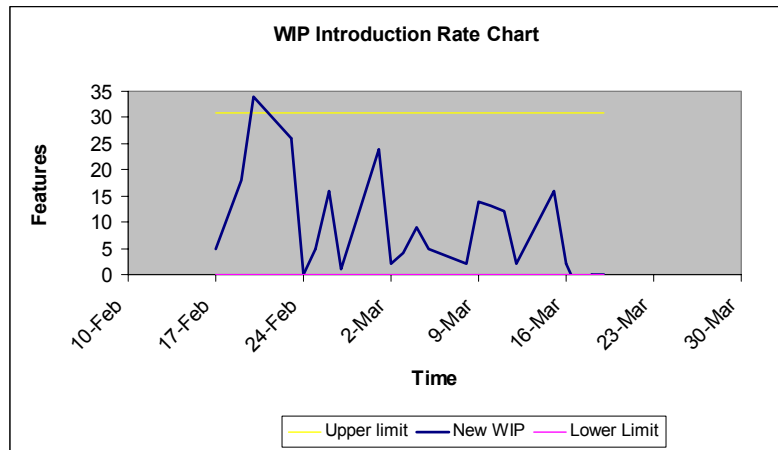
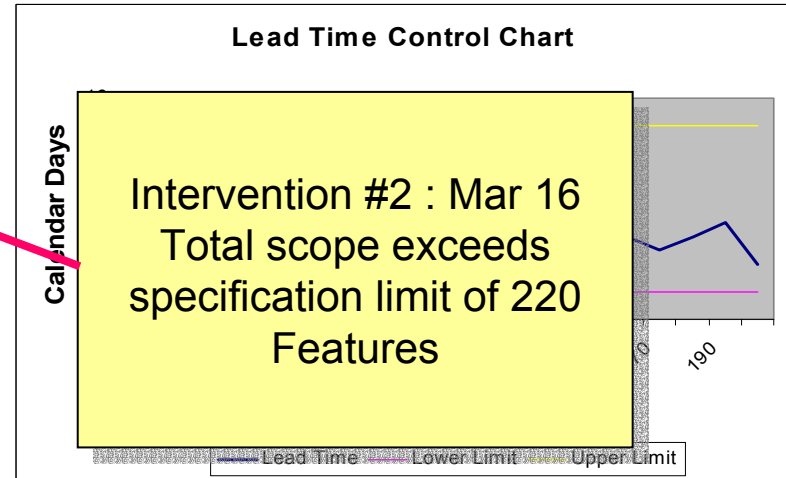
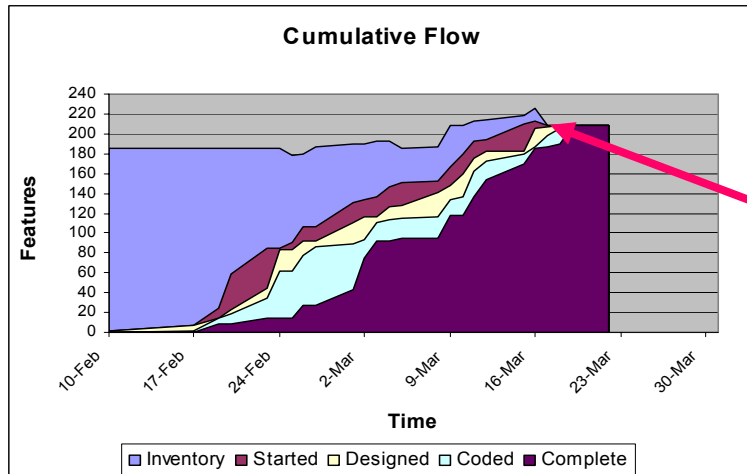


Intervention #1 : Feb 23
 Manager attends daily meeting
 No more WIP, start finishing and stop starting Features

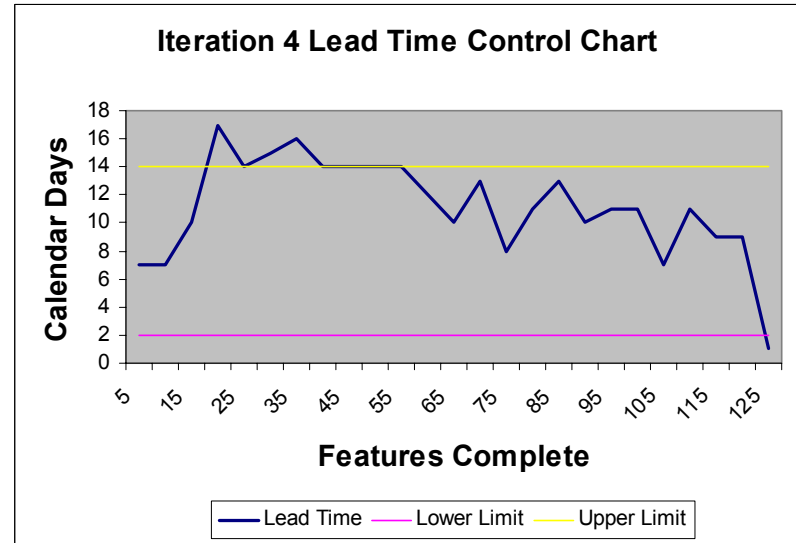
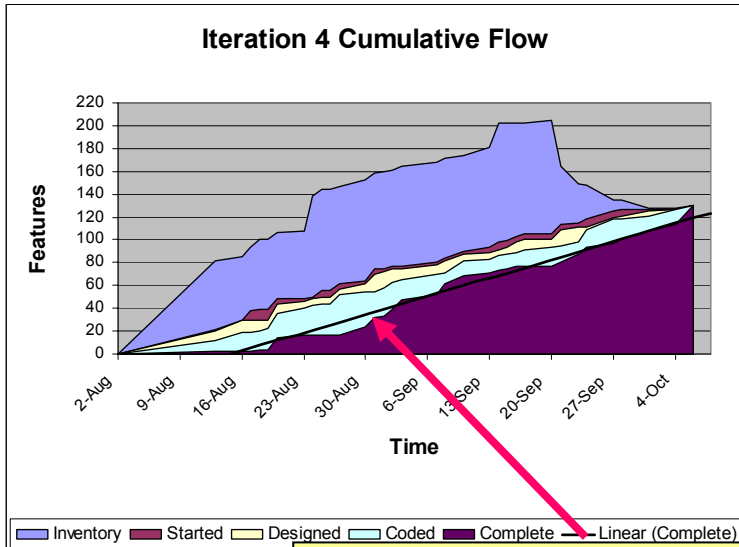
But...



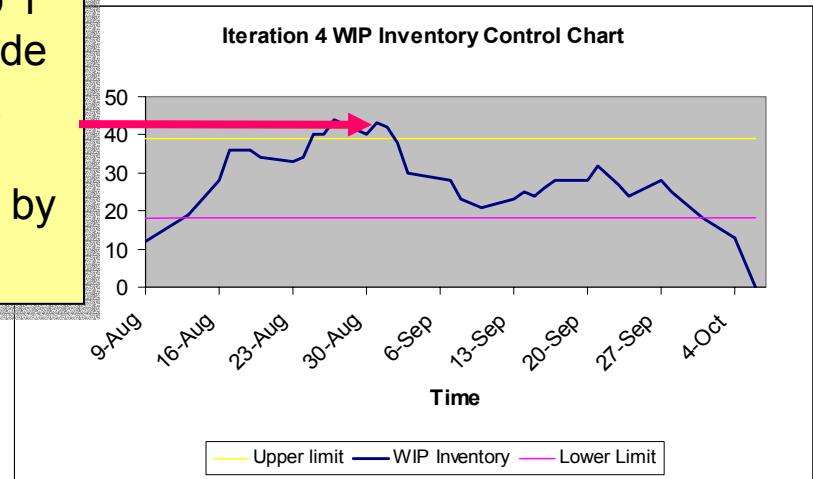
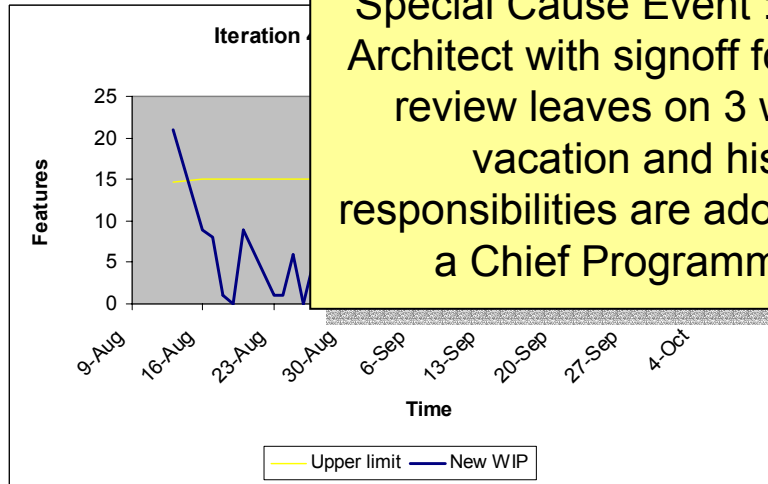
Using Control Charts with CFDs



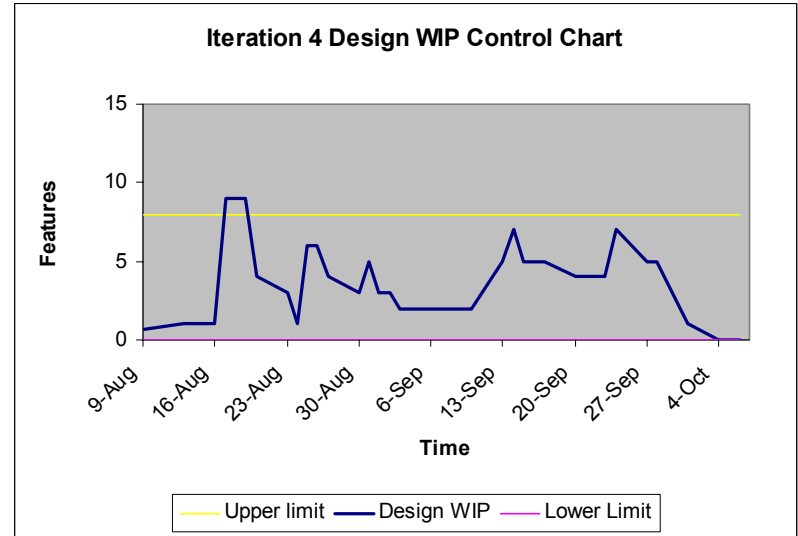
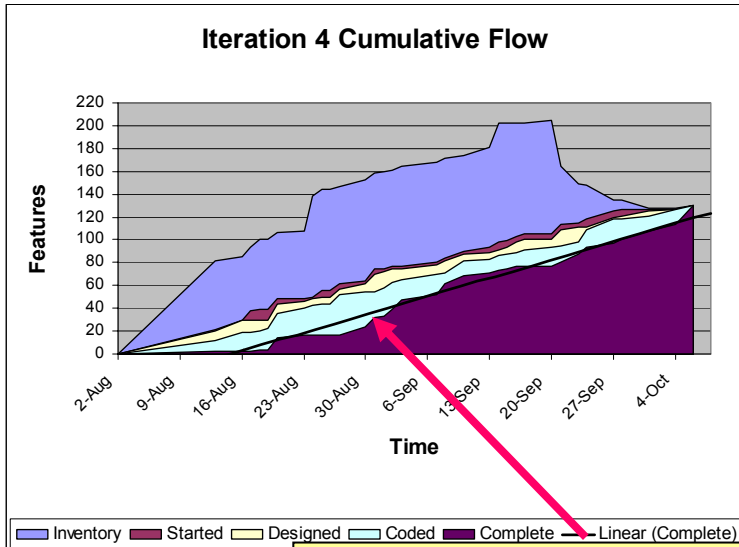
More Real Project Data



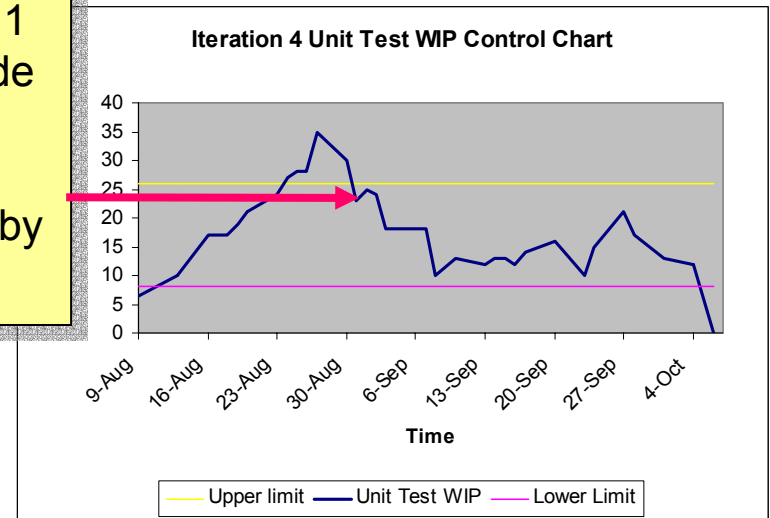
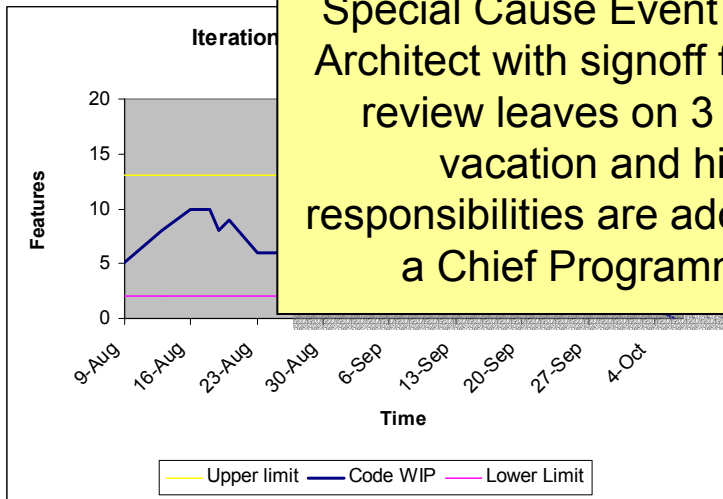
Special Cause Event : Sep 1 Architect with signoff for code review leaves on 3 week vacation and his responsibilities are adopted by a Chief Programmer




More WIP Control Analysis



Special Cause Event : Sep 1 Architect with signoff for code review leaves on 3 week vacation and his responsibilities are adopted by a Chief Programmer

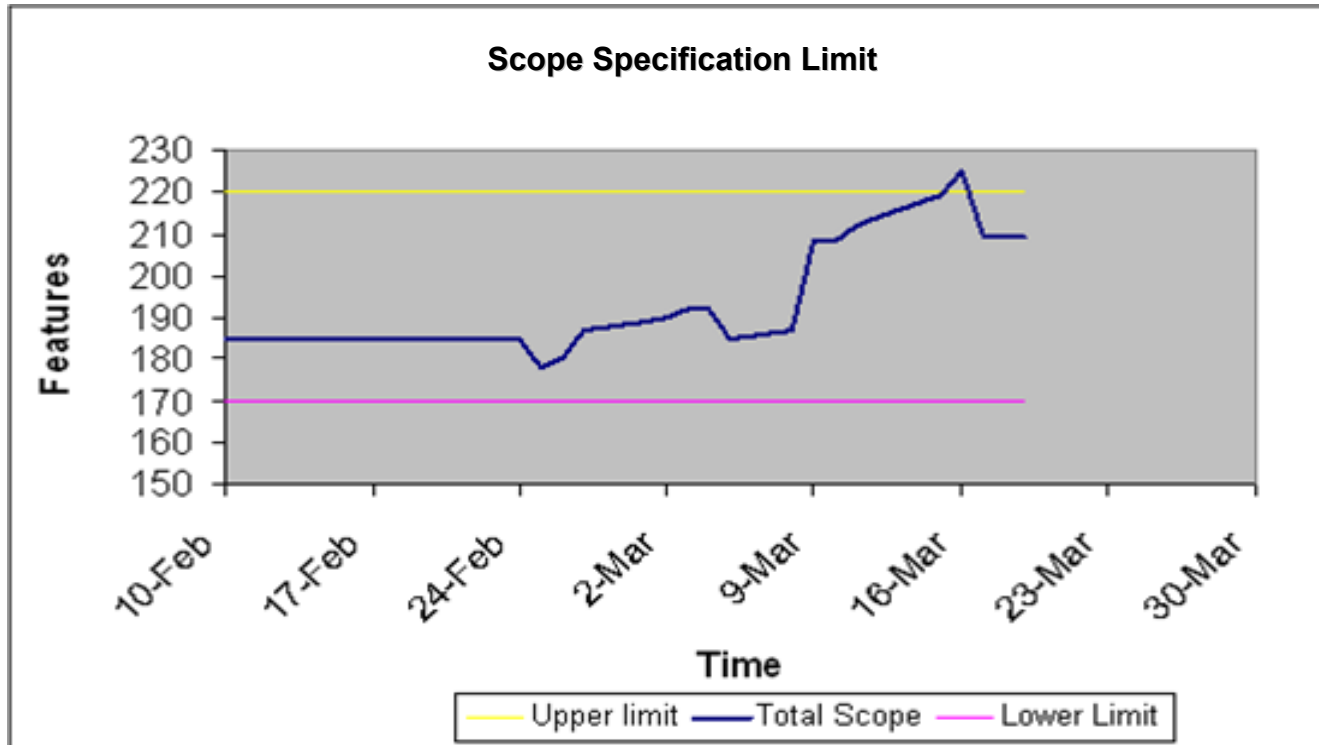


Wheeler's Maturity Scale

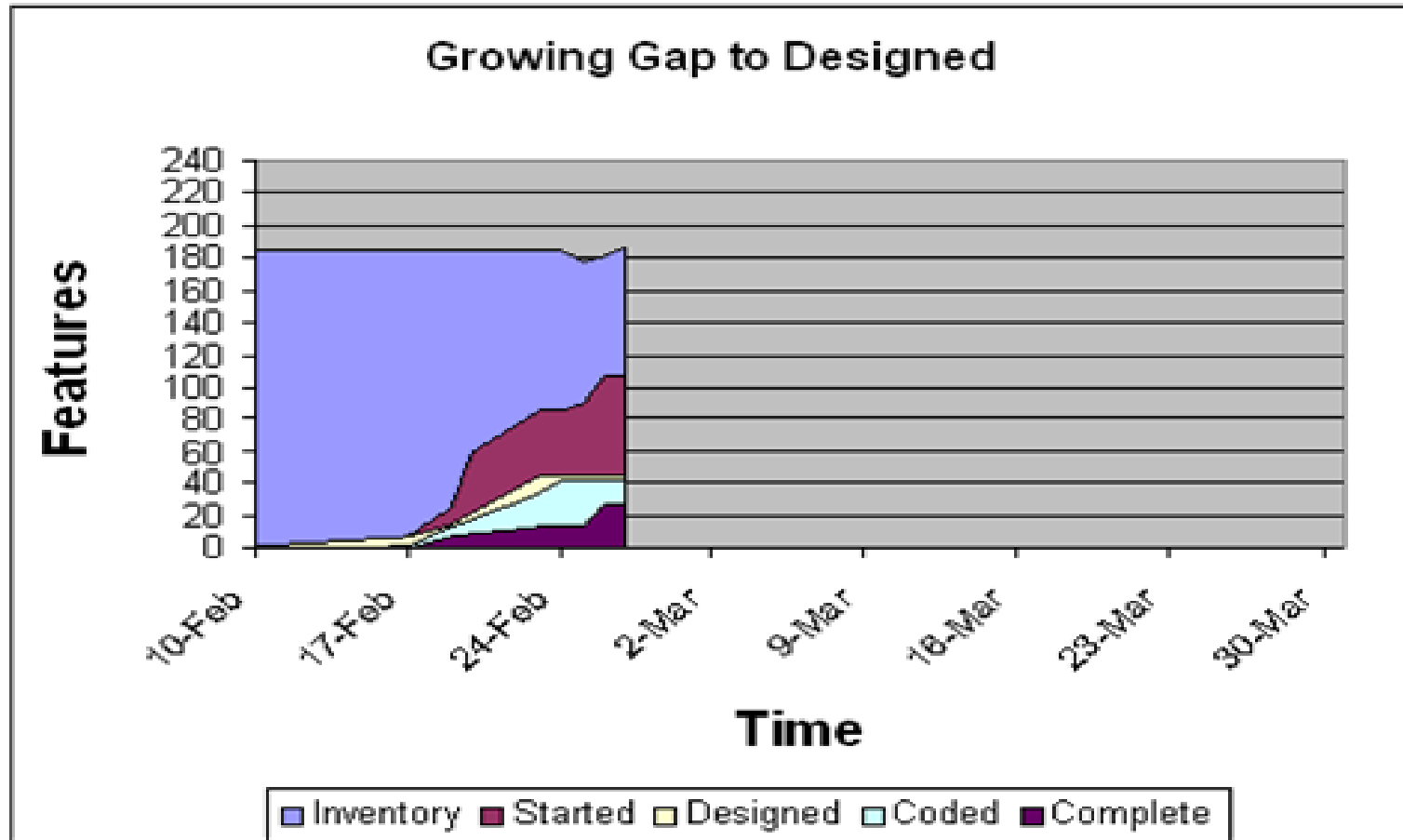
1. Report Cards
2. Process Adjustment
 - Consistent operation, feedback loop
3. Process Trial
 - Test new process to gain historical data and control limits
4. Extended Monitoring 
 - Multiple charts to see which provide best indicators
5. Continuous Improvement



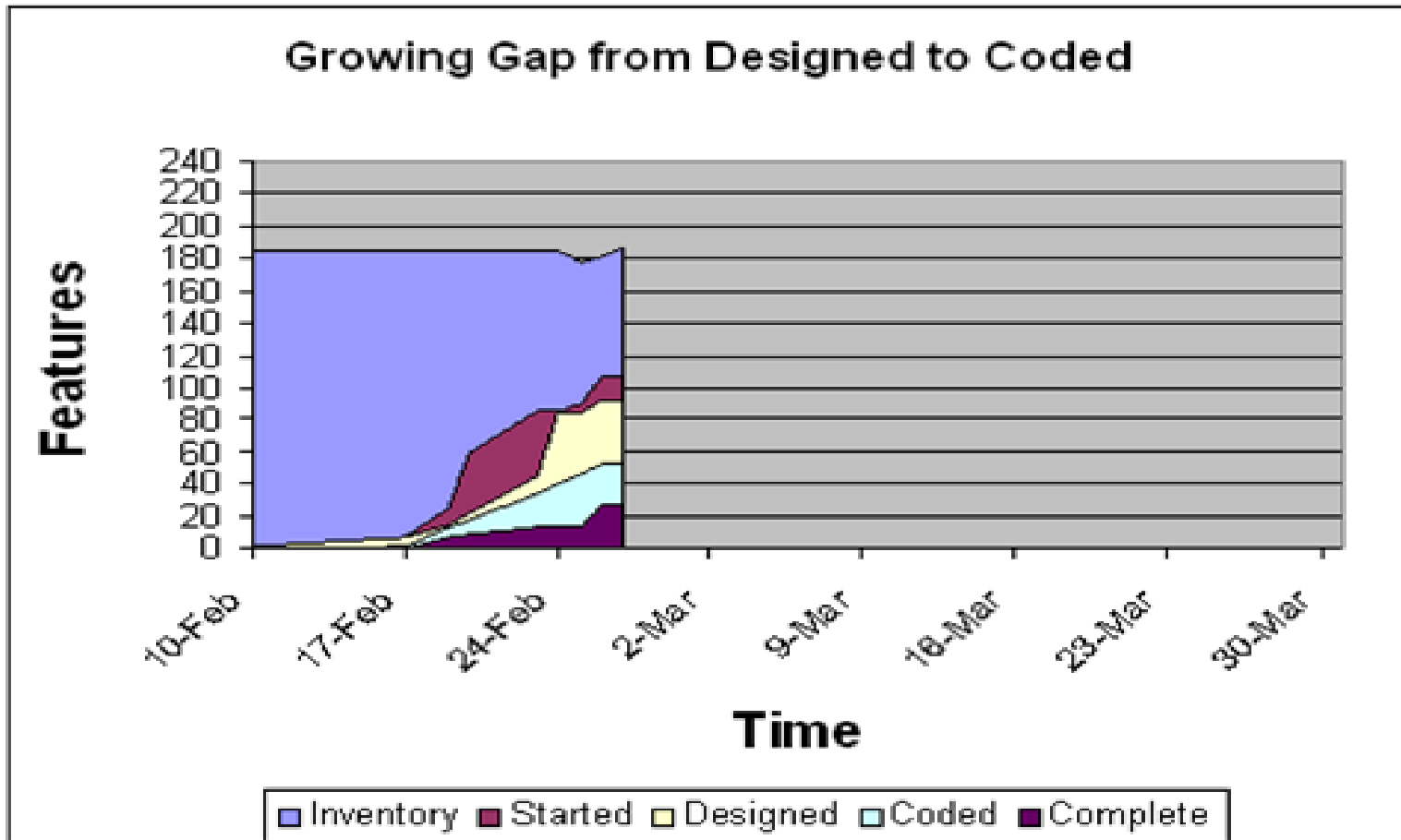
Buffering Scope Uncertainty



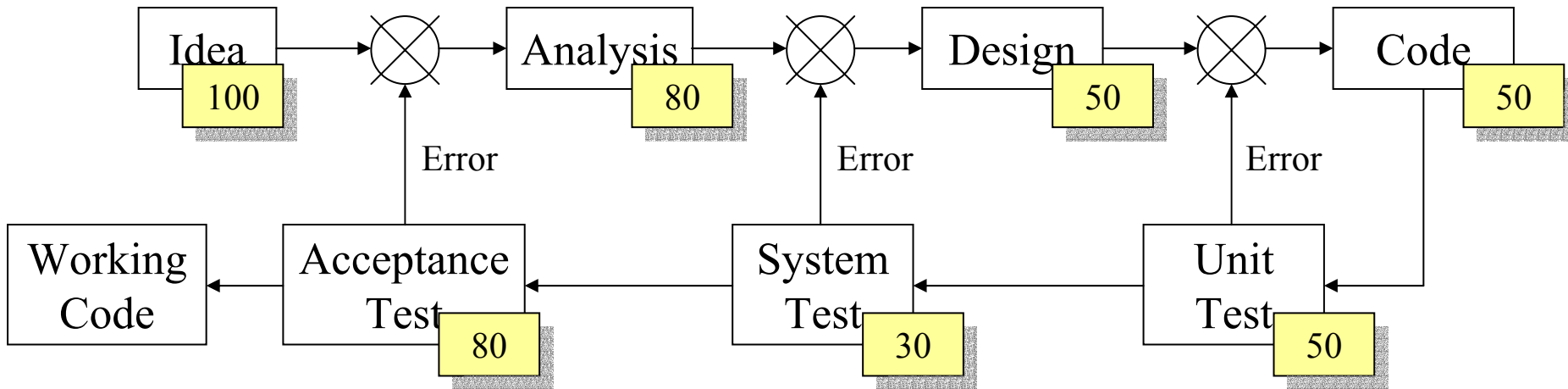
Requirements Uncertainty



Architecture or Refactoring



TOC – Drum, Buffer, Rope Solution

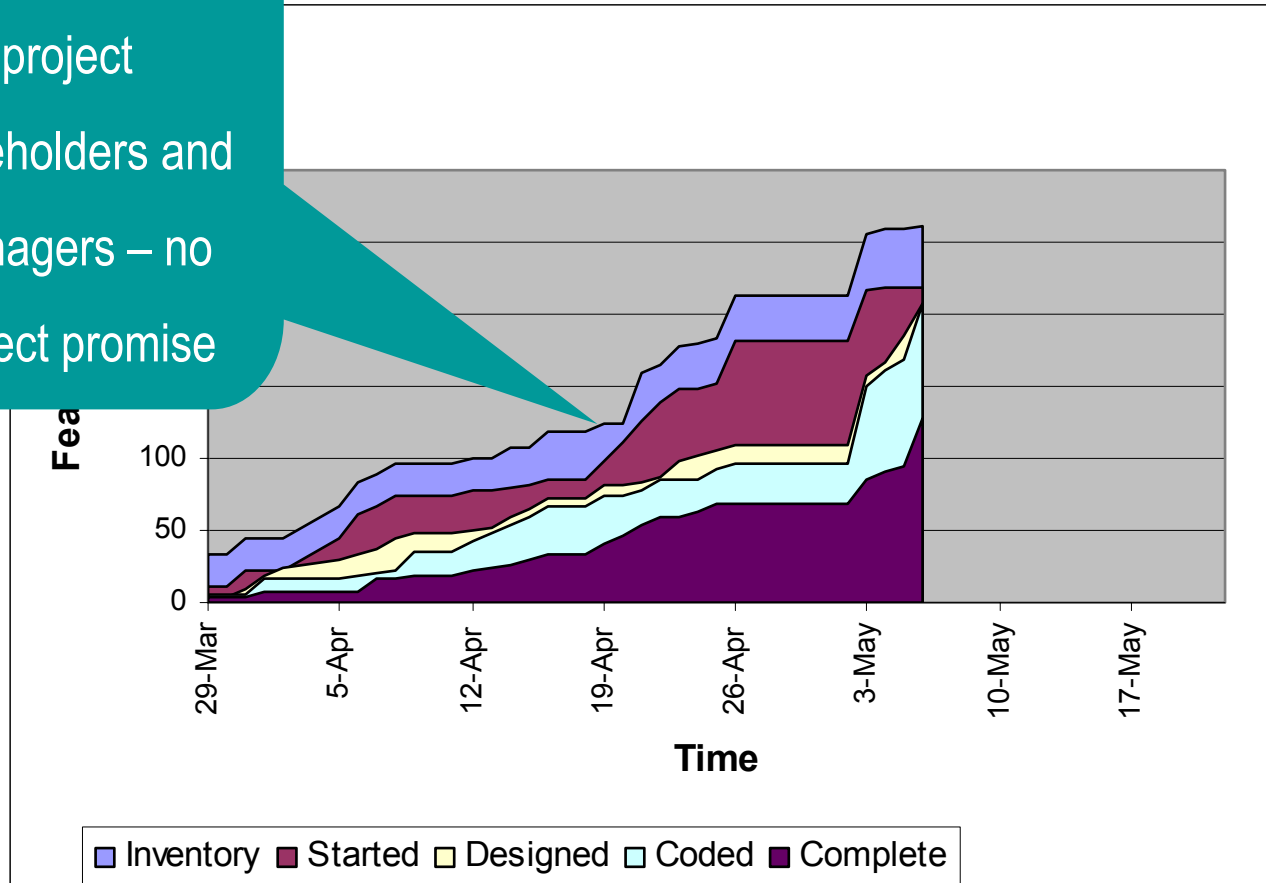


1. **Identify** - Current CCR is System Test – 30 per month
2. **Protect** - Testers relieved of all non-essential tasks, extra PMs assigned to complete administrative tasks, analysts assigned to future test plans
3. **Subordinate** - Requirements release restricted to 100 per quarter
4. **Elevate** - Plan to recruit 5 temporary staff immediately

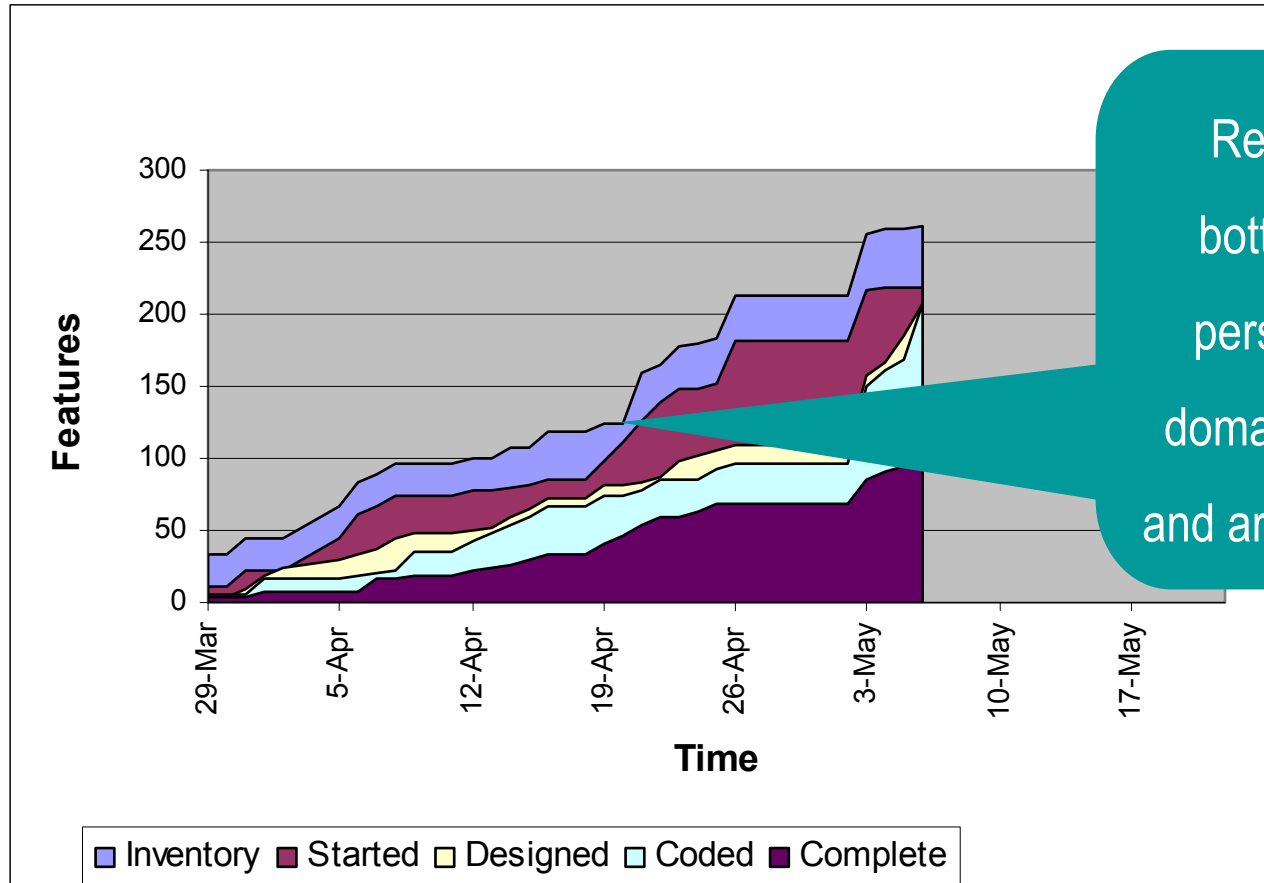


Just-in-time Domain Analysis

Problematic for project stakeholders and managers – no project promise



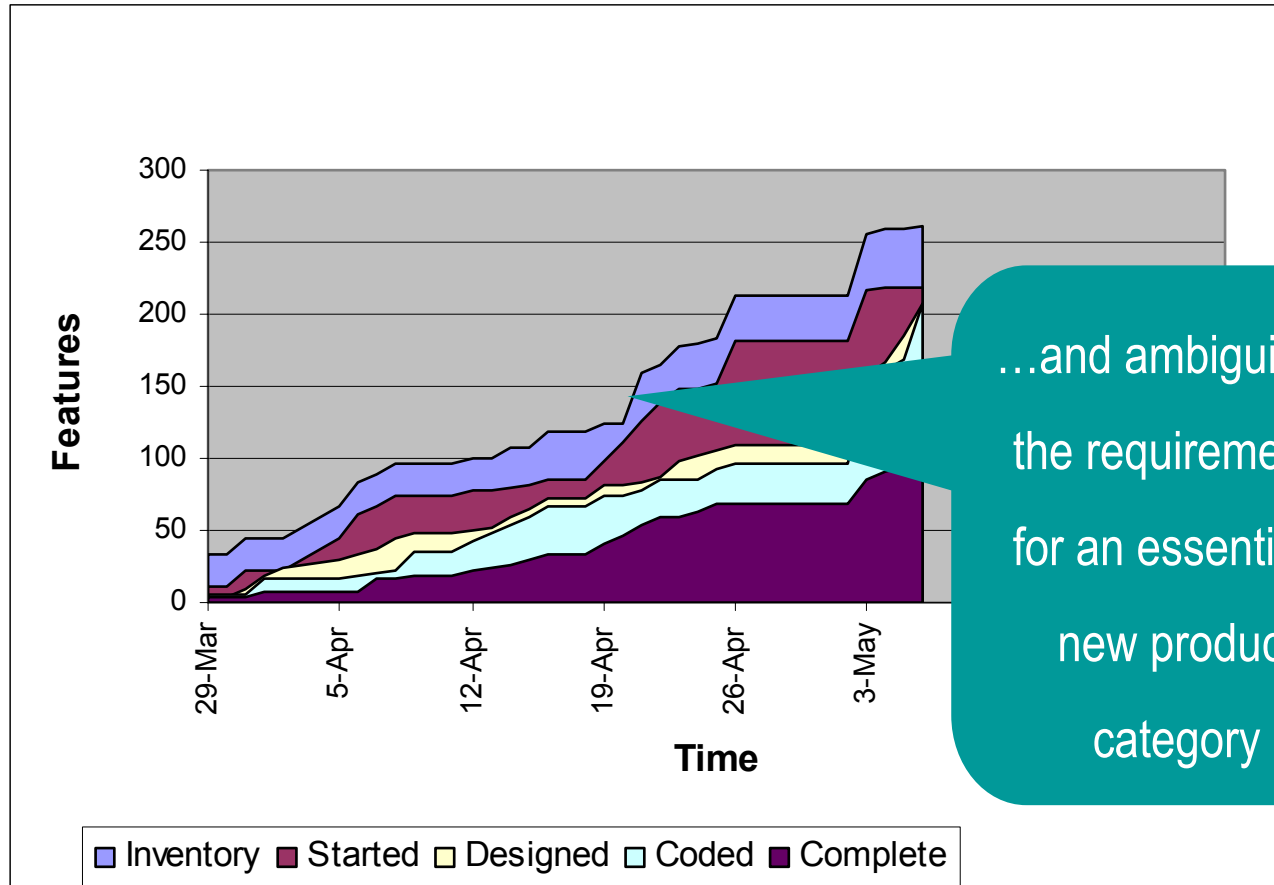
Just-in-time Domain Analysis



Reflected a bottleneck in personnel for domain analysis and architecture...



Just-in-time Domain Analysis



Development Summary

- Coad Method enables identification of fine-grained client valued functions
- Flow of value through the software lifecycle can be tracked with CFDs
- CFDs contain data which can be used in Control Charts
- CFDs contain information which can enable a TOC Drum-Buffer-Rope solution
- Use of SPC is experimental and unproven



Agenda

- Development Methods
- Project Management
- Requirements, Architecture, Components, and Distributed Development Programs



Variation in Project Management

- Separate out special cause from common cause
- Project management is about special cause variation
- Common cause system requires a theory of software engineering – not domain neutral project management
- Buffer for variation
- Aggregate to reduce buffers and tighten schedules

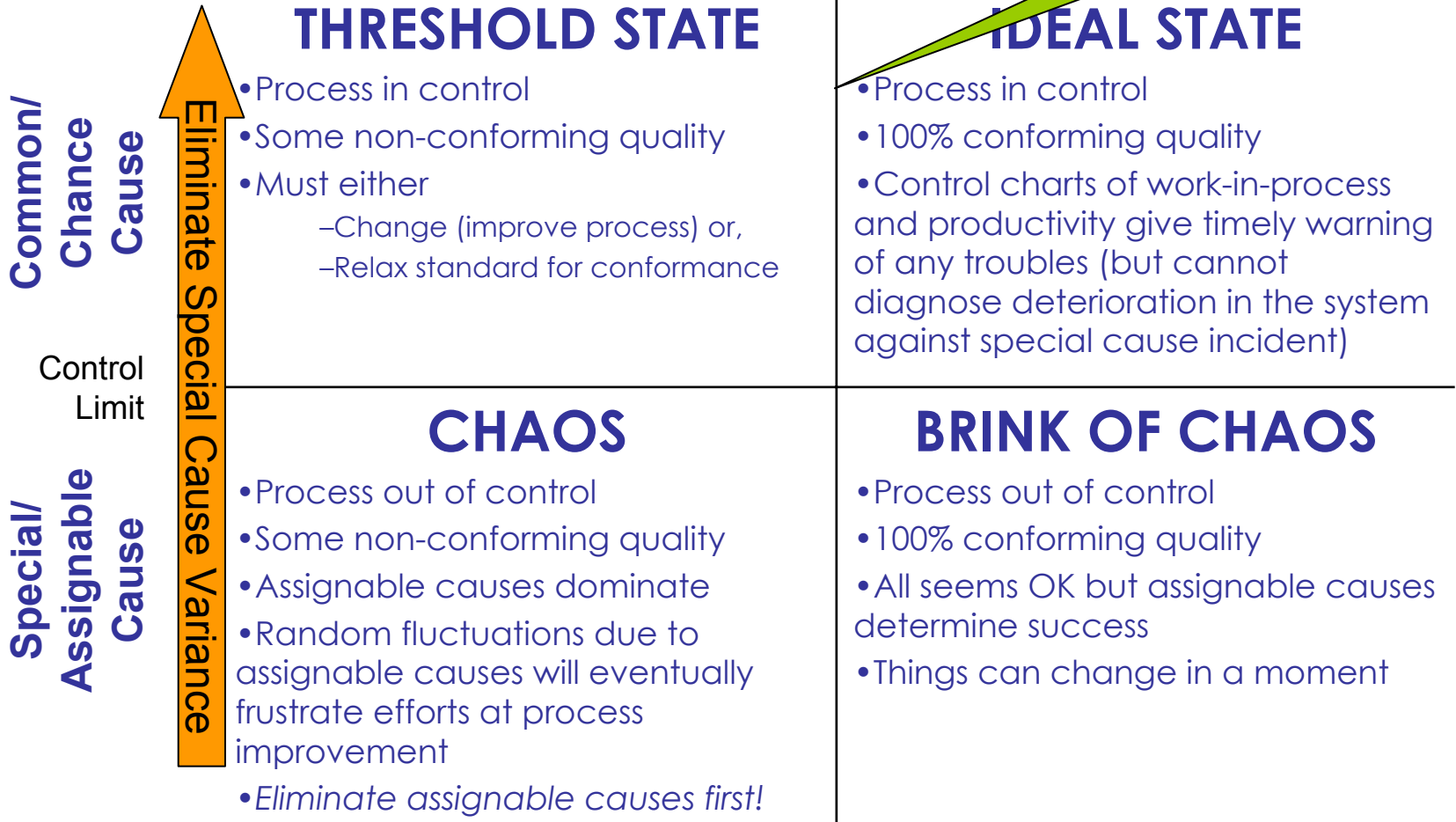


Variation and Wheeler's 4 States

On time,
On Budget,
Agreed Function

Non-Conformant Quality Spec Limit Conformant

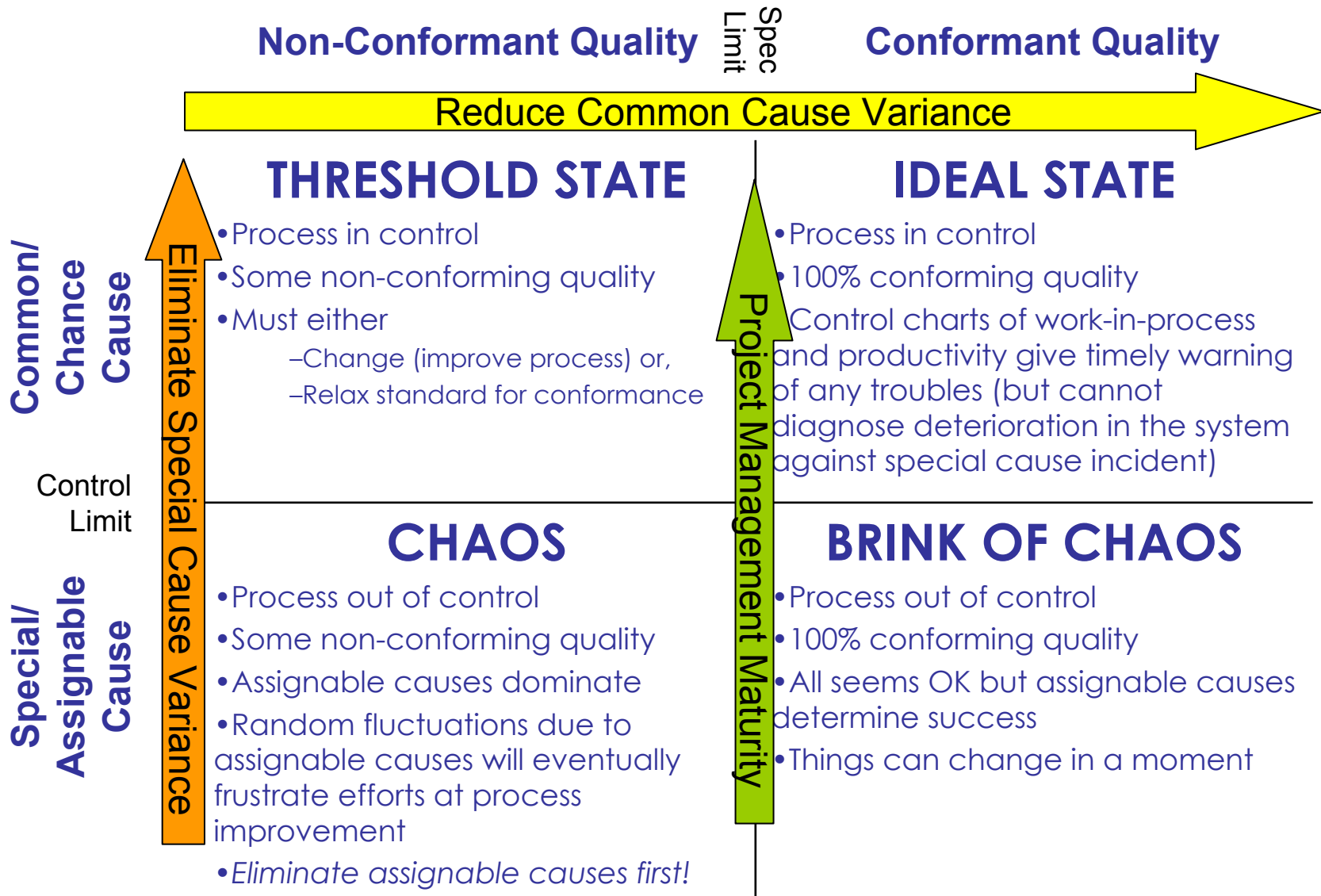
Reduce Common Cause Variance



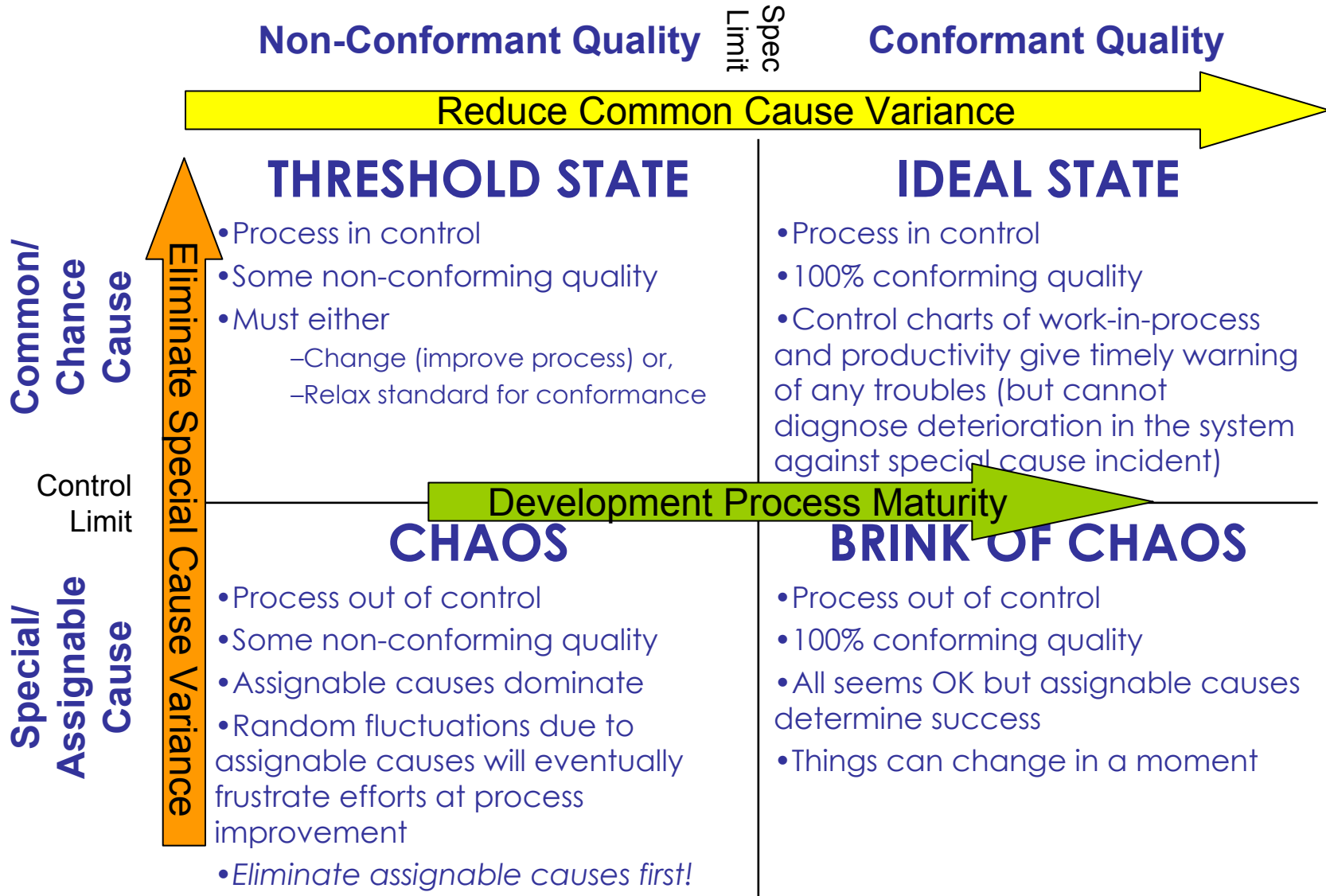
Eliminate Special Cause Variance



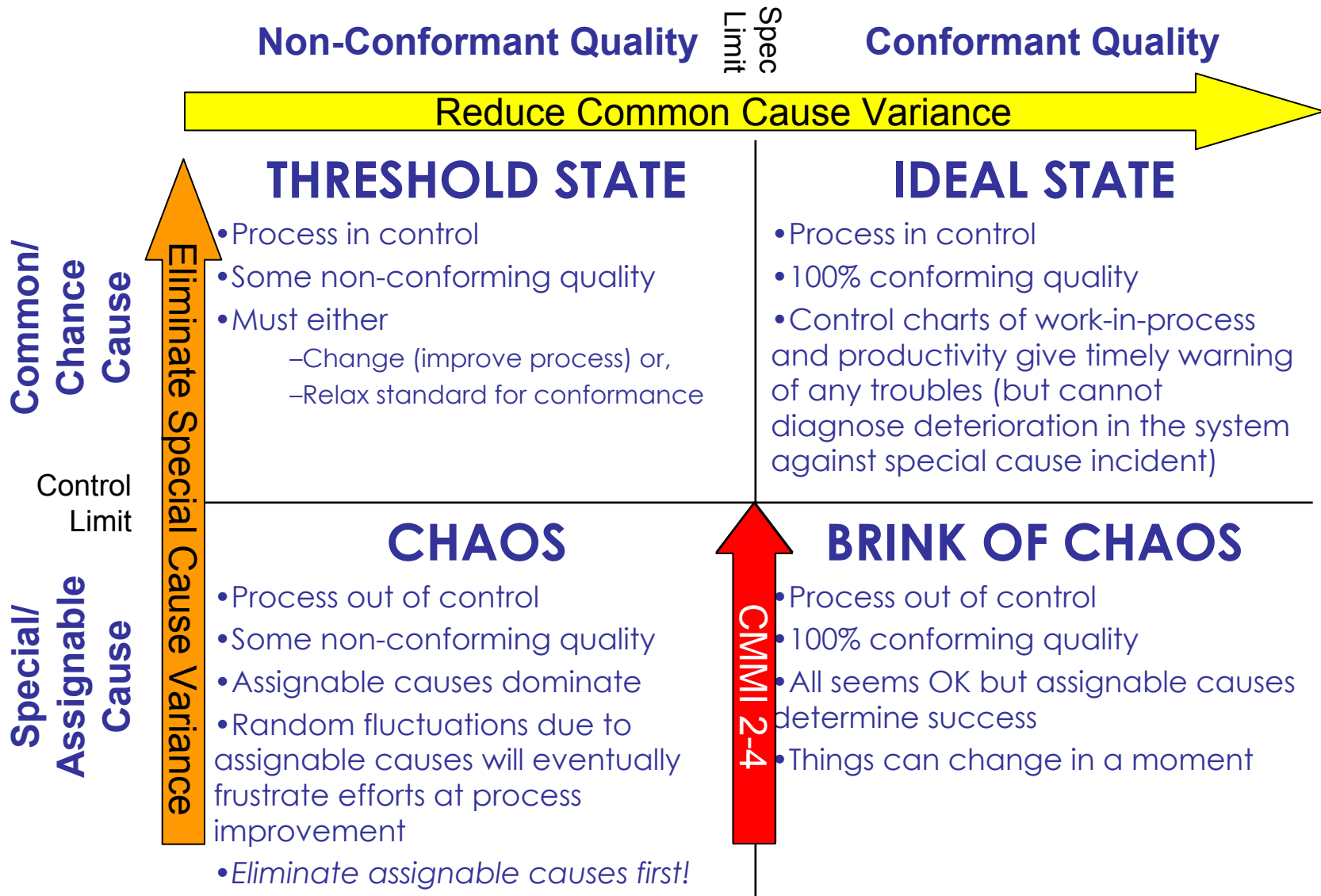
Variation and Wheeler's 4 States



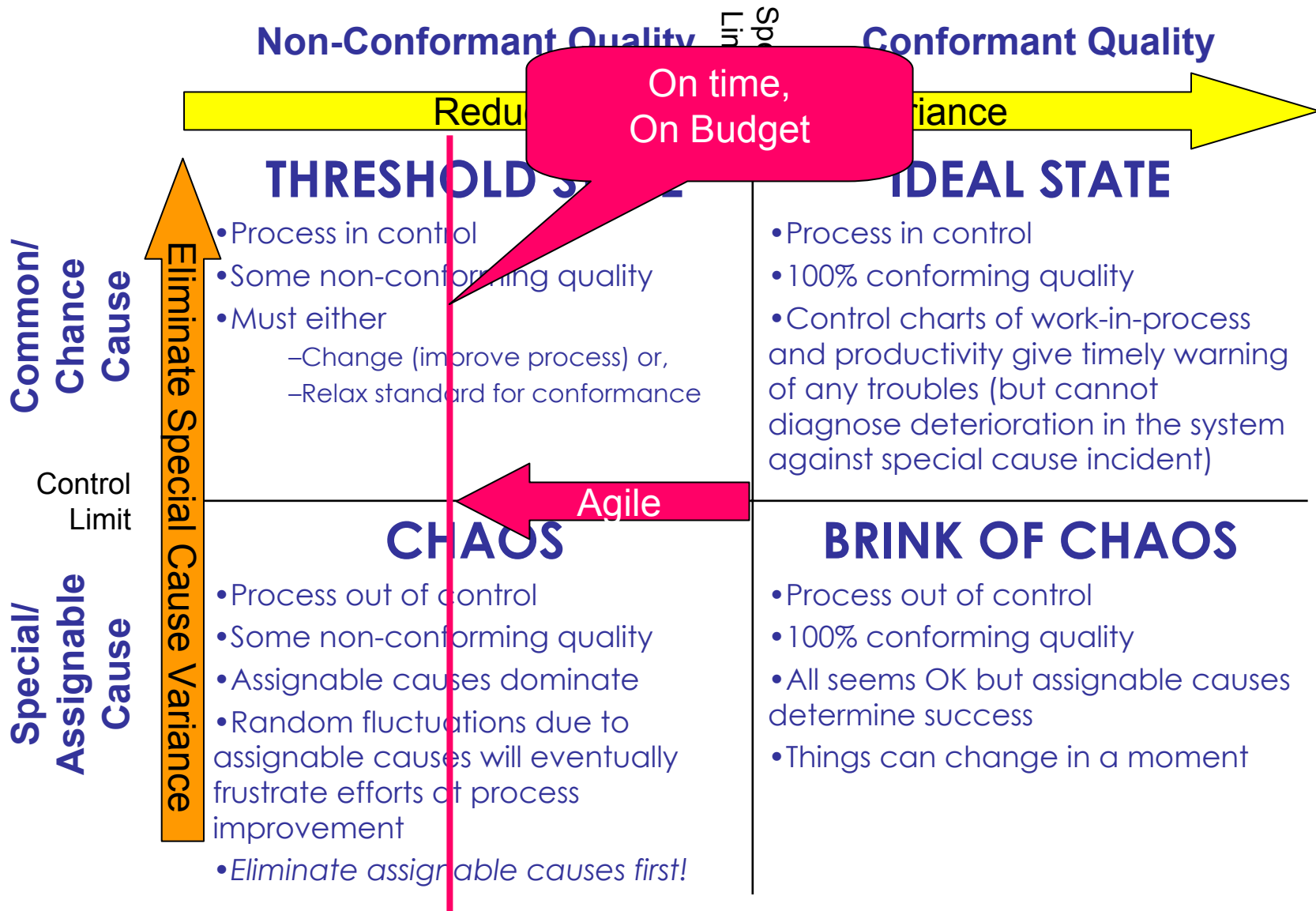
Variation and Wheeler's 4 States



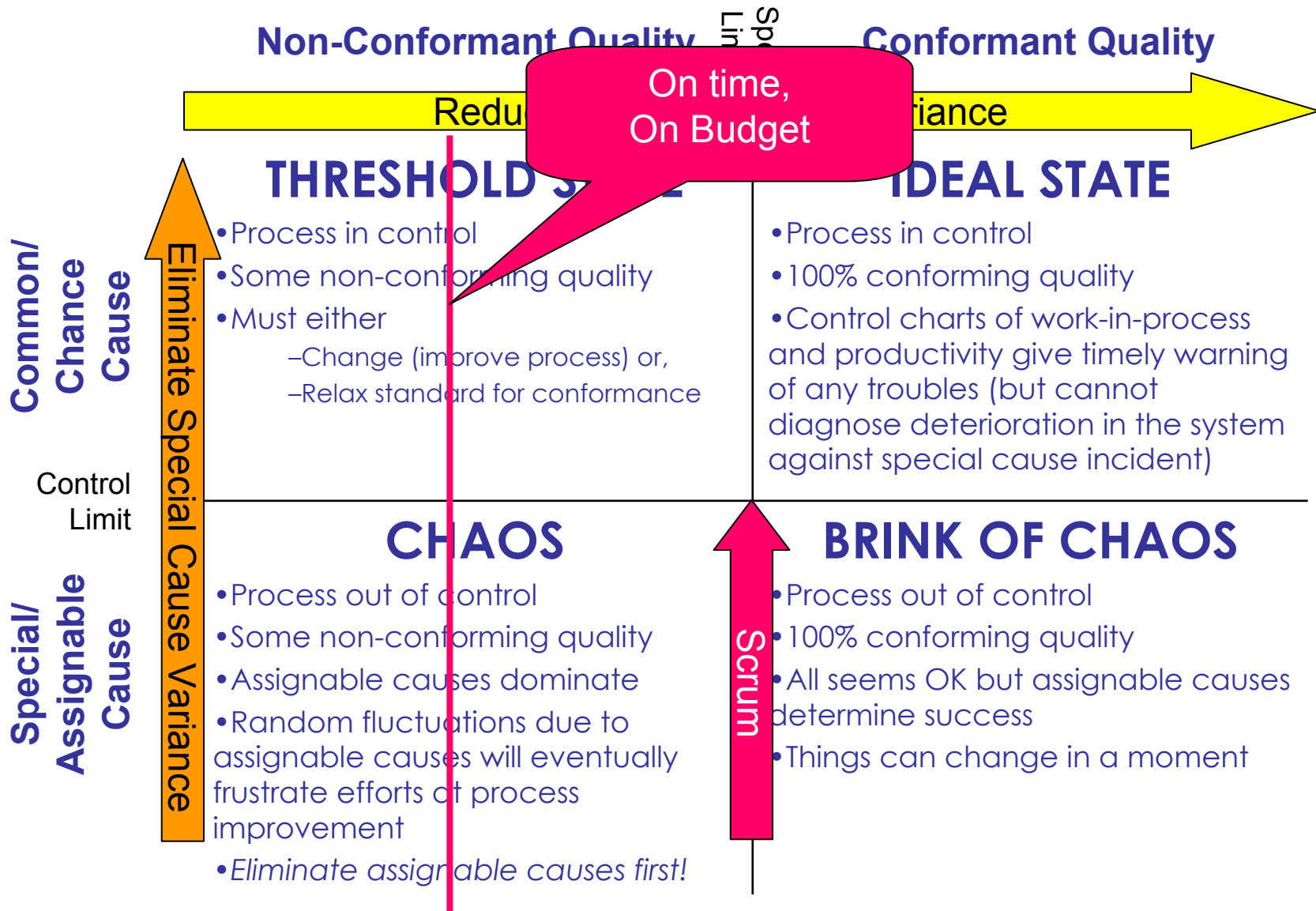
Variation and Wheeler's 4 States



Variation and Wheeler's 4 States



Variation and Wheeler's 4 States



Uncertainty Buffers

Well Understood



Poorly Understood



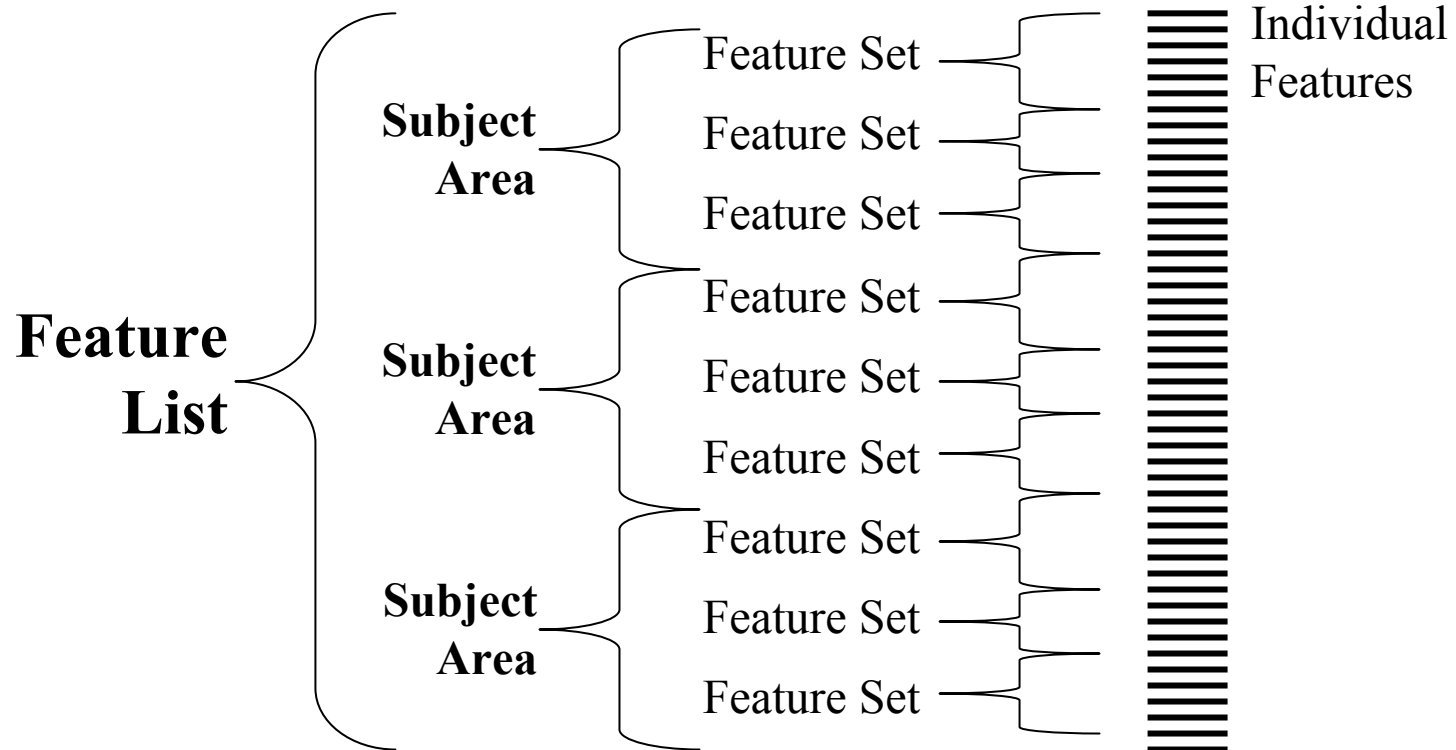
Build a Feature List

- Prioritize Features
 - 0 to 5 how important to be “in” release
 - 0 to 5 how problematic if left “out” of release
- Group Features into Sets and Sets into Subject Areas
- Groupings filtered by Release will be used for scheduling

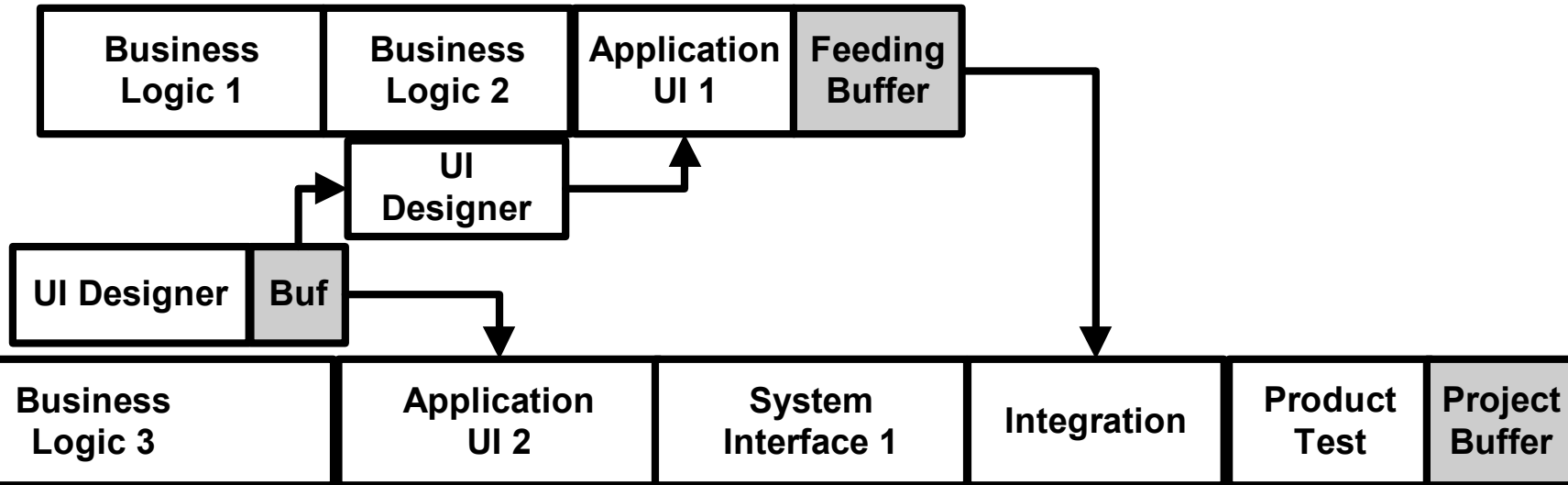
Feature / Feature	Release 1			Release 2			Release 3		
	IN	OUT	TOTAL	IN	OUT	TOTAL	IN	OUT	TOTAL
Set the Milestones for a Feature	5	5	10	5	3	8	5	5	10
List the Features for the Project/Release	5	5	10	5	5	10	5	5	10
Set the CPW for the Feature	4	3	7	5	5	10	5	5	10
Set the Subject Area for the Feature Set	4	3	7	5	5	10	5	5	10
Set the Feature Set for the Feature	3	3	6	5	5	10	5	5	10
List the Virtual Team members for the CPW	4	2	6	4	4	8	5	5	10
List the Feature Completion Dates for the Release	3	1	4	4	4	8	5	4	9
List the Feature Completion Dates for the CPW	3	3	6	4	4	8	5	5	10
Total the Features for a Product	3	1	4	4	1	5	5	5	10
Total the number of open issues in the Issue Log for a given	1	1	2	3	1	4	5	5	10
List Change Requests for the Release	1	1	2	2	2	4	4	5	9
List the Subject Areas for the Product	2	1	3	3	2	5	4	4	8
List all Feature Sets for the Subject Area	2	1	3	3	2	5	4	4	8



Aggregation of Features



Critical Chain Schedule



- Schedule based on Feature Set groupings
- Buffers aggregated across many Features
- This example has UI Designer as system constraint



From Variation to Chaos

Common/
Chance
Cause

Variation

- Control Charts
- Buffer Management / Critical Chain

Foreseen Uncertainty

- Identifiable and understood
- Distinct
 - Specific Recovery Plan
- Risk Management
 - Mitigation / Contingency Plan

Unforeseen Uncertainty

- Stable assumptions and goals, but...
- Unanticipated interaction of system elements
- Iterative Planning
- Monitor of early warning of lack of understanding of system

Chaos

- No stable assumptions or goals
- Research projects
- Iterate continually
- Parallel development (alternatives)

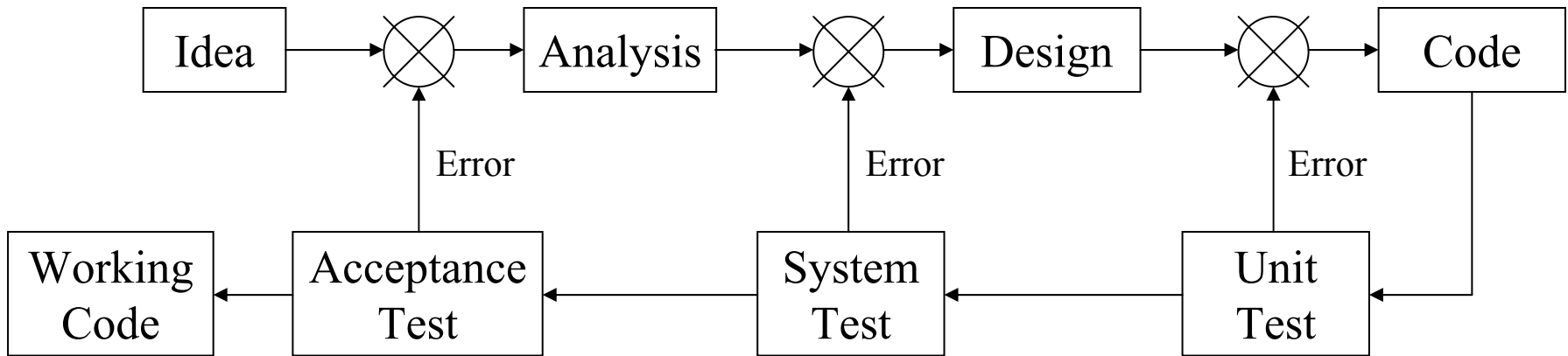
Understood
Market

Not Understood
Market

Special/
Assigned
Cause



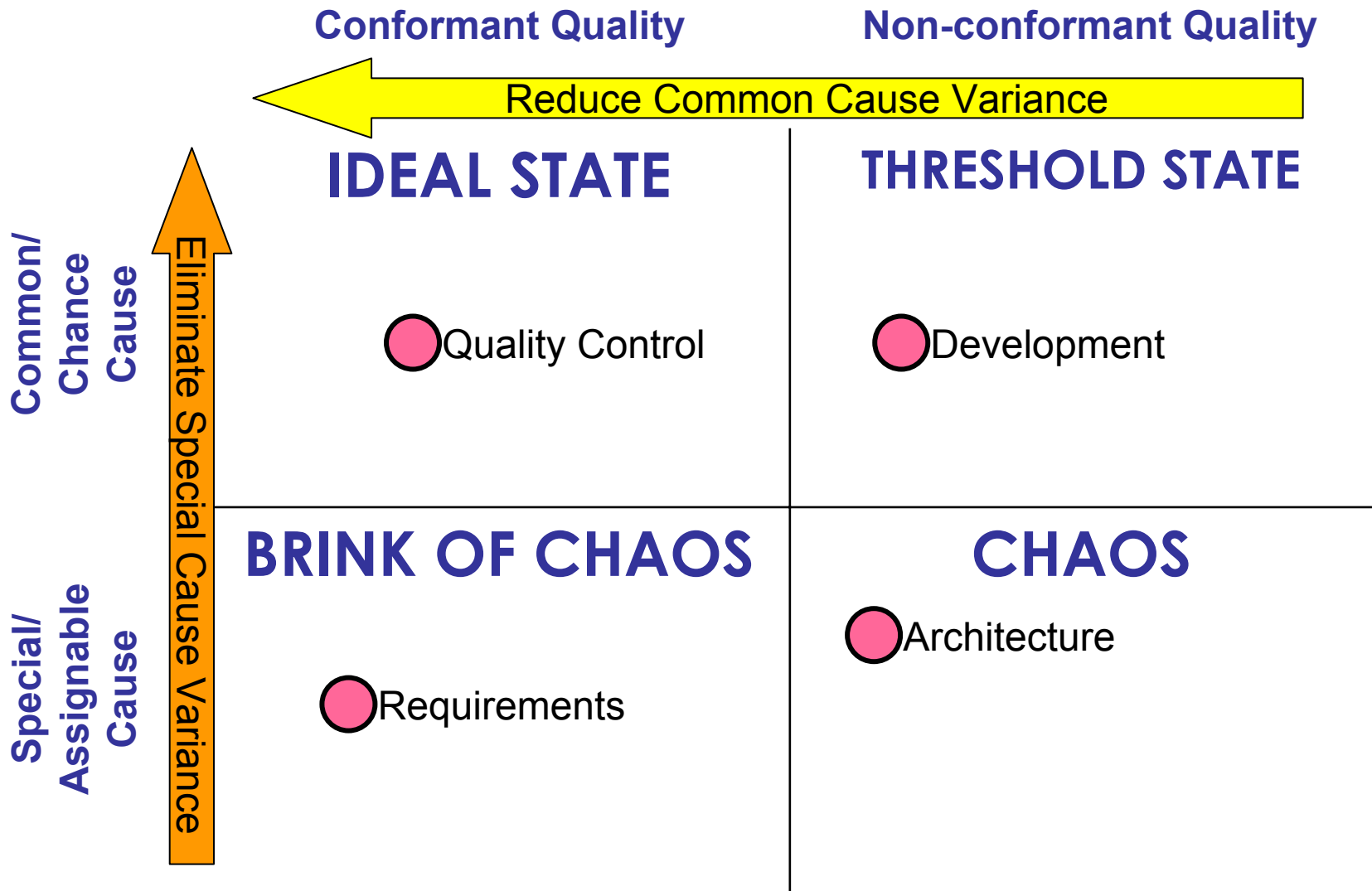
System Variability



- Understand the variability in each process element
- Does it deliver what was asked for within the specification limit? (on-time, on-budget)
- Does it experience special cause variation – issues unresolved hitting critical path or unmitigated risks which materialize?
- How reliable is input to next system step?



Value Chain States of Control



Project Management Summary

- Wheeler's 4 States provide a mental model for understanding variation in software engineering & project management
- Critical Chain can be used with Feature Driven Development to build variation resistant schedules
- A system is only as good as its most variable element, focus attention on improving the weakest link



Agenda

- Development Methods
- Project Management
- Requirements, Architecture, Components, and Distributed Development Programs

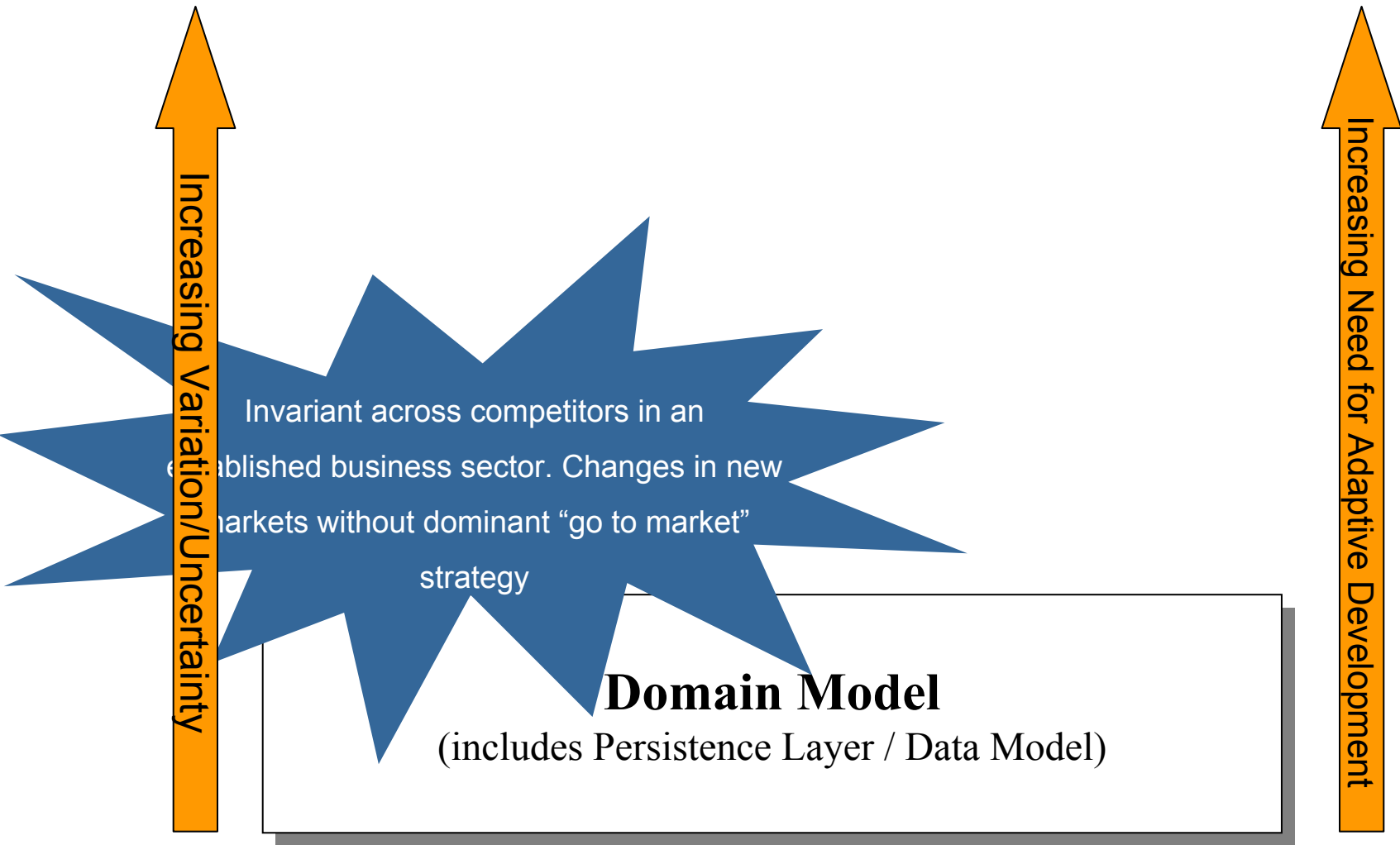


Architecting for Variation

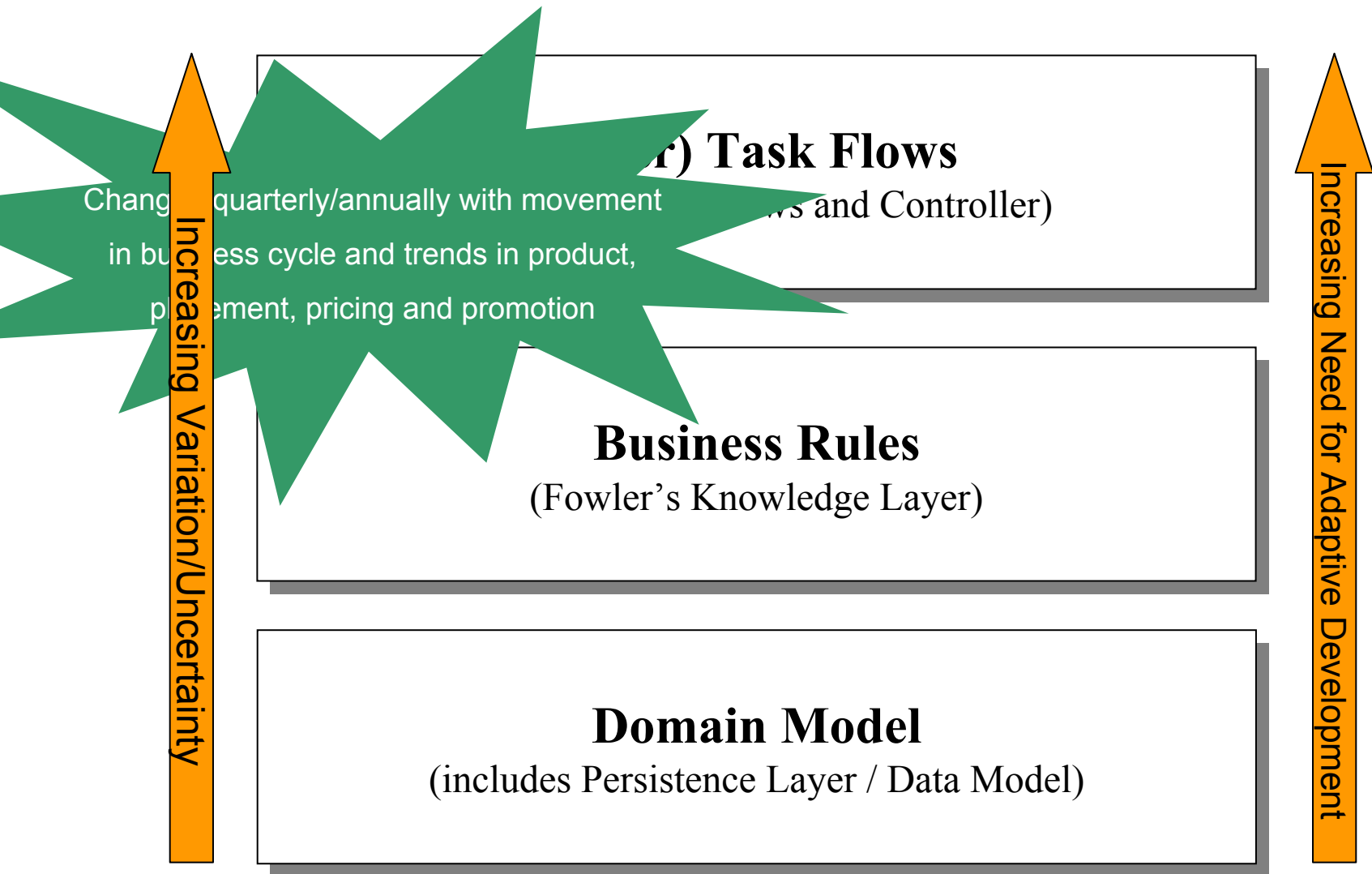
- Understand variation/uncertainty in requirements and partition requirements accordingly
- Partition architecture to align with variation in requirements
- Adopt architectural methods which allow postponement of design decisions and accommodate variation
- Distribute components by variation partitioning
- Enable different processes in distributed teams according to anticipated variation in component requirements



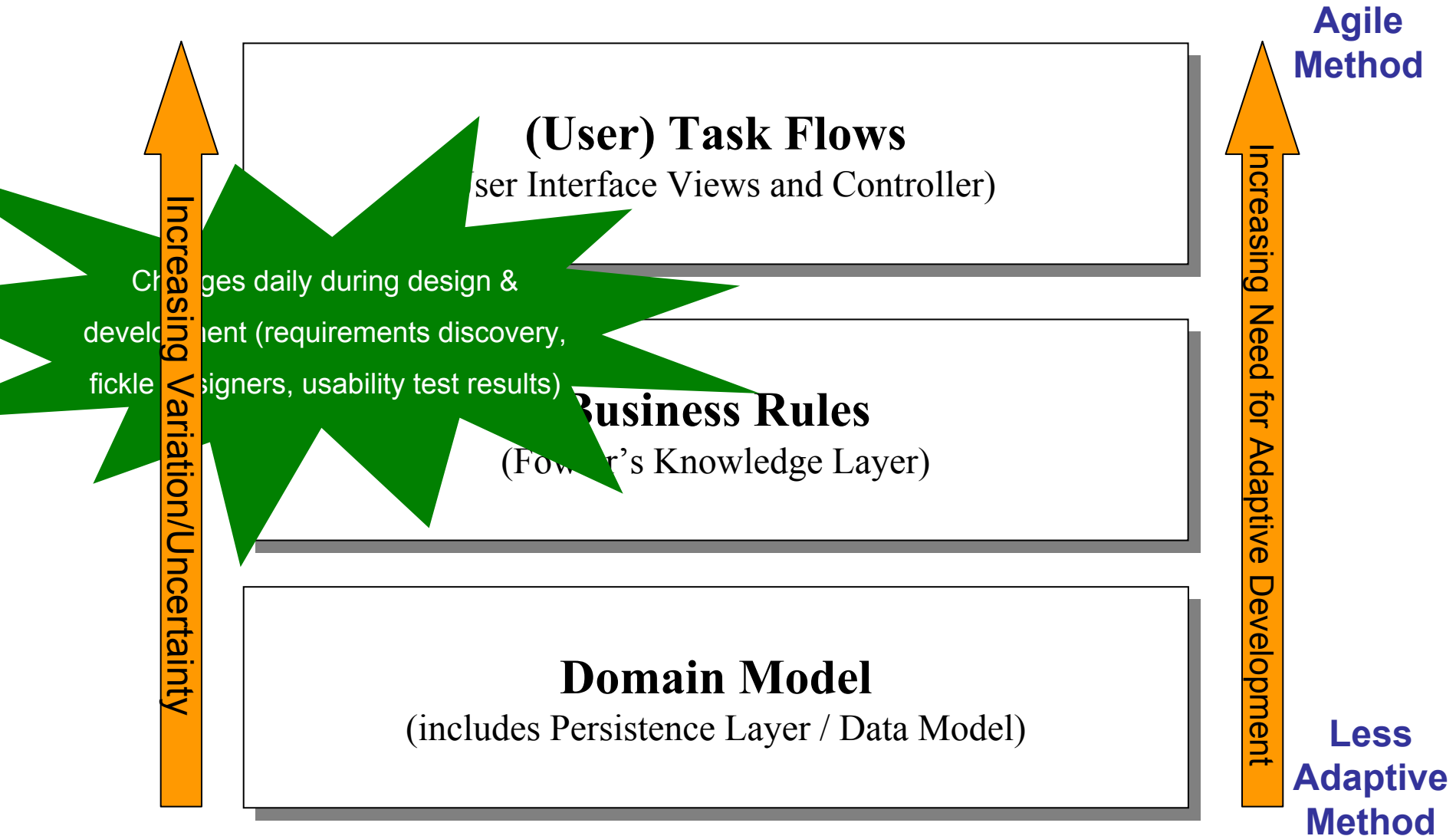
Architecting for Variation in Requirements



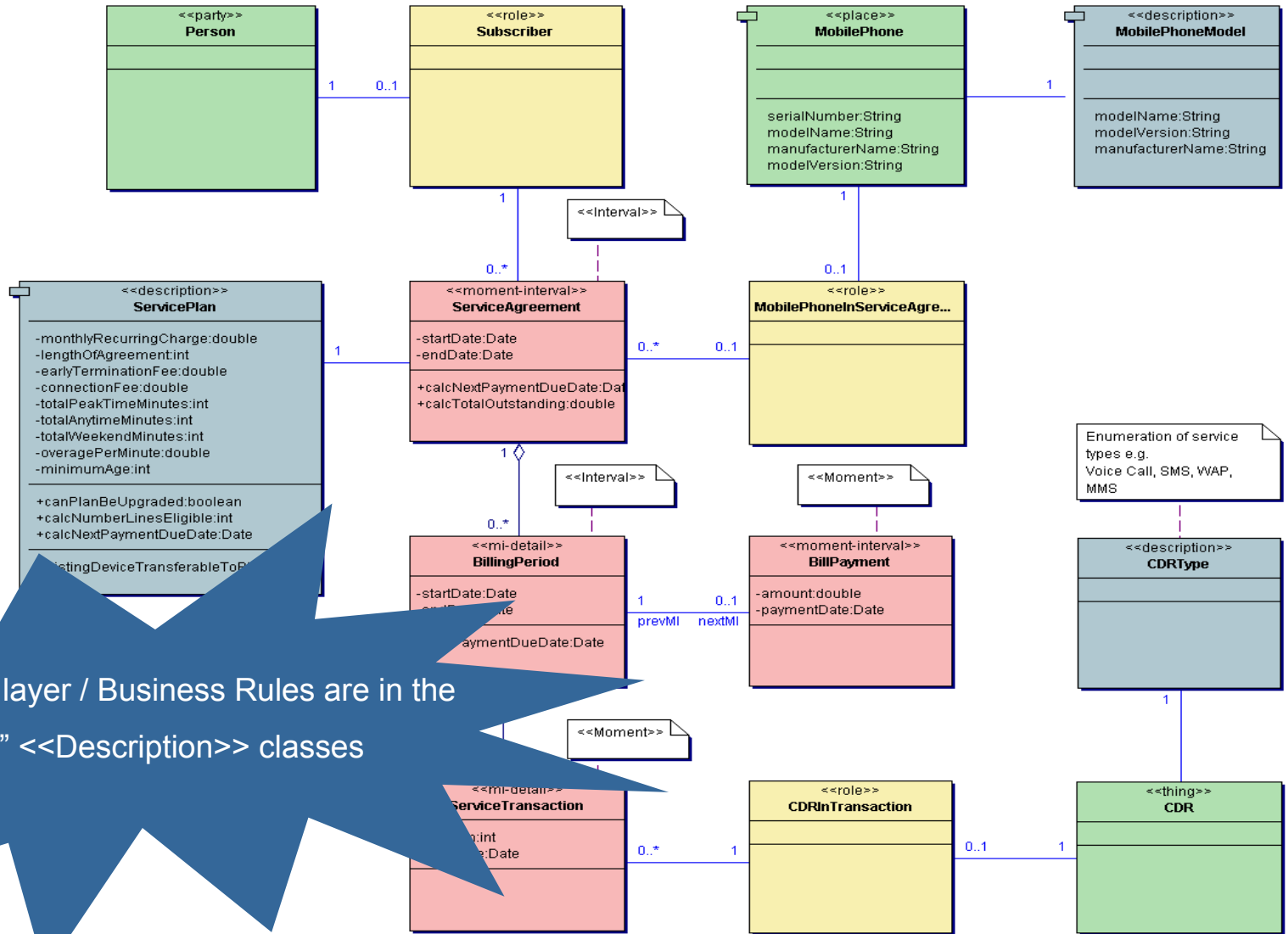
Architecting for Variation in Requirements



Architecting for Variation in Requirements



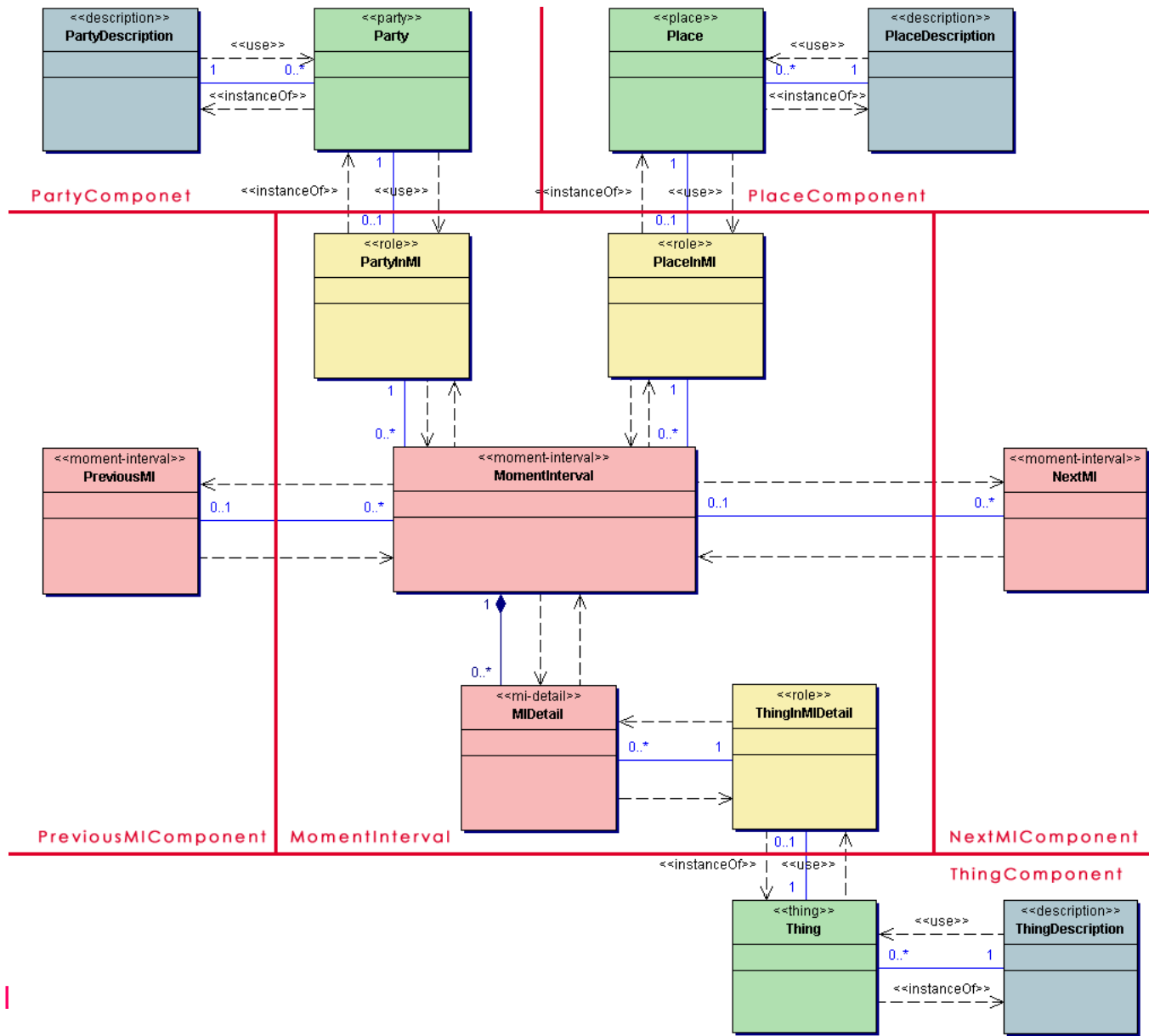
Wireless Telecom Domain Model



Knowledge layer / Business Rules are in the "blue" <<Description>> classes



Postponed Component Definition

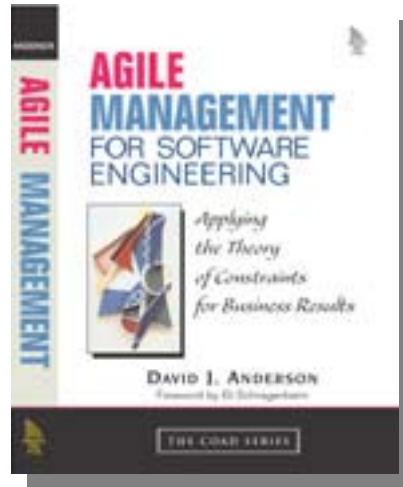


Architecture Summary

- Reinertsen – partition by variation
- Business Rules (Ross/Von Halle) partitions requirements by variation
- Partition 3-tier architecture (Domain, Business Rules, Screen Flows)
- (optionally) Use different processes on different layers
- Distribute components by variation
- The Coad Method enables distribution and postponement of component definition to absorb variation



Contact Details



David J. Anderson
dja@agilemanagement.net
<http://www.agilemanagement.net/>

