Fall 2009

Strategic Tool Building in Data Engineering: Using the Systems Engineering Process to Meet Strategic Business Goals

Amanda Gerdes

Loyola Marymount University

Follow this and additional works at: http://digitalcommons.lmu.edu/se_etdrps

Part of the Systems Engineering Commons

Recommended Citation


http://digitalcommons.lmu.edu/se_etdrps/2

This Oral Presentation - Campus Accessible Only (with IP restrictions) is brought to you for free and open access by the Systems Engineering at Digital Commons @ Loyola Marymount University and Loyola Law School. It has been accepted for inclusion in Systems Engineering Research Projects and Oral Presentations by an authorized administrator of Digital Commons@Loyola Marymount University and Loyola Law School. For more information, please contact digitalcommons@lmu.edu.
Strategic Tool Building
In Data Engineering
Using The Systems Engineering Process
To Meet Strategic Business Goals

Amanda Gerdes
Systems Engineering Integrative Project
Loyola Marymount University
14 December 2009
The Company, The Business, and Data Engineering

Introduction
Why does it cost so much for us to do what we do?

Can we do it for less?

What precisely is it that we do?

Is that what we should be doing?

Or do we change the strategic vision of this company to encompass quality and to stop tolerating operational inefficiencies?

Should we lower quality?

Should we build fewer units?
About The Company

• History
  – Founded in 1999
  – Strong ties to academia, strong focus on research
  – Primarily government contracts (DARPA, NSF, Air Force)
  – Expansion into commercial sector in 2002

• Structure
  – 40 employees
  – Three primary functional groups
    • Labs (Research)
    • Software Engineering (Platform Development)
    • Data Engineering (Application Development)
About the Business

- Data Extraction, Aggregation, Normalization
- Business Models
  - Software Licensing Model
  - Design Consultancy Model
  - Data Delivery (Hosted Solution) Model
- Implementation Models
  - Scheduled Batch Scrapes
  - Runtime Scrapes
- Sample Implementations
  - Background Search/Risk Management: Runtime Data Delivery
  - Competitive Analysis: Scheduled Batch Data Delivery
  - Events Aggregation: Scheduled Batch Design Consultancy
About Data Engineering

• Role Within The Company
  – Accept conceptual guidance from Labs
  – Use platform software from Software Engineering
  – Develop customer applications

• Roles and Responsibilities
  – Requirements collection and analysis
  – Software and data architecture
  – Application design and build-out
  – Training, Technical Support

• Heterogeneous Makeup
  – Software engineers
  – Software analysts
  – Data analysis
  – Business analysis
  – Offshore resources
Data Pipelines, Development Processes, Total Cost of Ownership

The Current State
Data Pipelines

- Analogy: Rail transport
- Purpose
  - Provide shared infrastructure
  - Limit variable ("per-agent" costs)
- Manifestation
  - Shared, Generalized Codebase
  - Standards and Specifications
  - Individual, Per-Site Units
- Advantages and Disadvantages
  - Lower per-agent costs
  - Easy addition of new sources over time
  - Higher front-end development costs, time
# Pipeline Development: Process

## Pipeline Development Process

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1</strong> Create SOW</td>
<td>- Customer creates a statement of work (SOW) for the pipeline development process.</td>
</tr>
<tr>
<td><strong>3.1</strong> Design shared infrastructure</td>
<td>- Data Architect designs the shared infrastructure.</td>
</tr>
<tr>
<td><strong>5.1</strong> Build components</td>
<td>- Developers build components.</td>
</tr>
<tr>
<td><strong>6.2</strong> Verify and validate</td>
<td>- Design and Document: Design and verify the components. Documentation is created.</td>
</tr>
<tr>
<td><strong>7.1</strong> Configure hosted environment</td>
<td>- Operations configure the hosted environment.</td>
</tr>
<tr>
<td><strong>7.2</strong> Configure delivery mechanisms</td>
<td>- Operations configure delivery mechanisms.</td>
</tr>
<tr>
<td><strong>7.3</strong> Create deployment operations specifications</td>
<td>- Operations create deployment operations specifications.</td>
</tr>
</tbody>
</table>

### Diagram

- **Customer** (1.1 Create SOW)
- **Data Architect** (3.1 Design shared infrastructure, 5.1 Build components)
- **Developers** (6.2 Verify and validate, 7.1 Configure hosted environment, 7.2 Configure delivery mechanisms)
- **Operations** (7.3 Create deployment operations specifications)

- **Checkpoints**
  - Not Approved
  - Approved
Agent Development: Process

- Build
  - Navigation Definition
  - Extraction Definition
  - Custom Output Mods
- Unit Testing and Refinements.
- QA / Certification.
Development Effort By Task

- Markup, Learning & Testing Rules: 43%
- Output Data V&V: 20%
- Hands-On Runtime: 14%
- Custom Output: 10%
- Model HTTP Requests: 7%
- Feed Anatomy V&V: 6%
- Other: 0%
Pipeline Costs

- Per-Pipeline Costs
  - Typically fixed
  - Amortized over the lifetime of the deployment
  - Cost varies by complexity of the pipeline or variability within the entire data set

- Per-Agents Costs
  - Variable costs
  - Includes development, maintenance, and ongoing operations
  - Cost varies by both the complexity of the site population as well as the total number of agents to be produced
Strategic Alignment, Strategic Trade-Offs, and the Vision

The Future State
# Strategic Trade-Offs

<table>
<thead>
<tr>
<th>Area</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowered Quality</td>
<td>• Immediate, short-term impact</td>
<td>• Compromised reputation in marketplace</td>
</tr>
<tr>
<td></td>
<td>• No implementation costs</td>
<td>• No long-term cost benefits</td>
</tr>
<tr>
<td></td>
<td>• Lowered acquisition costs</td>
<td>• Maximum cost benefits are firmly bounded</td>
</tr>
<tr>
<td>Decreased Volume of New Sales</td>
<td>• Immediate, short-term impact</td>
<td>• Limited opportunities for growth or expansion</td>
</tr>
<tr>
<td></td>
<td>• No implementation costs</td>
<td>• Introduces “all eggs in one basket” risks</td>
</tr>
<tr>
<td></td>
<td>• Lowered acquisition costs</td>
<td></td>
</tr>
<tr>
<td>Postponing Competing Strategic</td>
<td>• Lower future costs</td>
<td>• High implementation costs</td>
</tr>
<tr>
<td>Initiatives</td>
<td>• Greater operational efficiency</td>
<td>• High opportunity costs</td>
</tr>
<tr>
<td></td>
<td>• Maintains quality levels and market reputation</td>
<td>• Effects are seen over the medium- to long-term</td>
</tr>
<tr>
<td></td>
<td>• Lays foundation for scalable growth</td>
<td></td>
</tr>
</tbody>
</table>
The Vision

• Difficult Decision:
  – Redefine the strategic vision for the company
  • Temporarily putting growth areas and product line expansions on the back-burner
  • Making the lowering of costs through increased operational efficiencies a company-wide strategic goal

• Manifestation in Data Engineering
  – Identify operational inefficiencies in Data Engineering
  – Develop and implement plan for becoming efficient, self-drivin
Five Whys, Pareto Analysis, Process Analysis

Identifying The Problem
Root Cause Analysis: The Five Whys

Why Is It So Expensive To Build Agents?

... because it takes a long time to build an agent.

WHY?

... because the process is very complex

WHY?

... because QA/Certification takes a long time

WHY?

... because transitions between development tools takes a long time

WHY?

... because the build tool requires manual modeling

WHY?

... because devs sometimes need to use inefficient workarounds

WHY?

... there are many steps and many loops

WHY?

... there are many things that have to be done

WHY?

... there are no automatic mechanisms for checking and reporting unit quality

WHY?

... because there are bugs in the development suite

WHY?

... because the dev suite does not have all needed functionality AND

WHY?

... because bug fixes and enhancement requests have been accumulating for 3 years

WHY?

... because the tool is old and has not been upgraded

WHY?

... because new technology from Labs has not yet been folded in

WHY?

... because the platform does not have an integrated GUI.

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not fully closed the loop between application and platform engineering

WHY?

... because we have never identified this as a need and have not assigned resources to do it.

WHY?

... because we have never identified this as a need and have not assigned resources to do it.

WHY?

... because the platform does not have an integrated GUI.

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?

... because we have not allocated the resources to do it

WHY?
Root Cause Analysis: Pareto Analysis

Estimated Cost Reductions Per Improvement Area

Improvement Areas By Cost-Effectiveness

Process Improvements QA Tools Build Tool Upgrade Integration of Platform Tools Bug Fixes
## Process Analysis: Identifying Waste

<table>
<thead>
<tr>
<th>Waste Area</th>
<th>Prevalence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Over-Production</td>
<td>High</td>
<td>Example: Documentation</td>
</tr>
<tr>
<td>Inventory</td>
<td>Medium</td>
<td>“Soft” inventory – that is, electronic artifacts – is an issue. (Example: Documentation, archives.)</td>
</tr>
<tr>
<td>Transportation</td>
<td>High</td>
<td>Transportation is a significant issue in the agent development process. (Example: Toolset interfaces.)</td>
</tr>
<tr>
<td>Unnecessary Movement</td>
<td>High</td>
<td>Physical movement is not an issue. However, there is a huge amount of unnecessary movement in terms of system configuration and disparate systems access. (Example: Disparate storage, access environments.)</td>
</tr>
<tr>
<td>Waiting</td>
<td>High</td>
<td>While sibling processes have been successfully parallelized, individual processes themselves remain highly linear and highly sequential.</td>
</tr>
<tr>
<td>Defective Outputs</td>
<td>High</td>
<td>The rate of defective outputs is fairly low but the relative severity of each is increased by the fact that they often are not detected until too far downstream.</td>
</tr>
<tr>
<td>Over-Processing</td>
<td>Medium</td>
<td>Significant improvement in past 6-12 months as pipeline development processes have matured and decreased the amount of per-unit processing that must be done. However, there remains a risk that in order to generalize a pipeline, too much variability is accounted for, creating a pipeline-level over-processing issue.</td>
</tr>
<tr>
<td>Insufficient Design (Womack)</td>
<td>High</td>
<td>External customer needs are typically well captured and well accounted for. However, the Data Engineering group performs their work using an outdated set of tools that is currently insufficient to meet their needs in terms of application development.</td>
</tr>
<tr>
<td>Unnecessary Space (Womack)</td>
<td>Low</td>
<td>Space is not typically an issue.</td>
</tr>
<tr>
<td>Negative Emotions (Oppenheim)</td>
<td>Medium</td>
<td>THE COMPANY’s workforce tends to be highly motivated, highly compatible, and united under a very strong corporate culture. However, financial complications in the company has manifested as lay-offs and salary cuts. This has created an anxious environment, commonly manifesting in negative emotions such as a lack of motivation and even discontent. These are counterproductive to a lean operation.</td>
</tr>
</tbody>
</table>
Strategic View, Tactical View, Operational (Technical) View

Building The Solution
Strategic View: Developing Operational Efficiency

STRATEGIC GOAL
Lowering Operational Inefficiencies in Data Engineering

INITIATIVE
Quality Tools

PROJECT 1
Feed Anatomy

PROJECT 2
Attribute-Level Data Integrity

PROJECT 3
Schema-Level Integrity

PROJECT 4
Data-Level Spot-Checking

PROJECT 5
Entity-Level Spot Checking

INITIATIVE
Process Improvements

INITIATIVE
Build Tool Upgrade

INITIATIVE
Bug Fixes

INITIATIVE
Platform Integration
Tactical View: The Quality Tools Initiative

<table>
<thead>
<tr>
<th>Area</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal Quality and CI Program</td>
<td>• Marketing point</td>
<td>• “Buzz word”</td>
</tr>
<tr>
<td></td>
<td>• Could be used to generate credibility in</td>
<td>• May have/create infrastructure cumbersome</td>
</tr>
<tr>
<td></td>
<td>and for industry</td>
<td>to very small company</td>
</tr>
<tr>
<td>Develop In-House Program</td>
<td>• Compatible with strong corporate culture</td>
<td>• May be resource intensive</td>
</tr>
<tr>
<td></td>
<td>• Can be implemented at a grass-roots level</td>
<td>• Requires in-house expertise</td>
</tr>
<tr>
<td></td>
<td>• May be used to generate leadership position in</td>
<td></td>
</tr>
<tr>
<td></td>
<td>industry</td>
<td></td>
</tr>
</tbody>
</table>

The Plan

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
</tr>
</thead>
</table>
| Feed Anatomy             | • Objective: Create a tool which can validate the structure of all feed objects against a project-specific, configurable “gold standard”.  
                            • Expectations: Greater overall quality and consistency of feed objects within pipelines; faster verification times; lower rework  
                            • Mechanism: Detailed below                                                                                                                                                                         |
| Attribute-Level Integrity| • Objective: Create a tool which can validate data at the attribute level, e.g. “Do the values reported in column item_price look like prices?”  
                            • Expectations: Faster verification times, possible runtime validation, development of standardized, reusable internal schemas  
                            • Mechanism: Reusable type queries, reusable execution framework, reusable reporting framework                                                                                                      |
| Schema-Level Integrity   | • Objective: Create a tool which can validate XML output via validation against a well-defined schema  
                            • Expectations: Faster verification times, more accurate verification, encapsulation of project requirements in standard format                                                                                                                                 |
| Entity-Level Spot Checking| • Objective: Create a tool that provides an integrated view of extracted tuple and details page  
                            • Expectations: Vast reduction in recall/precision data collection and reporting  
                            • Mechanism: Given tuple, load and display corresponding page; given page, load corresponding tuple; specify “correct/not correct” for behind-the-scenes statistics gathering                                                                                                                                 |
| Web-Based Spot-Checking   | • Objective: Create a tool that is capable of facilitating recall metrics collection  
                            • Expectations: Vast reduction in recall data collection and reporting  
                            • Mechanism: Randomly crawl site, gather random identifiers from target population, verify that those entities appear in the output                                                                                                                                 |
Goals, Requirements, Trade Studies, Agile Software Development

The Feed Anatomy Project
Goals

• Goals
  – Build a tool or toolset capable of validating the structure of all feed components with minimal human interaction
  – Capture the project-specific information required to perform this check in a reusable fashion
  – Give all layers of the development process the same tools to use to promote visibility, consistency, and developer empowerment
  – Provide a means for capturing and reporting of errors so that statistical process control methodologies can be applied

• Schedule
  – Original: Completion by December 31, 2009
  – Modified: Completion by January 31, 2010
Requirements: Customer Requirements

- Customer Identification: Internal uses (Data Engineering, on- and off-shore)
- Requirements inputs: User interview

<table>
<thead>
<tr>
<th>Req. ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.001</td>
<td>The tool will be run on individual feeds during the development process.</td>
</tr>
<tr>
<td>C.002</td>
<td>The tool will be run on entire production repositories.</td>
</tr>
<tr>
<td>C.003</td>
<td>The data itself (rather than the validation logic applied to it) must be reusable for other applications.</td>
</tr>
<tr>
<td>C.004</td>
<td>Verification must include both departmental- and project-level standards.</td>
</tr>
<tr>
<td>C.005</td>
<td>The tool must accommodate both XML- and database-output feeds.</td>
</tr>
<tr>
<td>C.006</td>
<td>Any errors detected must be captured and reported both for operational purposes as well as ongoing tactical process reviews and control.</td>
</tr>
<tr>
<td>C.007</td>
<td>The system must be able to run in the standard Data Engineering development environments, both on- and offshore.</td>
</tr>
<tr>
<td>C.008</td>
<td>The system must be able to run in the current production release of the platform.</td>
</tr>
<tr>
<td>C.009</td>
<td>The system must be able to run with minimal human interaction.</td>
</tr>
<tr>
<td>C.010</td>
<td>The system must be eventually integrated into the development suite.</td>
</tr>
</tbody>
</table>
## Requirements: Functional Requirements

- Requirements inputs: Customer requirements, best practices documentation, project-specific implementations, user interview

<table>
<thead>
<tr>
<th>Req. ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.001</td>
<td>The system must produce a single summary file from multiple source materials.</td>
</tr>
<tr>
<td>F.001-A</td>
<td>The summary file must contain all data required for validation checks.</td>
</tr>
<tr>
<td>F.001-B</td>
<td>The summary XML document must conform to a standard format.</td>
</tr>
<tr>
<td>F.001-C</td>
<td>The system must be capable of reading from hierarchally-organized XML files in the file system.</td>
</tr>
<tr>
<td>F.002</td>
<td>The validation system must be configurable.</td>
</tr>
<tr>
<td>F.002-A</td>
<td>The configuration file must contain all data required for system configuration.</td>
</tr>
<tr>
<td>F.002-B</td>
<td>The configuration file must conform to a standard format.</td>
</tr>
<tr>
<td>F.002-C</td>
<td>The configuration file must be human-readable.</td>
</tr>
<tr>
<td>F.003</td>
<td>The system must be able to detect and report invalid Summary Values.</td>
</tr>
<tr>
<td>F.003-A</td>
<td>The system must be able to detect and report if a Summary Value does not exactly match a string in an enumerated list.</td>
</tr>
<tr>
<td>F.003-A-1</td>
<td>The enumerated list of values must be configurable.</td>
</tr>
<tr>
<td>F.003-A-2</td>
<td>The system must be able to select the appropriate enumerated list based on other criteria.</td>
</tr>
<tr>
<td>F.003-B</td>
<td>The system must be able to detect and report if a Summary Value does not match a regex pattern</td>
</tr>
<tr>
<td>F.003-B-1</td>
<td>The pattern must be configurable.</td>
</tr>
<tr>
<td>F.003-B-2</td>
<td>The system must be able to select the appropriate regex pattern based on other criteria.</td>
</tr>
<tr>
<td>F.003-C</td>
<td>The system must be able to detect and report if there is an incorrect # of occurrences of a Summary Value.</td>
</tr>
<tr>
<td>F.003-D</td>
<td>Negative Requirement: There is no requirement that the system use separate mechanisms for each of the Invalid Summary Value checks.</td>
</tr>
<tr>
<td>F.004</td>
<td>The system must be able to generate complex rules and apply them to Summary Values appropriately.</td>
</tr>
<tr>
<td>F.004-A</td>
<td>The system must accept configurable sets of rules in which the comparisons, operators, and values are customizable.</td>
</tr>
<tr>
<td>F.004-B</td>
<td>The system must be able to perform sequential comparisons on same or disparate Summary Values in order to apply rules.</td>
</tr>
<tr>
<td>F.005</td>
<td>The system must be able to generate reports with results.</td>
</tr>
<tr>
<td>F.005-A</td>
<td>The reports must include a &quot;pass&quot;/&quot;fail&quot; indicator for a given complex rule.</td>
</tr>
<tr>
<td>F.005-B</td>
<td>The reports must include the failed characteristic and failed value of any &quot;fail&quot;-status rule.</td>
</tr>
</tbody>
</table>
### Requirements: Performance Requirements, Design Constraints

#### Performance Requirements

<table>
<thead>
<tr>
<th>Req. ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.001</td>
<td>The time to verify structural quality using the tool must be less than the manual equivalent.</td>
</tr>
<tr>
<td>P.002</td>
<td>The system must be able to operate across full production deployments consisting of up to 250,000 feeds.</td>
</tr>
<tr>
<td>P.003</td>
<td>The system must be able to perform at least 150 separate tests on each feed.</td>
</tr>
</tbody>
</table>

#### Design Constraints

<table>
<thead>
<tr>
<th>Req. ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.001</td>
<td>The system must adhere to all applicable coding standards (company, department).</td>
</tr>
<tr>
<td>D.002</td>
<td>The system must be architected in such a way as to be compatible with or complimentary to existing Data Engineering products.</td>
</tr>
<tr>
<td>D.003</td>
<td>The system must be written in Java.</td>
</tr>
<tr>
<td>D.004</td>
<td>The system must employ existing Java libraries when feasible.</td>
</tr>
</tbody>
</table>
Technical Trade-Offs

• No general selection priority applied
  – Each functional need was evaluated in terms of the implementation trade-offs specific to that need.

<table>
<thead>
<tr>
<th>Area</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off-The-Shelf Product</td>
<td>• No development cost</td>
<td>• Inflexible</td>
</tr>
<tr>
<td></td>
<td>• Low relative resource requirement</td>
<td>• Creates external dependencies</td>
</tr>
<tr>
<td></td>
<td>• Product support</td>
<td>• Workarounds, tweaking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Licensing, support costs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Unique codebase</td>
</tr>
<tr>
<td>New In-House Development</td>
<td>• Internal dependencies only</td>
<td>• High relative development cost</td>
</tr>
<tr>
<td></td>
<td>• Flexible, agile, responsive</td>
<td>• High relative resource requirement</td>
</tr>
<tr>
<td></td>
<td>• Correct solution</td>
<td>• Internal support</td>
</tr>
<tr>
<td></td>
<td>• Easy platform integration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Library compatibility</td>
<td></td>
</tr>
<tr>
<td>Existing In-House Development (Data</td>
<td>• Low development cost</td>
<td>• Strong dependencies with other applications</td>
</tr>
<tr>
<td>Engineering)</td>
<td>• Quick development</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Smooth integration, support</td>
<td></td>
</tr>
<tr>
<td>Existing In-House Development (Software</td>
<td>• Low development cost</td>
<td>• Long lead time</td>
</tr>
<tr>
<td>Engineering)</td>
<td>• Strong platform integration</td>
<td></td>
</tr>
</tbody>
</table>
Agile Software Development

- Adaptive (rather than predictive)
- Emphasis on
  - Quick developer cycles
  - A constant working software suite, frequent deliveries
  - Responsiveness to change
  - Self-organizing teams

<table>
<thead>
<tr>
<th>Area</th>
<th>Adaptive (“Agile”)</th>
<th>Predictive (“Planed”)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criticality</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Requirements</td>
<td>Volatile</td>
<td>Relatively stable</td>
</tr>
<tr>
<td>Developer Experience</td>
<td>Senior</td>
<td>Junior</td>
</tr>
<tr>
<td># Developers</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Culture</td>
<td>Thrives on chaos</td>
<td>Demands order</td>
</tr>
</tbody>
</table>

- "Cowboy Coding"
Implementation

• Individuals and Interactions
  – Weekly Sync-Up
  – Daily “Stand Up” (10 minutes): What I did yesterday; What I am doing today; What, if anything, is blocking me

• Working Software
  – Weekly deliveries of fully-functional standalone incorporating all latest additions

• Customer Collaboration
  – Immediate deployment to production test beds
  – Fold-in of modifications, new requirements

• Responsiveness to Change
  – Completion of feedback loop
  – Real-time flexibility with regard to resource allocations and competing initiatives
Development Cycles

Sunday
Weekly Sync-Up for Build \(i\)
(Project Team)

Sun - Thu
Build \(i\)
(Software Engineers)

Thu - Fri
V&V of Build \(i\)
(Data Architect)

Sun

Mon Tue Wed Thu Fri Sat

Sun

Mon Tue Wed Thu Fri Sat

Sun - Thu
Build \(i+1\)
(Software Engineers)

Thu - Fri
V&V of Build \(i+1\)
(Data Architect)

Fri - Fri
Rollout of Build \(i\)
(Operations)

Fri - Sun
Close Out Build \(i\)
Launch Build \(i+1\)
(Data Architect, Operations)
Data Collection and Evaluation

The Results
Data Collection

- Data Collection Methodology
  - Target production pipeline consisting of 45 feeds
  - Structural validation process includes 25 validation checks
  - Side-by-side manual and automated validations performed

- Limitations of Data
  - Short time period
  - Limited deployment
  - Incorporation of strategically-mandated operational workarounds
  - Inappropriately granular level of detail requested
Findings: Effort

- Reduced to 3% of original person-effort
  - May be more: “Check” time is hands-on for manual process; the “Check” time for the tool is non hands-on.

<table>
<thead>
<tr>
<th>Area</th>
<th>MANUAL Time (s)</th>
<th>WITH TOOL Time (s)</th>
<th>SAVINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set-Up</td>
<td>30</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>Checks</td>
<td>13500 (300 per feed)</td>
<td>245</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>13530 seconds</td>
<td>365 seconds</td>
<td>13165 seconds</td>
</tr>
<tr>
<td></td>
<td>3.76 person-hours</td>
<td>0.1 person-hours</td>
<td>3.67 person-hours</td>
</tr>
</tbody>
</table>

Assumptions:
- Pay Rate: $25/hour (on-shore resourcing)

| COST (Labor) | $93.96 | $9.53 | $91.42 |
Findings: Precision and Recall

- Increased Accuracy Over Manual Process
  - Six structural flaws *undetected* by manual process
  - Successfully captured by tool
- Limitation: Alerting on non-alert conditions
- Precision and Recall
  - Recall: Number of relevant entities retrieved per number of relevant entities overall
    - Must be 1.0 for the tool to be *successful*
    - All error conditions detected
  - Precision: Number of relevant entities retrieved per total number of entities retrieved
    - Must approach 1.0 for the tool to be *efficient*
    - As few false positives as possible
Findings: Efficiency and Scalability

- **Scalability**
  - Sample pipeline consisted of 45 feeds only
  - Average pipelines contain hundreds
  - Performance of automated solutions scales!

- **Efficiency**
  - Multiple points in the agent development process require verification of structural quality
  - Savings scale throughout process
Statistical Process Control

- Goal: Implement a Statistical Process Control for the agent development process as part of January 31, 2010 launch
- Feed Anatomy Tool provides infrastructure for data collection and reporting to support SPC

![p-Chart based on initial data]
Conclusions
Current Status

- Feed Anatomy Project: Preliminarily successful!
- Already in production
- Already saving person-hours
- Closeout on target for January 31, 2009
Next Steps

• Near Term: Feed Anatomy
  – Process integration: Deployment to full-scale production use across deployments
  – Process control: Implementation of monitoring/reporting feature to allow for statistical process control
  – Continuous improvement:
    – Platform integration: Integration into the platform architecture
• Medium Term: Attribute-Level Data Integrity, Schema-Level Data Integrity
• Longer Term
  – Entity-Level Spot-Checking
  – Integrated Web-Based Spot-Checking
  – Standardized quality methodology and certification

... but this is only the beginning!
Conclusions

• Powerful Combination: Systems Engineering Concepts and Strategic Alignment
  – Company-wide alignment
  – Powerful tools that produce results that are meaningful and visible across all levels
Thank You.