Treating Patients Right
The Federal Healthcare Information Exchange (FHIE) in the US Government

MDA™ Implementers’ Workshop 2003
Tony Mallia
tmallia@ciber.com
FHIE Background

- State of the project
  - Near Term Solution initiated April 2001
  - In Production July 2002
  - Enhancements under test
  - Developers:
    - VHA SD&D
    - Northrop Grumman IT (Framework)
    - SAIC for DOD EI/DS
    - ESI for VHA
FHIE Near Term Overview

• Cooperating Healthcare organizations
  – 104 DoD Medical Treatment Facilities
  – 174 Veterans Health Administration Hospitals

• Provide VHA access to DoD health information under the necessary regulations

• Provide better healthcare
  – Previous history
  – Patient Safety
Patient Introduction

Check if Known

Cross Domain

Not Known

Introduction

Not Known

Domain

Known

Local Identity

Introduction

Domain

Check if Known
HL7 in FHIE

- Admission - Discharge - Transfer (ADT)
  - Patient Encounter with Diagnoses and Procedures
  - Discharge Summary
  - Allergies

- Laboratory Orders and results
  - Chemistry
  - Microbiology
  - Anatomic Pathology

- Medications (Outpatient Pharmacy)
  - Orders and Dispensing
Computational Independent Domain Model in UML

• Less than 200 classes
  – Person
  – Common Acts
  – Laboratory
  – Medications
  – Patient Encounter
  – Working Lists
  – Data Types
Record Templates

• Semantic Paths
  – UML Classes are types
  – UML Association End and Attribute names are roles

• Mapping semantics
  – Use UML dot notation for a single concept

• Relationships of Templates to Domain Model
  – Template is a Root class, Class Diagram and the first Association
Generation of the template data dictionary

patientEncounter.reason.a_qualified_code.coding_scheme_id.authority
patientEncounter.reason.a_qualified_code.coding_scheme_id.naming_entity
patientEncounter.reason.a_qualified_code.a_code
patientEncounter.reason.preferred_text
patientEncounter.comments.value
patientEncounter.comments.language.coding_scheme_id.authority
patientEncounter.comments.language.coding_scheme_id.naming_entity
patientEncounter.comments.language.a_code
patientEncounter.reasonComment.value
patientEncounter.reasonComment.language.coding_scheme_id.authority
patientEncounter.reasonComment.language.coding_scheme_id.naming_entity
patientEncounter.reasonComment.language.a_code
Platform Specific Formats

• Internal format - Java
• Connector Formats -
  – CORBA COAS Observation Data Structures
  – CORBA PIDS Traits
  – HL7 messages
Layers of Design

• Enterprise collaboration
  – Only Connectors are visible

• Component Collaboration
  – Framework is visible

• Component internal design
  – Classes are visible

• Recursive inward refinement
Enterprise Collaboration

- Use Case driven
- Collaboration diagrams to system level
- Data Dictionaries and ICDs
- Reference Information Model
- Template definitions
FHIE006.I6 (Deliver Patient Records) Use Case

Framework Boundary

COAS Query Access Adapter

: Client of Framework

: QueryAccess

: ObservationDataIterator

get_observations_with_policy(who, what, when, qualifier, policy, max_sequence, the_rest)

next_n(n, observation_data_seq)
FHIE Framework Architecture

• Graph Collections
  – Person Graph Collection
  – Record Graph Collection

• Templates and Graphs
  – Templates are the Record structure
  – Graphs are the Record instances

• Adapters convert from communication to/from GraphCollection operations
  – Push - Add
  – Pull - GetSelected
### Graph Collection

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getSelected(criteria : Selector, attrCriteria : AttributeSelector) : void</code></td>
<td>Get selected graph(s) based on criteria</td>
</tr>
<tr>
<td><code>getNext() : Graph</code></td>
<td>Get the next graph</td>
</tr>
<tr>
<td><code>getNext(n : int) : Graph[]</code></td>
<td>Get the next n graphs</td>
</tr>
<tr>
<td><code>addGraphs(graph : Graph[]) : boolean</code></td>
<td>Add graphs to the collection</td>
</tr>
<tr>
<td><code>getGraph(graph : Graph) : Graph</code></td>
<td>Get a specific graph from the collection</td>
</tr>
<tr>
<td><code>mergeGraphs(oldGraph : Graph, preferedGraph : Graph) : boolean</code></td>
<td>Merge graphs, prefering one</td>
</tr>
<tr>
<td><code>commit() : void</code></td>
<td>Commit changes to the collection</td>
</tr>
<tr>
<td><code>rollback() : void</code></td>
<td>Rollback changes to the collection</td>
</tr>
<tr>
<td><code>done() : void</code></td>
<td>Finish and clean up</td>
</tr>
</tbody>
</table>
Platform Independent Query

IGraphCollection.getSelected(criteria:Selector)

Selector
AND : String = "AND"
OR : String = "OR"
operation : String

0..*criteria

Criterion
weight : Float
EQUALS : String = "=="
NOTEQUALS : String = !=
LESSTHANEQUALS : String = "<="
LESSTHAN : String = "<"
GREATERTHAN : String = ">
GREATERTHANEQUALS : String = "=>"
CONTAINS : String = "["

Who_Criterion
+value 1..*
QualifiedPersonID
id : String

QualifiedName
local_name : String

AuthorityID
+authority_id
authority : int
naming_entity : String

When_Criterion
+value 1
TimeSpan
start_time : String
stop_time : String

Qualifier_Criterion
+value 0..*
DomainValue
absent : boolean

What_Criterion
+value 1..*
Platform Independent Computational Model

• Component Types
  – Adapters
  – Person Graph Collection
  – Clinical Graph Collection
  – Graph Collection Manager

• Adapter Operation driven
  – Collaboration diagrams to component level
Patient information is in HCS 11A and 2B and the patient is known in Nodes 11 and 2

get_observations

get_observations_by_time()
Adapter Design

• Implicit Mapping
  – Template is the message structure
  – Formatting of the message is to a programmatic rule
  – Adapter is simple

• Explicit Mapping
  – Message structure is the external Interface Control Document (ICD)
  – Elements must be converted to the Template
  – Elements are mapped or ignored
  – Mapping tables drive the adapter development
Component Internal Design

• Component façade operation driven
  – IGraphCollection operation definitions

• Class definitions

• Supporting class libraries
  – Datatypes
  – Coding schemes
  – Framework - other Components

• Sequence Diagrams to support Component operations
Testing

• Component test
  – tester always invokes IGraphCollection

• Subsystem test
  – tester invokes external connectors

• System test
  – user testing
Integrating Commercial Components

• Adapter wraps the commercial component

• Normalized to the Framework
  – All communication is normalized around
    • Graph Collection operations
    • Template record definitions

• Example of Commercial product integration
  – Commercial MPI wrapped at one stage of the project
  – Actual VHA MPI is wrapped in the VHA Domain
    Person Graph Collection
  – Commercial TQS wrapped during Proof of Concept
Wrapping Legacy systems

- Wrap the Legacy communication inside responding GraphCollection(s)
- Map the Templates to Legacy information
- Develop the adapter to transform the protocol
Wrapping Database access

• Databases are wrapped by the appropriate Graph Collection
• The Cache is wrapped with a SQLCacheGraphCollection and knows how to implement getSelected, getNext
• Graphs are serialized and persisted
Component Network Support

- Next steps
  - Component remote communication method hidden by GraphCollectionMgr
  - Protocol aware Proxy and Façade pattern buffers application form communication platform
  - Mix and Match protocols between Domains
    - RMI
    - CORBA
    - SOAP
    - JMS
Enterprise Integration

- Framework Architecture
- Domain Model and Templates
- Adapters and Components
- Component Model and Collaborations
- Distributed Development Process
- Enterprise Architecture Control