Our Mission:
Our mission is to provide our Business Partners with highly Available, Reliable, Scalable, Innovative, and Secure systems & services, in the most effective & cost efficient manner possible.

CVS Caremark Approach for Developing Large-Scale Distributed Systems Architecture

Requirements-Driven Composite Architectures

Farhad Etessami
Senior Architect
farhad.etessami@cvscaremark.com

CVS Caremark Corporation
May 10, 2012
• CVS Caremark Corporation (CVS Caremark) is a pharmacy healthcare provider in the United States

• Major business units:
  – Pharmacy Benefits Management (PBM) mail order and specialty pharmacy division
  – CVS/pharmacy, Longs Drug and Minute Clinic retail-based health clinics
  – Online pharmacy, CVS.com.

• $100+ billion revenue
• 1.3 billion prescriptions annually
• 2400 IT associates
• 2 major Data Centers in Woonsocket, RI and 1 in Scottsdale, AZ
• 19 Distribution Centers
• Over 200,000 associates
• Over 7,300 retail pharmacy stores
• 44 retail specialty pharmacy stores
• 560 Minute Clinics
• 249 Major current projects
• 525 major systems in production
Executive Summary

• Situation
  – Increasing complexity of enterprise class, distributed, large-scale systems
  – Increasing complexity of IT footprint due to number of software, hardware and networking platforms

• Challenges
  – Cost, time-to-market and quality of systems

• Solution
  – Consistent approach for development and documentation of system architecture

• Benefits
  – Significant cost and time reductions
  – Significant quality improvement
  – Significant impact in mean time to failure (MTF)
  – Significant impact in mean time to recovery (MTR)
**Architecture Definitions**

1. Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IEEE Std 1472000].

2. Architecture is the organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems. [UML 1.5]

A system architecture is a minimal collection of components that are deemed significant and with lasting effect:
- Organized to accomplish a specific function or set of functions
- Defines components relationship with each other and their environment
- Documented rationale for why architecture is the way it is
Lessons Learned: Formalize existing architectural practices at CVS Caremark

1. System architectures are not created in isolations, they need to interact with and use their neighborhood systems within Enterprise IT

2. A pragmatic and simple approach to enable consistent architectures and their descriptions across CVS Caremark

3. Limit complexity of architectural development to enhance adoption: Enhance organizational adoption by building peer level advocates

4. Since buy and not build is organizational standard, composition is used for creating system architecture

5. Clarify and emphasis demarcation between architecture and design: Focusing on architecturally significant requirements to enhance speed of system delivery and architectural quality

6. Complete adherence to the formalism outlined to enhance consistency and readability: Minimize formalism and complexity as much as possible
System Architecture (SA) at CVS Caremark

- System architecture (SA) describes the minimal set of components and their relationship with each other and their neighborhood within CVS Caremark enterprise IT

  - SA is a framework and foundation within which subsystems/applications are architected, designed and implemented

  - SA defines set of technology and architecture, standards and best practices governing the overall system being developed and deployed into production

  - Each SA describes how that system is configured and relates to other systems within CVS Caremark enterprise IT

  - Systems are constructed using set of components connected at their seams through series of well defined Architecture Integration Complexity Point (AICP). AICP defines the required conditions (consistent state) that must exist between the two adjacent components and based on Architecturally Significant Requirements (ASR) for the purpose of interoperability.
SA of highly complex system: High Level Work Flow

- System architectures are not created in isolations, they need to interact with and use their neighborhood systems within Enterprise IT.

**Rules of Thumb for Component Decomposition:**
Lots of behavioral interactions with other components (chattiness factor) indicates the possible need for further decomposition (usually no more than 2 or 3 levels).

Decompose into its subsystems / components.
Seams and AICP

- A SEAM is defined by one or more AICP and pair of components
- Each AICP is defined by a set of parameters called Integration Attributes (IA)
- Instantiated AICP is defined by associating values (V) to IA
  - Instantiated AICP defines the required conditions that must exist between its components for the purpose of interoperability and to existence of a consistent state by its components.
- AICPs within a Seam are translated in to set of operational interface (OI) for related components

Integration Attributes (IA):
1. Business/functional information exchange (behavior)
2. Type/Format of information exchange
3. Frequency of exchange
4. Size of unit of Data exchange
5. Is it transactional (ACID)
6. Online/ Batch
7. Interoperability Protocol
8. Performance
9. Throughput
10. Availability
11. Reliability
12. Security

Values for IA based on ASR:
PeopleSafe system needs to retrieve and save data with EAS online.
Type/Format: TXT file
Structure of Data:
Frequency of exchange: hourly
Size of unit of exchange: < 1kb
Is it transactional: Yes
Online/ Batch: online
Protocol: HTTP
Dependency: PeopleSafe dependent on EAS
Direction: bidirectional
Performance: < second
HA: 24x7
Distribute ASRs to Seams to instantiate AICPs for each seam

- Distribute ASRs to seams as appropriate defining requirements for interoperability between two adjacent components
**Architecturally Significant Requirements (ASR)**

- Initial total number of users
- Initial total number of concurrent users
- Rate of growth over the next 3 years
- Availability needs
- DR Tier
- Security level and needs
- Response time
- Outage notification to users (lead time)
- Maintenance windows
- Backup needs - frequency
- Size and frequency of data transfers
- What is the frequency of change of system

**Performance, Throughput and Scalability**
The application must be able to support approximately 235 concurrent users with possible three-year growth to 470 concurrent users.

**Reliability and Availability**
The system must be available Monday through Saturday during the hours of 6:00 AM to 10:00 PM.
Downtime for nightly system processing and maintenance is expected to occur in the overnight (10:01 PM to 5:59 AM). Need to establish if Business will still have access to application.
In case of server interruptions, degraded performance accepted for up to eight hours until server is available.
Database / server maintenance window will be from Sunday 2:00 AM – Sun 4:00 PM.
System Architecture Artifacts and relationship

- System Integration Matrix
- Technology Integration Matrix
- Architectureurally Significant Requirements (ASR)
- System Architecture (SA)
- High Level Bill of Materials
- Deployment Architecture

Relationships:
- System Integration Matrix is input to System Architecture (SA).
- ASR is input to System Architecture (SA).
- Technology Integration Matrix is input to System Architecture (SA).
- System Architecture (SA) is part of High Level Bill of Materials.
- System Architecture (SA) is part of Deployment Architecture.
<table>
<thead>
<tr>
<th>PIM Integration Points</th>
<th>Integration Requirements</th>
<th>Integration Approaches</th>
<th>Implementation Team</th>
<th>Pros/Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- PIM Real Time (online) integration with M/F</td>
<td>1- Type/Format(BLOB, TXT, XML, etc.): TXT, XML</td>
<td>1- JCA (Java Connector Architecture) - Java Adapter Mainframe (JAM) - CRM on the mainframe - TCP/IP Connection to Mainframe - SNA(VTAM) from CRM to Comm Area - Dynamic Programming Link (cobol/cics programming on the mainframe)</td>
<td>1- Requires mainframe (Perot resources) 2- Requires transactional programming by Soft Solution 3- Requires using JAM adapter 4- Requires DPL style cobol/cics programming on the mainframe</td>
<td>This requires soft solution to develop hooks to leverage this capability of the platform they are using (WebLogic). It does however provide tighter integration between M/F and Soft Solution and support full global transaction. Requires using WebLogic experience resources. Requires purchase of the Adapter from iWay (IB) and some consulting from them</td>
</tr>
<tr>
<td></td>
<td>2- Structure of Data:</td>
<td>2- GIS Sterling as as EAI technology</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3- Frequency of exchange: 3000 per day (.05TPS)</td>
<td>3- Using CICS and MQ with MQ adapter for CICS - Recommended Approach Skill- MQ and CICS/COBOL</td>
<td>1- Setup MQ queues on M/F and EMH (Nicolay, Brown) 2- Develop CICS/Cobol program (M/F CICS/Cobol Developer) 3- Develop JMS client on the Application server to enable interaction through MQ to VSAM files on M/F for Item updates (Soft Solution Developer) 4- Application Architect- Ber 5- Enterprise Architect - Farhad</td>
<td>1- This requires experience in those technologies which we have in house and the technologies are already in place. This is a low level approach to the integration problem. 2- Does not support global transaction</td>
</tr>
<tr>
<td></td>
<td>4- Size of unit of exchange: One record</td>
<td>4- Use WDZ which provides Web Services solution for this integration- Skill- WDZ, Web Services, JAVA and CICS/COBOL</td>
<td>1- Install WDZ and New version of Cobol compiler on M/F (Perot) 2- Install WDZ client on desktops (desktop support/engineering) 3- Develop Web Service implementation on the M/F for read and update from VSAM files and publish WSDLs for the services(Kathy Varela and IBM consultant) 4- Develop Web service client to consume WSDLs provided by the service for read and update of VSAM files on the M/F( Soft Solutions Developers) 5- Application Architect - Ber 6- Enterprise Architect - Farhad</td>
<td>Basically provides a high level abstraction on approach 3 above by providing powerful IDE and client side workbench to support rapid development. We have limited exposure to this technology (past six months) through some training provided by IBM. They are also willing to help us in using this technology.</td>
</tr>
</tbody>
</table>
Evolution of AICP through Development Life Cycle

System Architect: AICP

- Develop and Use ASRs
- System Architect identify Key components and neighborhood components in IT
- Define Seams/AICP
- Instantiate AICP

Application Architect: Operations Interface (OI)

- Using AICP
  - Application Architect Develop formal specification interfaces (operations) consistent with AICP

Designer: Design and implementation of operations

- Using formal specs. for interfaces
- Designer design and create actual interfaces using specific technology such as Web Services, SOAP, XML, etc

ASR To AICP

1. Type/Format (BLOB, TXT, XML, etc.): TXT, XML
2. Structure of Data:
3. Frequency of exchange: 3000 per day (.05TPS)
4. Size of unit of exchange:
5. Is it transactional: Yes
6. Online/ Batch: Online
7. Protocol: TCP/IP

AICP To OI

IO To Design and Implementation

11. XmlRecord input = new XmlRecord("ItemRecord");
12. MappedRecord output = rfact.createMappedRecord("ItemRecord");
13. String xml = "<?xmlversion='1.0'?><itemRecord>
<ItemCode>688829</itemCode>
<ItemDescription>
Axe Instinct Deodorant Bodyspray
</itemDescription>
</itemRecord>"
14. input.setData(xml);