INTEGRATING AGILE SOFTWARE DEVELOPMENT WITH:

- SIX SIGMA
- BUSINESS PROCESS MANAGEMENT
- CONFIGURATION MANAGEMENT

Presented By:

Jason Tice - Vice President - Asynchrony
Email: jason@jasontice.com - jason.tice@asolutions.com
Twitter: @TiceThoughts
Agenda

- Background
- Analysis Methodology
- Agile & Six Sigma
- Agile & Business Process Management
- Agile & Configuration Management

Questions / Comments:
Just speak up, Twitter or Email (afterwards)
Tweet Me: @TiceThoughts

Want these slides (for notes / mark-up) - get them now at:
Learning Objectives

• Define the following Enterprise disciplines and explain why organizations use them:
  – Six Sigma
  – Business Process Management (BPM)
  – Configuration Management (CM)

• Highlight knowledge, skills and abilities of practitioners of other Enterprise disciplines

• Discuss how each of these disciplines can interact with Agile software development
Additional Objectives

• Build relationships

• Experiment with a few analysis / decision tools

• Successfully complete “Operation After Lunch”
  – Share and leverage ‘your’ knowledge
  – 3 Exercises (Pro/Con 3 to 1, SWOT, Visualization)
  – Data collection to advance ideas
  – Follow Up (using your data)
  – Acceptance Criteria: Nobody falls asleep

• Continuation of work
  – Thoughts shared will be used to spawn future ideas
  – Watch for an upcoming blog post (week of 8/20/2012)
My Background:
- Did Microprocessor/FTL design & testing
- Did Software Development (Assembly/Device Drivers, then OO)
- Learned how to write tests (order agnostic)
- Optimized and Standardized Build Systems (process tendencies)
- Overcame multiple CM Disasters
- Became a Six Sigma Green, Black Belt
- Did BB projects to drive financial system improvements
- Went to business school - started doing BPM
- Joined an Agile Consultancy (Asynchrony)
- Learned Agile & XP (write the tests first)
- Did Agile development (learned to write other kinds of tests)
- Supported an Enterprise Agile Transformation
- Linked Agile Dev, Six Sigma, BPM & CM via Enterprise Architecture (EA)
- Leading the ‘tech side’ of a large scale EA practice
Background

- **Enterprise Background:**
  - Very Large IT enabled global logistics Enterprise
  - Multiple Process Centric Systems
    - Administered as separate programs
    - Duplicative systems / brittle interfaces
    - "Transformation" needed but struggling to "run"
  - Systems Centric Development patterns
  - Pre-dominance of waterfall SDLC
  - Stakeholders highly influenced by industry trends
  - Rigorous CM requirements - information assurance focus
  - Developed Six Sigma (CPI) competencies (active COE)
  - Simultaneous in-sourcing of BPM & Agile dev (new disciplines)
  - Ground everything via and into the Enterprise Architecture
Analysis Methodology

- Patterns - ways to integrate Agile with other disciplines
- Services - activities & information needs to facilitate integration
- Messaging - communication strategies & activities for change management to promote success
Agile & Six Sigma

Image Source: Dilbert - Scott Adams

© 2012, Asynchrony Solutions, Inc. All rights reserved.
Six Sigma: Background

- Improve the quality of process outputs through rigorous quantitative (statistically significant) controls to identify and remove causes of defects and variability.
- Created by Motorola, popularized thanks to Jack Welch at GE.
- Source of solid quantitative data (both anticipated and achieved) to execute Enterprise strategy.
Six Sigma Practitioners

• **Names & Titles:**
  - Apprentices / Junior Green Belts
  - Green Belts
  - Black Belts
  - Master Black Belts

• **Required KSAs:**
  - Process definition/decomposition
  - Root cause analysis
  - Measurement System design/implementation
  - Statistical/Mathematic analysis
### Flavors of Six Sigma

<table>
<thead>
<tr>
<th>DMAIC</th>
<th>DMADV</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Define the problem</td>
<td>• Define what needs to be done</td>
</tr>
<tr>
<td>• Measure the 'as-is'</td>
<td>• Measure 'Critical To Quality' (CTQs) attributes</td>
</tr>
<tr>
<td>• Analyze to find the root case</td>
<td>• Analyze high level designs or COAs</td>
</tr>
<tr>
<td>• Improve targeting the root cause</td>
<td>• Design detail based upon the best design / COA</td>
</tr>
<tr>
<td>• Control the future state</td>
<td>• Verify intended results will be achieved</td>
</tr>
</tbody>
</table>

Targeted at improving existing processes & products

Targeted at optimizing new processes & products
Kaizen: Any Similarities?

Improvement through small incremental and iterative changes

Potential benefits:
- Awareness of Kaizen benefits & techniques widely understood
- "Kaizen" to “Agile" linkages may resonate with new stakeholders

Image Source: Taking the First Step with PDCA
Patterns: Agile & Six Sigma

- Agile linked to tollgates
- Agile throughout the Six Sigma lifecycle
- Six Sigma to define/drive "patterns" vs. "projects"
Pattern: Agile linked to tollgates

Lean / Six Sigma

- Define
- Measure
- Analyze
- Improve
- Control

Agile Software Development

- Pre-Iteration Planning Meeting
- Iteration Planning Meeting
- Develop/Test/Build
- Deploy to Integration
- Stories piloted during “Improve”
- Demonstrate Solutions to the Customer
- Document Lessons Learned from the Process
- “MEASURE” generates stories to identify information needs
- “IMPROVE” generates stories to provide improvement
- Monitor KPIs to achieve process “Control”

First Attempt: Some benefit, but revealed opportunities for improvement
Pattern: Agile linked to tollgates

Implementation:
- Black belt was a member of the Agile development team
- One war-room, whole-team approach, full-team pairing, etc
- Single set of stories (including Six Sigma stories), single Kanban board

Benefits:
- Understand the team’s purpose - BB ensure business case was understood
- Knowledge / skill transfer - interested developers got primer on Six Sigma
- Development team had "rock solid" quantitative acceptance criteria

Challenges:
- Work imbalances & idle states - tollgates were too long
- Value stream was not optimized across the team
- Questionable benefit of mixing of extremely diverse skill sets
  - Are software devs really going to be Black Belts and vice-versa?
Pattern: Agile throughout the Six Sigma lifecycle

Lean / Six Sigma

- Define
- Measure
- Analyze
- Improve
- Control

Agile Software Development

- Pre-Iteration Planning Meeting
- Document Lessons Learned from the Process
- Demonstrate Solutions to the Customer
- Develop/Test/Build
- Deploy to Integration
- Iteration Planning Meeting
- Tollgate related development stories & technical tasks

Next Attempt: Improved utilization & value-stream
Pattern: Agile throughout the Six Sigma lifecycle

During or After "Define":
- Decide on an appropriate tech stack / engineering Spikes
- Build dev-test-demo-prod environments (what about PaaS/SaaS?)

During or After "Measure":
- Common measurement services, API, dashboards & reports
- Determine how to get data, create mock services (required for env)

During "Analyze":
- Use analysis data to draft stories and acceptance criteria
- UI / UX design work & dev team Spikes to prototype ideas

During "Improve":
- Implement documented user stories - receive customer feedback

During "Control":
- End-user training & mentoring, bug fixes, performance tuning
Pattern: Six Sigma to define/drive "patterns" vs. "projects"

Implementation:

- Six Sigma and tech staff work together to define an improvement strategy around a common technical problem.

- See if a specific problem in a project, can be applied to multiple projects?

- Six Sigma defines the quantitative measures (acceptance criteria) by which "Control" (ROI) can be achieved.

- Agile Dev teams implement the "pattern" to achieve the desired control.

- Means to execute strategies to promote common patterns and software assurance across a large organization / Enterprise.
Provide criteria for:

• **Measurement system requirements**
  - Implementation via logging
  - How much metrics data needs to be captured / persisted
  - Integration with external metrics services / software

• **What needs to be measured (KPIs)**
  - Agile Dev teams ensure the software can provide this data
  - Measurement requirements can drive design patterns

• **What are key performance thresholds**
  - Establish software / service SLAs (performance targets / acceptance criteria)
  - Requirements for alerting and/or notification features

• **How will required parties use metric data collected**
  - Bulk data exports for offline analysis
  - Performance dashboards / alerts within applications
  - Services needed to support external dashboards
  - Analytics within applications to identify improvements
Messaging For Six Sigma

Link process improvement goals to software dev activities

• Inputs to Agile dev - develop required Service Level Agreements (SLAs) to meet performance improvement objectives (service to reduce cycle time)
• Using ideas from Six Sigma to drive/confirm team improvements
  – Six Sigma Lite perhaps in Retros?

Messaging of Metrics:

• Six Sigma Metrics/KPIs - cycle time, number of defects/deviations, etc
• Agile Metrics - flow, velocity, escaped defects, etc
• Agile metrics should demonstrate progress toward achieving Six Sigma KPI

Use Six Sigma to synergize multiple projects and/or teams

• Targeted improvements around common 'patterns' or 'problems’
• Establish Enterprise SLAs to execute strategy and deliver results
• Measurable Enterprise level acceptance criteria spanning software dev activities
Exercise: Thinking About Six Sigma

- Create 2 groups per table
- Determine a facilitator
  - Whoever most recently celebrated their birthday

- List up to 3 pros & 3 cons based related to integrating Agile & Six Sigma
- As a group decide what the ‘key’ takeaway is
- Capture the key criteria - why selected?
- Lightning sharing (from several tables)
Agile & Business Process Management

BPM

- Model
- Simulate
- Implement
- Execute & Control
- Optimize
Modeling a business process so that it can be orchestrated, executed, and optimized within a business process management system (BPMS) leveraging existing IT assets and services within the Enterprise.

--Enterprise specific definition

What this really means:

- Model the process
- Make the process executable
- Wire the process to services and/or create tasks
- The BPMS drives the process
- The BPMS analyzes/optimizes the process
- The BPMS can analyze the 'process ecosystem’
Value Proposition of BPM

Additional recapitalization of sunk costs in existing IT assets

- Use systems / services 'as-is' to orchestrate processes
- Use the BPMS to "bolt on" functionality and business rules to systems that cannot be changed

Improve process execution for "spreadsheet over Email" processes

- Reduce cycle time
- Understand and better distribute process load
- Improved awareness and dispatching

Additional value propositions

- Empower additional staff to improvements to IT enabled processes
- Determine which processes (and activities) support value-add
- Generate & analyze data about how processes impact each other
- Understand which processes (and steps) are under/over loaded
BPM Myths

BPM allows for "codeless" development
• Some truth
• Modeling and process configuration become tool supported activities
• Integration of existing systems/services to the BPMS can be highly technical
• Software developers are better skilled at coding vs. using a BPMS

BPM allows for "real time" process improvement
• Can be true with several important notes
• Assumes full process execution (control of state) in the BPMS
• Required service/data/interactions are available
• There is value to testing a process
BPM Practitioners

- Understand process and process decomposition techniques
- Establish roles, groups, responsibilities specific to processes
- Understand information needs and capabilities of roles and groups
- Familiarity with value-streams and how to optimize them
- Many have a strong metrics, process improvement (Six Sigma) background
- Technically inclined, but not necessarily software developers
  - Software developers shy away from BPM since it is tool driven
  - Software developers build or configure the interfaces/services/adapters to support BPM
What can be Agile about BPM?

• Customer focused - visual practice
• Iterative
  – May generate need for targeted / rapid improvements
  – A "sweet spot" for Agile development to complement BPM
  – Kanban has become popular with BPM teams
• Tooling allows for rapid development, testing, delivery
  – Assumes the BPMS has been configured and data is available
• Whole team approach
  – Allows a different group of staff to improve the value-stream
• Continuously improving for the future
  – Analyze/optimize processes to determine future improvements
• Similar traits/patterns to software development
  – Can have bugs, need to be tested, have a lifecycle, etc
Patterns: Agile & BPM

- BPM done prior to development
- BPM done during development
- BPM done after/without development
Pattern: BPM done prior to development

Benefits:
• Technical requirements for services / systems supporting the process
  – Information & data needs known up front
  – Contract-first service development
• BPM simulation establishes target SLAs
• Process model can be used as part of test strategies

Challenges:
• Working to avoid process-specific services
• Defining a small end-to-end process that adds value
• Establishing Enterprise criteria to justify BPM enablement
Pattern: BPM done during development

Benefits:

- Fast demonstrable ROI
  - Incremental process and software releases
  - May require much coordination (MRC between dev and BPM)
- Optimized value-stream
  - BPM and development activities kept in sync
- Maintain proper / desired architecture
  - Fix 'code' the right way vs. 'kludges' in the BPMS

Challenges:

- Two moving targets to manage
  - Agile Services/Software Dev + Process Model & Orchestration
- Coordination of releases and versions
- Managing different requirement sources
  - Potential for scope creep and/or waste
Pattern: BPM done after/without development

Benefits:
• IT capabilities & services stable throughout BPM lifecycle
• Provide new capabilities for legacy IT assets
• Implement improvements with reduced technical staff

Challenges:
• Required data or services may not be available
  – Potential impacts to desired outcomes
  – "Data Upload" upload pattern risks
• Managing "bolt on" functionality in the BPMS
  – Implications for changing BPMS tools
• Avoiding "automated yet over-complicated" processes
  – Putting capabilities in processes vs. services
  – Managing a collection of common sub-processes
  – Governance to ensure sub-processes are used
Services for BPM

• Define how systems and services need to interact with the BPMS
  – Information exchanges
  – Web service contracts (WSDLs, etc)

• Use BPMS simulation activities
  – Establish target SLAs for software/system components
  – Use BPMS simulation as part of acceptance testing

• BPM inputs to story prioritization
  – Process / value-stream analysis - simulation data & ecosystem data
  – Coordinate process needs with IT capabilities

• Provide standard dashboards for performance metrics
  – Determine if Enterprise KPIs are needed
  – Define metrics to manage the "process ecosystem"

• Define a standard API to interface with the BPMS
  – Establish process & task naming conventions
  – Standards about what does in the BPMS vs. in code
Messaging for BPM

Benefits integrating of BPM and Agile development

• Determine acceptance criteria for contracts and interfaces
• Simulation data defines target SLAs and test scenarios
• BPM simulations can complement user acceptance testing
• Link process improvement strategies & goals to specific software development
• Determine common services/components to improve multiple processes
• Enable more staff (non-developer) to improve value-streams and supporting business processes
# Exercise: Thinking about BPM

Work with 1 or 2 other folks at your table
Do a SWOT analysis for Agile dev (internal) & BPM (external)

<table>
<thead>
<tr>
<th>Strengths:</th>
<th>Weaknesses:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Advantages for Agile to support BPM</td>
<td>• Disadvantages of Agile for integrating with BPM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opportunities:</th>
<th>Threats:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Benefits to Agile practices from doing BPM</td>
<td>• Agile practices put at risk by doing BPM</td>
</tr>
</tbody>
</table>

Determine a ‘key’ takeaway & reasoning why
Agile & Configuration Management

Locking Things Up

- vs -

Frequent, Effortless, and Sustainable Continuous Updates
Configuration Management is a process for establishing and maintaining consistency of a product's performance and functional and physical attributes with its requirements, design, and operational information throughout its life.


Question: Doesn't Continuous Integration do all of my CM for me?
CM Practitioners

• Good at keeping track at lots of details and variables
• Familiar with the technologies being used
• Be focused on optimization of the CM function
  – Automation of CM processes
  – Integration of CM specific information in different tools / repos
• Able to determine/leverage capabilities of CM tools
  – Avoid duplication of information
  – Determine authoritative sources
• With current tooling, CM should think beyond the spreadsheet
• Scope of CM and number of projects determines volume of CM need
Patterns: Agile & CM

• Ongoing CM (throughout the development cycle)
  – Work to automate CM activities via tooling
  – "CM as a service" (part of a SaaS offering)
Pattern: Ongoing CM

Benefits:
- Minimize additional 'work' for software developers
- Improve software assurance
- Minimize wait/waste states at beginning or end of dev cycles

Challenges:
- Maintain proper queuing through the release pipeline
- Avoid introducing artificial bottlenecks
- Achieving governance
  - Value stream optimized through common patterns
  - Uniform understanding of tooling and patterns
Services for CM

- Optimized patterns for CM tool use
- Project, build & repo templates that meet dev & CM reqs
- Maintain shared utilities, libraries, and components
  - Potential for EA linkages
- Capture and inventory of created IT assets
  - Additional EA linkages
- Standardized conventions for version & release identification
  - Mechanism to extend standard convention if needed
- Help with awareness and notification for software updates
  - Notifications can be linked to automation strategy
Benefits of an Enterprise CM function

• Only pay to setup CM once
  – Each team/project shouldn't need to figure out CM for itself
  – Achieve additional cost savings through economies of scale
  – Greater adoption of Enterprise CM drives ROI
  – Infrastructure / tools optimized for common CM functions

• An optimized & effective tool set
  – Avoid costly, error-prone ad-hoc data integrations between tool sets
  – Avoid paying for duplicative tools

• Linkages to common backup and software assurance strategies
  – Ensure important / less-visible software dev activities aren't omitted
  – Drives greater ROI and decreases non-development tasks

• Increased sustainability
  – Common patterns and practices between dev efforts
  – Reduced changing costs
Exercise: Thinking about CM

The Question: What needs to go into CM?

Draw a picture this captures:

#1 - What goes into CM (beyond source code)
#2 - Appropriate relationships of what is in CM
#3 - Relationship between different projects or teams in CM

It’s a given that source code goes in CM - what else matters?

Perhaps stuff to include might be:

- Builds/Binaries/Executables (built from source, shared, third-party, etc)
- Business Process Components
- Unit Tests
- User Acceptance Tests
- Stories / Features / Requirements
- Enterprise Requirements
- Funding Lines/Governance Artifacts
- Env-specific application config files, Physical server config files, VM templates & infrastructure level CM
Conclusions

Agile development is complementary to:

- Six Sigma
- Business Process Management
- Configuration Management
Agile & Six Sigma: Top 3

• Use Six Sigma to generate quantitative acceptance criteria for software development activities

• Use Agile development to begin to realize Six Sigma ROI faster

• Use Six Sigma to define over-arching performance standards for software development to support Enterprise strategy
Agile & BPM: Top 3

- Use BPM to generate technical requirements for business process interfaces /contracts with services

- Use BPM simulations to generate Service Level of Agreements that IT systems (and supporting infrastructure) must achieve

- Use BPM simulations to augment acceptance test scenarios
Agile & CM: Top 3

- CM processes must be optimized (ideally automated) to not impact efficiencies gained through Agile

- Properly envisioned patterns and templates to help CM are welcomed by developers

- A shared & common CM function offers the potential of significant ROI
Questions & Comments

Thanks for attending!

Jason Tice - Asynchrony (St. Louis, MO)

Contact:
Email: jason@jasontice.com - jason.tice@asolutions.com
Twitter: @TiceThoughts

Content:
• Web: www.jasontice.com (under transformation)
• Blog (via Asynchrony): blog.asolutions.com
• Podcast: www.thisagilelife.com
Backup
Is Six Sigma Agile?

- Hypothesis: It could be based upon some attributes.
- Possible decision factors: Project Scope, Tollgate cycle time, etc
- Perhaps best to let ‘Six Sigma’ be ‘Six Sigma’

Industry trends:

- Six Sigma is known for rigors to ensure large-scale, high ROI process improvement.
- Big successes (big scope) gets big press.
- The 'bigness' of Six Sigma challenges the 'smallness' of Agile.

Takeaways:

- Agile software development can be a key component 'big' successes
- Small Agile (Kaizen) improvements drive 'significant' impacts
- Agile development can strongly complement Six Sigma projects
Another $64,000 question?

Should you model and simulate a team's Agile development process in a BPM tool?
• **Probably Not!** *(aside from the obligatory high level sketch)*

Supporting Thoughts:
• Dev processes need to be flexible to support change & innovation
• Scope of process/activity & number of actors typically small
• Actors seek to improve and overcome challenges
  – As soon as you model it, it may change (for the better)
• Some developers are bothered by it
• ‘Adaptive Case Management’ may be a better approach
  – Define success criteria agnostic of how success is achieved
• If you must model - consider inputs/outputs to Agile dev
  – Non-automated integration/test/CM/release processes
  – Multi-party requirements approval & governance processes
What about Kaizen?

- Japanese for "improvement" or "change for better"
- Not a part of Six Sigma, but highly related
- Kaizen techniques used to implement Six Sigma projects
- CPI COEs & SMEs often do both
  - Continuous Process Improvement Centers of Excellent and Subject Matter Experts often do both
- Kaizen specific techniques:
  - Toyota Production System
  - Quality Circles
  - Plan-Do-Act-Check (PDCA) Cycles
Pattern: Agile throughout the Six Sigma lifecycle

Implementation:

• Black belt was a member of the Agile development team
• One war-room, whole-team approach, team paired with BB, etc
• Single set of stories (including Six Sigma specific stories), single Kanban board

Benefits:

• Stronger "whole-team" approach - team maintained good understanding of business case
• Improved consistency of staff utilization - black belt helped with QA during later tollgates
• Tech staff pushed to not defer "difficult" problems until the end of the project (getting data/services, setting up builds/release scripts, etc)
• Metrics dashboards / alerts were available to support test activities
• Still applicable: Knowledge / skill transfer and "rock solid" quantitative acceptance criteria
Pattern: Agile throughout the Six Sigma lifecycle

Challenges:

- Utilization increased but not necessarily optimized
  - Dev team building environments vs. PaaS/SaaS
  - Black Belt supporting QA activities
- Sustainment of value-add work across all tollgates during repeated cycles
Bonus Pattern: Six Sigma on the boundaries of Agile development

Implementation:

• Six Sigma staff separate from Agile dev team
• Six Sigma staff operated via a shared bureau (Center of Excellence)
• Six Sigma project cycle and Agile iterations executed separately
• Six Sigma criteria integrated into details of Agile dev features/stories
• Agile Dev teams worked on projects which had stories ready to go
Bonus Pattern: Six Sigma on the boundaries of Agile development

Benefits:
- Optimized value streams for all staff
  - Perception of 'busy' work for dev teams mitigated
  - Black belts were able to support multiple dev teams
- Realized additional ROI through PaaS/SaaS offerings
- Strong quantitative acceptance criteria still present
- No 'out-of-balance' (risk prone) work activities
  - Risks learned writing too many acceptance tests up front

Challenges:
- Less understanding of Six Sigma and team vision amongst development staff
- Team leads had to ensure that focus on being 'proactive' was not lost
  - Configuration of metrics API & dashboards
  - Determine strategies to obtain data for testing and prod operations
A Six Sigma "pattern" driven Use Case

• Problem:
  – Network/datacenter maintenance required having more than a dozen application engineers on call to enable/disable applications

• Root Cause:
  – Lack of a common mechanism to enable/disable applications for system/network/data-center maintenance

• Six Sigma Defined:
  – A common pattern for a web service interface and admin application that could be retro-fitted to all applications

• Desired Control:
  – Datacenter staff could enable/disable/smoke-test all applications without the need for application engineering / support

• Outcome:
  – Recurring costs of maintenance windows were reduced - ROI achieved through reduction/elimination of on-call staff hours
  – The correlation of maintenance window labor cost to number of Enterprise applications was reduced
Intended benefits:

• Additional source of acceptance criteria
  – Specific (measurable) service levels needed to achieve success

• Proactive measurement system design is typically more effective and efficient
  – Initial investment can support future Six Sigma (CPI) efforts

• Decreased time to realize ROI from Six Sigma efforts
  – Iterative deliveries offer incremental performance improvements
Six Sigma is a heavy methodology to improve Agile dev activities

- Six Sigma improves non-commodity complex processes
- Agile dev should be a simple (commodity-like) process

Elements of Six Sigma can help Agile teams

- Define quantitative criteria make improvements from Retrospectives measurable
- Generate short ‘DMAIC’ cycles within Retros
  - Increase whole team pairing by X
  - Ensure test suite execution time does not exceed Y
- Create a climate where Agile dev teams think more about metrics across all activities and end-to-end processes
Is BPM software development?

No - it is a complementary and/or enabling practice
• Software dev provides the services and systems that support BPM processes

But
• BPM enabled processes can have bugs / errors
• BPM enabled processes should be tested
• BPM enabled processes have a lifecycle

Many Agile development techniques can be applied to BPM work
• Can and should utilize test driven design
• Continuous integration/testing of processes
• Strategies to accelerate process ROI (MRC for processes)
Bonus Pattern: Development done without BPM

Benefits:
- Simplified solution architecture
  - No need to manage a BPMS / Simplified IT integrations
- Simplified customer relationship
  - Dev team works for the customer
- Implement systems/service specific to processes
  - Use software dev best practices
  - Avoid 'quirky' items needed for BPM integration

Challenges:
- No out of the box BPM capabilities
  - Automated analysis and/or optimization is lost
  - Custom code and/or other integration for optimization, etc
- Development staff needed for system enhancements
- Loss of BPMS provided capabilities to determine SLAs & testing
Motto: "If you build it, they will come."

- Provide a common CM repo and build system to dev teams
  - CM repo includes:
    - Source code management
    - Library / Binary / Dependency management
    - SDLC capabilities linked to technical artifacts (source, libraries, etc)

- Establish common project and build templates
- Use common builds to define common outputs for each dev pattern
- Automate deployment and testing of standard packages
- Establish governance to use standard builds
- Provide a flexible mechanism for needed customizations
CM Challenges Experienced

- **Automation challenges**
  - KSA gaps (tool use / automation implementation)
  - Working around IA requirements
  - Lack of tools

- **Lack of standards & governance**
  - Working software vs. standardizing the process

- **Resource challenges**
  - Improving a live process as during execution

- **Myth: “You can't automate if you can't route.”**
  - Solution: Automate around the air-gap
Messaging for CM

Perceptions & challenges with Enterprise level CM

- **Control Boards are inefficient**
  - Kanban can help with this

- **Determining an effective scope of governance**
  - Manning challenges indicate other problems
  - Myth: CM cannot scale to an Enterprise level
  - Myth: Enterprise CM is too big to be automated

- **Managing expectations across many stakeholders**
  - Use quantitative measures (ROI, etc) to decide change requests
  - Use “scientific” criteria to drive evaluations (not emotions)