EMFT 1.0
Release Review
(OCL, Query, Transaction, and Validation)

June 16, 2006

Christian Damus
EMFT Developer
IBM, Ottawa

Agenda

- Features
- Non-code Aspects
- API
- Tool Usability
- Architectural Issues
- End of Life
- Bugzilla
- Standards
- UI Usability
- Schedule
- Process
- Communities
- IP Issues
- Project Plan

Based on Release Review Guidelines version 032 (Jan 15, 2006)

EMFT 1.0 Features – OCL, Validation and Query

- OCL: The OCL component provides capabilities for queries, constraint parsing, constraint validation and content assist for user models. It defines the API for constructing, validating, and evaluating OCL queries and constraints on EMF model elements. The OCL expression syntax is used to implement OCL queries and constraints. It provides support for OCL syntax completion and parsing on the meta-model and user model level.
- Validation: The validation component provides capabilities used to ensure model integrity.
- Query: The query component provides capabilities to specify and execute queries against EMF model elements and their contents

EMFT 1.0 Features - Transaction

- Transaction: The transaction component provides a model management layer built on top of EMF for managing EMF resources. It provides API that include extensions to the EditingDomain and related APIs of the EMF.Edit framework, and an internal model of transactions. It consists of two layers: a non-Eclipse core, providing primarily the "transaction model", and an Eclipse workspace integration layer.

EMFT 1.0 Features (contd.)

- Additional Noteworthy Features:
  - 114921 IOclHelper should evaluate OclExpressions (pre-parsed)
  - 109469 EcoreEnvironment to configure arbitrary EPackage.Registry
  - 114105 Adoption of ICU4J requirements - OCL, Query and Validatio...
  - 121753 Write globalization tests - OCL, Query and Validation Com...
  - 112953 OCL engine lacks support for stereotypes
  - 113884 Provide ability to recognize attribute inheritance by the...
  - 114106 Remove dependency on ANTLR from the OCL component
  - 127113 Implement changes in OCL 2.0 final specification

Non-code Aspects

- Documentation available at http://download.eclipse.org/technology/emft/javadoc/index.php
- Localization/Externalization
  - Adopted ICU4J as per platform requirement
  - Translation verification test-cases updated
- Increased JUnit / automated testing coverage
- Complete SDK/Documentation/Tutorials and Examples for OCL, Validation, Query and Transaction components in the examples feature
- OCL article submitted by committer Christian Damus for review (https://bugs.eclipse.org/bugs/show_bug.cgi?id=143580)

API – Validation, OCL, Query and Transaction

- Either public or within ‘internal’ namespace (no “provisional”)
- API is Eclipse Quality (http://www.eclipse.org/projects/dev_process/eclipse-quality.php)
- The project is operating fully in the open using open source rules of engagement
- Full use of Bugzilla, newsgroups, mailing lists. Published development guidelines and build process

API – Validation

- The validation component provides the following capabilities.
  1) Constraint Definition - Provides API for defining constraints for any EMF meta-model (batch and live constraints).
  2) Customizable model traversal algorithms - Extensibility API to support meta-models that require custom strategies for model traversal.
  3) Constraint parsing for languages - Provides support for parsing the content of constraint elements defined in specific languages. The validation framework provides support for two languages: Java and OCL. Validation also supports the "EMF" language for hooking up EMF-style constraints to the framework
  4) Configurable constraint bindings to application contexts - API support to define "client contexts" that describe the objects that need to be validated and to bind them to constraints that need to be enforced on these objects.
  5) Validation listeners - Support for listening to validation events.
  6) Eclipse Integration - The validation framework (through the UI plugin) provides support for requesting a UI dialog to come up for live validation errors/warnings in a particular validation context. The framework provides support for eclipse markers (e.g. MarkerUtil)

API – OCL

- The OCL component provides the following capabilities to support OCL integration.
  1) API for the OCL abstract syntax model, including the visitor pattern for processing ASTs.
  2) Provides API for parsing and evaluating OCL query and constraint expressions
  3) OCL expressions are used to implement OCL queries (Query) and constraints.
  4) API for OCL integration into the Query framework
  5) OCL already supports stand-alone deployment, though the packaging must be tweaked by the consumer

API - Transaction

- The transaction component provides the following capabilities.
  1) Multi-threading - Supports a protocol for clients to read and write EMF models on multiple threads.
  2) Model Integrity - Semantic integrity is ensured by automatic validation to detect invalid changes and semantic procedures to proactively maintain correctness of semantic dependencies.
  3) Batched Events - Clients are notified of groups of related changes in batches, rather than as a stream of EMF notifications. In particular, this allows applications to analyze change sets in their entirety.
  4) Undo/Redo - For a simplified programming model, the API automatically tracks changes applied to models without the need for client code to use EMF edit Commands. These changes are encapsulated in transactions/operations that can undo and redo themselves.
  5) Editing Domain - Support cooperative editing of models by multiple editors/applications. EMF resources can be shared amongst different editing domains.
  6) Eclipse Workspace - The API provides traceability between EMF resources and workspace resources.
  7) Eclipse Operations - The API supports the Eclipse operation history as an undo stack for undo/redo of resource changes. The API provides a framework for undoable operations that automatically capture undo/redo information, which can be interleaved on the same history with dependent operations that do not modify the EMF model.

API - Query

- The query component facilitates the process of search and retrieval of model elements of interest in a flexible yet controlled and structured manner. Provides API support for the basic EObject based Condition objects that are used to formulate queries for EMF models.
  The query component provides the following classes/interfaces to support queries.

  1) The IEObjectSource interface provides the search scope elements to be used in a query.
  2) The SELECT class implements a template-function that does the iteration over model elements and applies the search condition on each; it collects the resulting elements into a QueryResultSet object and returns it to the caller.
  3) The FROM class represents the elements to search. It is responsible of providing an appropriate iterator for the elements in the search space.
  4) The WHERE class applies the search conditions over the elements in the search set.
  5) The UPDATE class passes the elements who satisfy the search condition to a caller-supplied modification function. It collects the modified elements into a QueryResultSet object and returns it to the caller.
  6) The QueryResultSet class represents the set of elements returned by a given query.
  7) The EObjectCondition class is the abstract parent of all conditions that deal with model-elements (i.e., EObjects). It incorporates the services of a PruneHandler in order to answers whether or not to prune the element tree at a specific element and thus ignore its children.
  8) The ConditionPolicy class is used to allow the user to decide how to apply a given condition on a collection of a model-element EAttributes or EReferences values. Supports both the: exists (ANY) and for-all (ALL) semantics.
  9) The EObjectStructuralFeatureValueCondition class is the parent class for conditions that are responsible for checking the values held in model-elements' attributes or references.

Tool Usability

- EMFT provides a comprehensive set of examples for the OCL, Validation, Query and Transaction components.
- These examples are backed by tutorials that walk the user through the code (API usage) in these examples.
- These examples are available through the SDK feature and can be installed into the workspace by the user using the Eclipse Examples Wizard.
- The examples demonstrate the functionality using a library example meta-model

Architectural Issues

- Core architecture is stable
- EMFT introduces non-critical “new” features as internal or in examples (OCL Example, Validation Example, Transaction Example)
  - Gives us a chance to address any problems, revamp architecture if necessary, gather usage data
- Execution independent of Eclipse - Next release must determine the level of Eclipse independence that EMFT should support
- Query JUnits currently depend on UML metamodel, which adds an artificial build dependency on the UML2 project
- The Query infrastructure could be made more optimal in its searches in the next release

End of Life

- This is 1.0 Release. So all API is new.
- No classes / methods were deprecated in 1.0

Bugzilla

- Between October 5, 2005 and today:
  - 157 reports were created
  - 149 were fixed
  - 7 bugs deferred
- Existing P1s and P2s – 0
- Bugs outstanding – 2
- Zero blockers or critical bugs remain

Standards

- Relevant standards:
  - In use by EMFT OCL:
    - OCL - Object Constraint Language (OCL 2.0 Specification)
      - EMFT OCL based on ptc/2005-06-06 revision
      - Syntax compliance: support for parsing the complete grammar, with all appropriate type conformance checks. Most well-formedness rules are implemented
      - XMI compliance: EMFT OCL supports XMI serialization of expressions according to the Abstract Syntax, but deviates slightly in the definition of the standard library from the recommendations in Section 13.2 vis-a-vis data type operations. Also, some EMOF/UML types are not imported from their respective schemata but are replicated in the OCL schema where Ecore provides none
      - Evaluation compliance: support for allInstances()
      - Loosely aligned with EssentialOCL, with Ecore as the metamodel. Provides API extensibility for UML constructs (CompleteOCL)

UI Usability

- EMFT tools use Eclipse platform APIs to provide accessible UI
  - No outstanding accessibility bugs
- All text is externalized, and has been translated, including BiDi
  - Translation verification testing in progress, to complete mid-June

Schedule

- Development schedule closely followed the Eclipse Project's
  - Milestones every 6 weeks, trailing Eclipse's by one week
  - API freeze and feature complete at M6
- Now settled in to bi-weekly RCs, scheduled to end June 26, 2006

Process

- EMFT is developed using an open, transparent, and inclusive processes – this release follows its charter principles
- EMFT makes appropriate use of Bugzilla, mailing lists and newsgroups
  - All changes are described by a Bugzilla report
  - Bugzilla used to compile release notes for every integration, milestone, and release candidate build
  - Release notes link bugs to CVS commits
- There were no committer elections/removals during the 1.0 release cycle, although contributions were made by 2 new contributors

Communities

- Continuous effort to build community through newsgroup and Bugzilla responsiveness
  - Need to do more to promote visibility of all four projects and growth of the committer base
- Increased interaction in Bugzilla and on newsgroup
- Continued coordination with EMF, GMF and GMT projects
- The EMFT OCL project is based on the software developed by the LPG Project ("LPG") at SourceForge.net.
- EMFT OCL committers should participate more directly in the development process of the OCL specification in OMG
- Orientation with new Modeling top level project
- EMFT is one of the “live” modeling projects at Eclipse.org

Checkpoint – Exiting Incubation

- ✓ A working and demonstrable code base
- ✓ Active communities:
  - Active user community (4+ groups in IBM Software Group have confirmed usage, representing 5+ products, plus services engagements)
  - GMF leverages EMFT OCL, Query, Transaction, Validation
  - GMT leverages EMFT OCL
  - Active tool user community
  - An active multi-organization community
    - 5 committers + 2 contributors from 2 organizations
- ✓ The project is operating fully in the open using open source rules of engagement
  - Full use of Bugzilla, newsgroups, mailing lists
  - Published development guidelines, FAQs, build process, etc.
- ✓ The project team members have learned the ropes and logistics of being an Eclipse project (the project "gets the Eclipse way“)
  - Abides by Eclipse Development Process, posted guidelines, IP policies, open and transparent nature, etc.
- ✓ Now included in the Eclipse Modeling Project

IP Issues

- About files and license files are complete and correct
- All significant non-committer code contributions have been documented in the project log and reviewed by the Eclipse Foundation's legal staff
  - Initial IBM contribution OCL, Query, Validation and Transaction
  - Christian Vogt's contribution of the OCL code based on the LALR parser generator.
- All third-party libraries have been documented in the release
  - **LALR Parser Generator v1.0 (**pending review by the Foundation's legal staff)
  - The OCL plug-in is based on the software developed by the LPG Project ("LPG") at SourceForge.net.
  - The content in the org.eclipse.emf.ocl.internal.parser package ("OCL Parser") includes grammar definitions derived from LPG and code generated by LPG from these grammar files.
- A project Log has been created and reviewed by the Foundation's legal staff, and is available at
  http://www.eclipse.org/emft/eclipse-project-ip-log.csv

Project Plan

- No draft plan is available yet for the next release
- Items being considered for the EMFT OCL plan include:
  - Continued API and grammar updates as the OCL 2.0 specification evolves, possibly following the pattern of the UML2 project
  - Better support for UML and clearer distinction of EssentialOCL (for EMF/Ecore) and CompleteOCL (for UML), especially for XMI compliance and API usability

Questions

- Thank you!

