MDHT
Capabilities & Success Story

April 12th
Alexandria, Virginia

Rama Ramakrishnan - ONC
Cynthia Levy - ONC
Dan Kaplan – Mirth Corp.
Agenda

• Introduction to MDHT
• MDHT Architecture & Capabilities
• Existing MDHT Models
• MDHT Consolidated CDA Model
  – C-CDA Modeling Accomplishments
  – What Remains?
  – How do we move towards MU2
• Release Process
  – Open Source Development
  – Testing Process
  – Derived Documentation – Implementation Guide
• An MDHT User Story : Mirth Corporation
INTRODUCTION TO MDHT
MDHT – What & Why

• MDHT – Model Driven Health Tools under OHT.

• Motivation Factors:
  • Healthcare interoperability standards for clinical documents have:
    ➢ Steep learning curve: lengthy and complex specifications
    ➢ Lack of tooling: support for template model design & implementation
    ➢ No formal methodology/best practices for developing templates and implementation guides
  
• Current implementation approaches are inadequate
  ➢ XML processing technologies e.g. SAX, DOM, Xpath
  ➢ XML Binding techniques e.g. JAXB, EMF-XSD, XMLBeans
  ➢ RIM based approaches
MDHT Benefits

• Decreasing cost & Accelerate adoption of CDA R2.
• Provide standard OOAD-based methodology/tooling for modeling CDA templates
• Provide a model-driven framework for generating runtime API that support
  • Domain specific API (e.g. BloodPressureReading instead of Observation)
  • Construction of instances that conform to one or more templates
  • Consumption of XML instances that de-serialize into appropriate template
• Support the validation of instances against constraints defined in model
MDHT Users

- Healthcare IT Standards Developers/Publisher
  - Create new models/templates
  - Combine and extend existing models
  - Publish Implementation Guides (IG), IHE Profiles, Data Dictionaries

- Healthcare IT Standards Users/Implementer
  - Use generated runtime API in healthcare data exchange applications (e.g. EMR adapters to export/import CDA instances)
  - Minor modifications to existing models/templates
MDHT ARCHITECTURE & CAPABILITIES
MDHT Architecture

- **CDA**
  - Includes
  - CDA Support
  - CDA Templates
  - HITSP & IHE models

- **HL7 Core**
  - Includes V3 datatype support, MIF->UML support, HDF

- **Terminology Services**
  - Includes support for terminology sets

- **Custom Tools**
  - Includes
  - UML Table Editor
  - New Project Wizards
  - New CDA templates
  - Property Editors

---

**E M F**
- Includes
  - EMF Core
  - EMF.Editing
  - EMF.Codegen

**Model to Model Transformation**
- Includes Domain Model to implementation models.

**DITA Document Generation**
- Includes generation of DITA xml files

**JAVA Classes**
- Includes generating Java source code for models and the constraints

**Validation Suite**
- Includes testing instances of CDA and non-CDA models
MDHT Technical Overview

- MDHT uses the Eclipse Modeling Framework (EMF)
  - Core functionality can be applied to CDA & non CDA based models.

- UML Model Table Editor
  - Uses columnar spreadsheet like methodology, a tree based navigation structure and provides various user friendly features.
  - Can add /edit constraints (OCL, Property, Terminology based..)

- Model – Model Transformations
  - MDHT extends the EMF transformation capabilities for handling OCL constraints and CDA based templates
    - Ecore model – contains annotations with constraints from original UML model
    - Generator model – contains information on how to generate Java code from Ecore model
MDHT Technical Overview (contd.)

• Java API
  • Create / Edit CDA documents
  • Validation routines
  • Unit Test code generation for every constraint.

• Document Publishing
  • Generate Implementation guides based on the Model using the “SHALL”, “SHOULD” & “MAY” keywords
  • Uses the standard DITA-OT toolkit for document generation
  • Choice of formats (PDF, Eclipse Help, ePub, KindleMobi etc.)

• Handling Non CDA models :
  • Extensible to handle non CDA models.
MDHT Technical Overview (contd.)
MDHT – Inputs & Outputs

- Add Custom Validations & Constraints
- Domain Model
- MDHT
  - Java API (Source Code)
  - Validation Code Generation
  - Implementation Guide(s)
High Level View
High Level View (Contd.)

• MDHT Implementers create **UML Models** which express the:
  • Domain Template Model modeled as a UML Profile.

• Constraints modeled using Object Constraint Language (**OCL**).
  • Constraints allow data elements to have
    • Properties (e.g. Integer, 1, NULL)
    • Vocabularies (e.g. SNOMED CT, RxNORM)
    • Codes (e.g. HL7ActClass = SPLY, ActMood = INT)
  • Template-related constraints
    • Required Template id to be used
    • Required Template Relationships
      • Inheritance (e.g. From an IHE, CCD Template)
      • Association or Containment
        (e.g. ProblemAct has a Problem Observation)
Domain Template Model

- The following is an example of components from a HITSP C32 and C83 Template Model:
Do we want to use a Consolidated CDA example? Do you have one available

Cynthia Levy, 3/30/2012
Constraint

- Each Model Constraint is given a unique number within the model. This is known as a Conformance Rule
- From CCD:
  - **CONF-140**: CCD SHOULD contain exactly one and SHALL NOT contain more than one Problem section (templateId 2.16.840.1.113883.10.20.1.11). The Problem section SHALL contain a narrative block, and SHOULD contain clinical statements. Clinical statements SHOULD include one or more problem acts (templateId 2.16.840.1.113883.10.20.1.27). A problem act SHOULD include one or more problem observations (templateId 2.16.840.1.113883.10.20.1.28).
  - **CONF-141**: The problem section SHALL contain Section / code.
  - **CONF-142**: The value for “Section / code” SHALL be “11450-4” “Problem list” 2.16.840.1.113883.6.1 LOINC STATIC.
- From C-CDA:
  - SHALL contain exactly one [1..1] Problem Section (entries required) (templateId:2.16.840.1.113883.10.20.22.2.5.1) (**CONF:9449**).
  - SHALL contain exactly one [1..1] templated (**CONF:9179**) such that it
    a. SHALL contain exactly one [1..1]
      @root="2.16.840.1.113883.10.20.22.2.5.1" (**CONF:10441**)
Did we have
Cynthia Levy, 3/30/2012
Runtime Java API

- Convenience methods added to CDA implementation model to assist in building constraints and constructing documents
- Additional UML operations specified in the template model are carried through to the Java source code and can be implemented:
  - Directly in the model using OCL
  - By specifying the method body in the generated code
  - Gives the modeler the ability to add convenience into runtime API at design-time
- Annotations generated from template model used to populate runtime instance for default/fixed values, reducing the number of method calls required to build a document.
- Path Expression Support
  - Ability to create/query parts of the document
  - Transitional API for XML developers
Example of Client Code for HITSP C32 Summary

```java
PatientSummary doc = HitspFactory.eINSTANCE.createPatientSummary().init();
II id = DatatypesFactory.eINSTANCE.createII("2.16.840.1.113883.3.72",
    "CCD_HITSP_C32v2.4_16SectionsWithEntries_Rev6_Notes");
doc.setId(id);

ActiveProblemsSection problemList = doc.createProblemListSection();

Condition condition = HitspFactory.eINSTANCE.createCondition().init();
problemList.addAct(condition);

ProblemObservation obs =
    CCDFactory.eINSTANCE.createProblemObservation().init();
condition.addObservation(obs);

ProblemHealthStatus healthStatus =
    CCDFactory.eINSTANCE.createProblemHealthStatus().init();
obs.addObservation(healthStatus);
CE healthStatusValue = DatatypesFactory.eINSTANCE.createCE("xyz",
    "2.16.840.1.113883.1.11.20.12", "ProblemHealthStatusCode", null);
healthStatus.getValue().add(healthStatusValue);
```
Validation

• The items which are available to validate:
  • Constraints specified in the form of standard UML constructs such as cardinality
  • OCL constraints specified by the modeler or generated during the model-to-model transformation

• The output of validation is a diagnostic tree
  • Validation severity and message specified in the model are used at runtime
  • Diagnostic tree can be processed using CDA utility class
Validation

• User can decide the appropriate level of validation messages.

<table>
<thead>
<tr>
<th>API</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| ValidationResult::getAllDiagnostics()  
ValidationResult::getSchemaValidationDiagnostics()  
ValidationResult::getEMFValidationDiagnostics() | All Messages or specific to EMF or Schema issues. |
| ValidationResult::getErrorDiagnostics()  
ValidationResult::getWarningDiagnostics()  
ValidationResult::getInfoDiagnostics() | Messages that are generated for: SHALL, SHOULD,MAY |
| ValidationResult::getDiagnostics(Filter<Diagnostic>) | Specialized Filter |
| CDADiagnostic::getPath() | Gets the Xpath string for the message from the input CDA XML |
Implementation Guide

Chapter 2

DOCUMENT TEMPLATES

Topics:
- Consultation Request
- Consultation Summary
- Discharge Instructions
- Discharge Summary

This section contains the document level constraints for CDA documents that are compliant with this implementation guide.
Looking Ahead..

• Collaboration with other OHT Projects
• Publish Implementation Guide from template model
• Integration with Template Registry
• Integration with Terminology Services
• Extend approach to support all RIM derivations and serializations
• Integration with Semantic Web Technologies
• Model-driven design of services
EXISTING MDHT MODELS
Existing MDHT Models

- MDHT Models currently exist for the following CDA Domain Templates:
  - HL7 Continuity of Care Document (CCD)
  - HL7 Common Document Types
  - IHE Patient Care Coordination Technical Framework (PCC)
  - HITSP C83 Sections and Entries
  - HL7 Consolidated CDA (DSTU Dec 2012)
  - HL7 CDA IG: Public Health Case Reports (US Realm)
  - HL7 CDA IG: Personal Healthcare Monitoring Report (PHMR)
MDHT Consolidated CDA Model Accomplishments

• HL7 Consolidated CDA (C-CDA) was released as a DSTU Specification in December of 2011 and is referenced in the MU2 NPRM.

• MDHT has created a complete UML Model for the HL7 C-CDA DSTU.
  • This model includes all Document Templates, Section and Entry Level Templates specified by the DSTU.
  • All of the Constraints have also been modeled, including Vocabulary constraints for small static value sets.
ToC – Reference Implementation

• The MDHT C-CDA has been integrated into the S&I ToC Initiative Reference Implementation (RI)

  • The MDHT ToC Implementation models the 4 Information Packets using the full MDHT C-CDA Model
    • ToC Discharge Summary
    • ToC Discharge Instructions
    • ToC Consultation Request
    • ToC Consultation Summary

• Only a version of the Discharge Summary exists in the C-CDA IG, but the Medications Section is optional. So the MDHT ToC Implementation created unique Document Templates for the ToC Information Packets.
Items being worked on for CCDA

• During the integration process, a number of items were identified:
  • MDHT C-CDA modeling issues - reported by the MDHT team & the ToC RI team.
  • MDHT modeling limitations created by new features introduced by the HL7 C-CDA IG (Constraints with multiple keywords). - Resolution by April Release
  • HL7 C-CDA IG issues – These items have been submitted to the HL7 SDWG. (Resolution TBD.)
Moving towards MU2

- MDHT has plans for a V1.1 MDHT Release and will contain the C-CDA models. *(Scheduled to be released by the end of April 2012)*
- HL7 C-CDA IG issues/changes will be integrated with the next release of the HL7 C-CDA DSTU.
- Additional MDHT C-CDA changes will continue to be addressed as they are identified, prioritized and scheduled based upon the bi-monthly MDHT SPRINT planning meetings.
- If additional changes are made to the C-CDA to incorporate some of the ToC requirements for the MU2 NPRM, they can quickly be incorporated into the MDHT C-CDA Model.
MDHT RELEASE PROCESS
MDHT Methodology

• MDHT is an open source project being developed under the auspices of Open Health Tools http://mdht.projects.openhealthtools.org

• MDHT Follows the Open Health Tools development process and uses Agile methodology based on Scrum processes:
  
  • **Features** tracker maintains the list of activities that would be split across multiple user stories or sprints.
  
  • **Backlog** tasks & defects are used to maintain a complete list of product features that have been identified as necessary by the stakeholders and community.

  • Prioritized issues will be pulled into the current sprints and tracked to completion.

  • **User Stories / Defects** are selected for the 2 week **Sprint Duration**.
MDHT Tracker
MDHT Defect Tracker
MDHT Testing

• JUnit Testing is an integral part of the MDHT agile development process.
  • Automated regression testing of changes to the model and transformations.
  • A basic JUnit test requires at least one pass and one fail for each constraint.
    • JUnit Framework generates much of the surrounding java code so a developer will create two methods – updateToFail and updateToPass.

• MDHT maintains reports on the testing that is done on each release:
  • JUnit Results here [http://www.cdatools.org/reports/html/](http://www.cdatools.org/reports/html/)
  • Coverage report here [http://www.cdatools.org/reports/coverage.html](http://www.cdatools.org/reports/coverage.html)
MDHT Releases

• MDHT builds are typically created for each Sprint, and are available for Use.

• Builds include Release Notes, Derived Implementation Guides, JavaAPI and Validation Code.

• MDHT maintains a forum for Users to post questions [https://www.projects.openhealthtools.org/sf/discussion/do/listForums/projects.mdht/discussion](https://www.projects.openhealthtools.org/sf/discussion/do/listForums/projects.mdht/discussion)

• MDHT is planning a Formal Release V1.1 for the end of April. This includes an Intellectual Property review by IBM. This release will include all of the Models developed by MDHT.
MDHT Releases
How can you participate in developing MDHT?

• MDHT operates under the auspices of the Open Health Tools.

• Access to the Open Health Tools can be requested at:
  • https://www.projects.openhealthtools.org/sf/projects/fhims/

• Once subscribed to the Open Health Tools, MDHT can be found at:
  • https://www.projects.openhealthtools.org/sf/projects/mdht/

• There are a number of ways to become involved. Become:
  • Part of the *Developer Discussion Forum*
  • A submitter of *bugs and Enhancement Requests*
  • A contributor of *Source Code*
  • Eligible to be elected as a *Committer* (full development rights)
Who is currently using MDHT?

- The following is a list of Organizations currently using MDHT within their development environment. (This is only a partial list.)
  - IBM
  - Mirth Corporation
  - Orion Health
  - ONC S&I Framework
Questions !!!
AN MDHT USER STORY: MIRTH CORPORATION
Mirth and C Documents

- Mirth has a need in many projects to parse and generate C documents
- C documents are complex, simplify it
- Developers not experts with C documents; should not have to become experts - focus on application data and not on message formats
Mirth CDAPI

• A high level abstraction of MDHT
• CDAPI can be used to produce C documents or convert C documents into an easy to understand Java class called the ClinicalDocumentModel
• For cases where we need more flexibility, CDAPI can expose MDHT to have control over the low level details
Why Mirth chose MDHT for CDAPI

- Abstraction: Lets you work at a higher level than the raw XML
  - `createAdvanceDirective()` vs `<observation…>`
- Boilerplate: Generates required values for you (template ids, etc.)
- Leverages the Java language (type safe, object oriented, etc.) and therefore is less error prone than generating/consuming XML directly
- Simplifies the amount of code you write (XPath and DOM are verbose)
- MDHT has a helpful and responsive community
- It is open source (contribute feedback, bug fixes)
Inputs and Outputs of CDAPI

Clinical Document Model → CDAPI → CDA

CDA → CDAPI → Clinical Document Model

MDHT

CDA

Clinical Document Model
Structure of the ClinicalDocumentModel
CDA XML compared to MDHT code

```xml
<observation classCode="OBS" moodCode="EVN">
  <templateId root="2.16.840.1.113883.10.20.1.54"/>
  <code code="ASSERTION" codeSystem="2.16.840.1.113883.5.4"/>
  <statusCode code="completed"/>
  <value xsi:type="CD" code="247472004" codeSystem="2.16.840.1.113883.6.96"
    displayName="Hives"/>
</observation>
```

```java
ReactionObservation reactionObservation =
    CCDFactory.eINSTANCE.createReactionObservation().init();
reactionObservation.setCode(new Code("ASSERTION", "2.16.840.1.113883.5.4").getCD());
reactionObservation.getValues().add(new Code("Hives").getCD());
```
MDHT GLOSSARY

• **API** – *Application Programming Interface* is a source code-based specification intended to be used as an interface by software components to communicate with each other.

• **CDA** – *Clinical Document Architecture* is an exchange model based upon HL7v3

• **EMF** – *Eclipse Modeling* is a modeling framework and code generation facility for building tools and other applications based on a structured data model.

• **OCL** – *Object Constraint Language* is a declarative language for describing rules that apply to UML.

• **OOAD** – *Object-oriented Analysis and Design* is a software engineering approach that models systems as a group of interacting objects.

• **UML** – *Unified Modeling Language* is a standardized general-purpose modeling language in the field of object-oriented software engineering.

• **XML** – *Extensible Markup Language* is a markup language that defines a set of rules for encoding documents in a format for both human-readable and machine-readable format.