RSD Extensibility with consideration of UML Debug and Trace

Steven R. Shaw

January 29th, 2008
Agenda

- RSD Tooling Introduction
- RSD Architecture
- Extensibility API
  - EMF
  - UML2
  - GEF
  - GMF
  - Modeler (RSM / RSD)
- Example plug-ins (Walk-through)
IBM Rational Systems Developer v7.0.5

Leveraging the Eclipse platform...
to provide a complete lifecycle solution for systems development

- UML-based, model-driven development, design & construction tool for C/C++, J2SE and CORBA IDL implementations
- Serves as a framework for enabling business partner value-add development

Rational Systems Developer

- UML visual editors
- Architectural structure review & control
- Lifecycle integrations
- Transformations & patterns
- UML modeling

Rational ClearCase and Rational ClearQuest Integrations

Eclipse

Java Development Toolkit (JDT)
C/C++ Development Toolkit (CDT)
Device Software Development Platform (DSDP)
Solution: Extensibility

- Open application program interface (API) to support customizing and extending the modeling environment
- UML profile creation and editing to customize the properties stored in UML models
  - Allows organizations to develop plug-ins and customize the analysis and design tools for their environment and process. Supports the creation of an ecosystem allowing vendors to develop integrations.
- Comprehensive extensibility infrastructure for creating specialized extensions to the product
  - Leverages Open Source API and frameworks (UML 2.1, EMF, GEF)
  - Extension points for UI, menu, layout, command management, query
  - Extensions created in Java using Eclipse plug-ins
  - “Pluglet” support for lightweight scripting using Java
  - Provides User assistance using wizards and samples
Solution: Extensibility

- New reminders framework
  - Built on EMFT Validation Framework, and Model Indexing.
  - Reminders integrate into Tasks View
  - Reminders are always live, keep the user informed about the missing items (best practices, incomplete model) in the model.

- New queries framework
  - Contribute to Explore Palette using System Queries
  - Creation Executors for custom query types
  - Create or reuse result presenters for custom query types

- New model analysis and metrics framework
- APIs for customizing/consuming the model compare-merge framework
- OCL Integration
RSD Architecture

- Eclipse-based application.
  - Java / XML foundation allowing multi-platform support
    - i.e. Linux (Redhat / SUSE), Solaris 10, Win32, Vista
  - Allows for vertical or horizontal deployment (individual client vs. shell sharing)
  - Open source components – transparent development of core. Based on open standards, OMG specifications. 3rd party integrations
  - OSGI plug-in architecture – allow for strict versioning, update capability and on-demand loading for enterprise scalability.
RSD Architecture – Open Source Components

- RSD is built on a foundation of open source components.
  - Eclipse platform / SWT / JFace
    - http://www.eclipse.org/platform/
  - GEF (Graphical Editing Framework)
    - http://www.eclipse.org/gef/
  - EMF (Eclipse Modeling Framework)
    - http://www.eclipse.org/modeling/emf/
  - GMF (Graphical Modeling Framework)
    - http://www.eclipse.org/gmf/
  - UML2 (UML meta-model)
    - http://www.eclipse.org/modeling/mdt/?project=uml2
  - Others
    - JET (Transformation Framework)
    - BIRT (Reporting Framework)
    - TPTP (Test and Performance)
RSD Architecture – Dependency Block diagram

IBM Rational Systems Developer

RSM Modeler Components

TPTP

UML2

BIRT

emf

eclipse

JET
Extensibility API – EMF

- org.eclipse.emf.*
  - Data management: Persistence mechanism using XML, Robust editing, generative capabilities
  - Whole books have been written on EMF extensibility!
  - In a nutshell: domain semantics are defined using a meta-meta-model ECore. This is generated into a set of interfaces and corresponding implementation classes. Interfaces are public. A Factory class is used to instantiate the implementation and an interface is returned.
  - EPackage class define constants
  - Other useful utility classes - EcoreUtil
Extensibility API – UML2

- `org.eclipse.uml2.uml.*`
  - UML2 meta-model is generated using EMF.
  - Set of interfaces with utility methods for creation
  - UMLPackage
    - EClass constants
    - StructuralFeature constants
  - UMLUtil – more advanced function for package merge etc.
Overview of GEF (Graphical Editing Framework)

- Model – View – Controller Architecture

Model

Business Model

java.lang. Object

Controller

GEF

org.eclipse.gef.EditPart

View

Draw2D

org.eclipse.draw2d.Figure
GMF (Graphical Modeling Framework) Diagram Architecture

- Model
  - Business Model
- Controller
  - GEF
- View
  - Draw2D

- «semantic» EObject
  - «notational» o.e.g.notation.View
- «notational» commands
  - IGraphicalEditPart
  - editpolicies
  - requests
- figures
- handles
- layouts
- routers
GMF Diagram runtime dependency diagram

Controller

Model

View

org.eclipse.gef.GraphicalEditPart

org.eclipse.gmf.runtime.notation.View

org.eclipse.emf.ecore.EObject

org.eclipse.draw2d.IFigure

<Use>

<Use>

<Use>
GMF Notation meta-model

- `org.eclipse.gmf.runtime.notation.*`

- Separate meta-model from semantic model (UML2). Allows for separation of concerns. Can target different semantic models.

- Common utilities:
  - `ViewUtil`
Services
Services

- IconService: retrieve an icon based on an element type
- ParserService: retrieve a parser for an element type
- ModelingAssistantService: diagram assistant for a diagram
- ViewService: retrieve view factory to construct view
- EditPartService: gets the controller for a given view
- EditPolicyService: allows installation of an editpolicy on existing editparts
- PaletteService: definition and customization of palette drawers
- DecorationService: decoration of a EditParts with adornments.
- LayoutService: layout / arrange a diagram
Extensibility API – Modeler (RSM / RSD)

- `com.ibm.xtools.modeler.ui.UMLModeler`
  - Main static entry point for model creation, save and open.
  - Sub helper interfaces
    - IUMLHelper – search methods
    - IUMLDiagramHelper – diagram, notation element creation / layout
    - IQueryHelper – EMF query API
    - IOclQueryHelper – OCL query API
Examples: plug-in walk-through

- **Parsing Trace Data**
  - Assumption: some trace data exists in some proprietary format. The trace information describes some behavior of the system that occurred during period of time.
  - Requirement: we want to visualize the trace information as a UML diagram (either Activity or Sequence / Interaction diagrams).

- **Decorating Diagrams**
  - Assumption: some diagrams exist that have representation / traceability into a runtime system.
  - Requirement: need to decorate elements in the diagram that are being traced

- **Animating Diagrams**
  - Assumption: trace execution can be played back or data can be retrieved during runtime
  - Requirement: diagram that corresponds semantically to the trace execution (activity / state / interaction etc.) will animate / highlight the elements showing the flow of execution.
Example: Parsing Trace Data

1. Visualize the trace data directly.
   - Data is in proprietary format – need to provide a mapping of that data to UML. There is an example plug-in in RSD that demonstrates how to do this →
   - Allows data files to be opened and visualized in UML format.
Example: Parsing Trace Data

1. **Generate a domain specific editor using the GMF generative framework.**
   - Assumes trace data is persisted using an EMF meta-model.
   - Can work well for “class diagram-like” domain visualizations.
     - End user not as concerned with UML compatibility
   - Possible to use Model-to-Model transformation framework in RSD to persist to UML model.
   - Compatible with RSD – allows co-visualization of domain specific models with UML.
Example: Parsing Trace Data

1. Trace data is persisted in UML format already
   - Programmatically create UML semantics by parsing trace data.
   - Extensibility will allow creation of diagram and “canonically” create diagram views.

   Example → com.ibm.xtools.mdsdp.example.trace
   - Command handler – generates trace data (simulating trace execution data capture).
   - Diagram is created for semantic data and layout is invoked.
Example: Decorating Diagrams

- **com.ibm.xtools.mdsdp.example.decorate**
  - Utilizes the Decoration service. A decorator provider is registered (through xml)
  - Provider (TraceDecoratorProvider) “provides” for specific semantic elements (OpaqueAction)
  - Decorator displays symbol on shape. (TraceDecorator)
  - Decorator listens for semantic changes to determine where symbol should be displayed or not.
Example: Animating Diagrams

- `com.ibm.xtools.mdsdp.example.animate`
  - Action is registered through `objectContribution` to start the trace animation from a certain element (`OpaqueAction`)
  - From the semantic element (model), the corresponding `IGraphicalEditPart` (controller) is found. From the editpart, the `IFigure` (view) is used to change the color of the foreground color temporarily.
  - Semantic object is used to navigate to next element through the `ActivityFlow`.

WinZip File
IBM Rational Partner Program

- No requirements to extend RSD
- Partner program provides web-site publication and extensibility validation
  - IBM PartnerWorld
Challenges

- **Extensibility**
  - Generic Open source vs. Modeler API. When to use each?

- **Trace:**
  - Data capture in EMF is expensive. Trace data could overwhelm the system.
    - Realistically a conversion process from optimized trace format to EMF is a prerequisite before visualization.
    - Drinking from a fire hose! What aspect of trace data should be visualized as UML.

- **Animation:**
  - Realtime animation is not practical since state changes can occur much faster than humans can process the visual data.
    - More practical for trace playback.
End