Remus – Creation Review

Submitter: Tom Seidel – Independent
Review Date: April 7, 2010
Communication Channel: eclipse.remus Newsgroup
Introduction

• Background
  – Motivation
  – Goal
  – Concepts

• Proposed Components

• Scope

• Relationship with other Eclipse Projects

• Community Feedback

• Organization
  – Initial Committers
  – Code Contribution
  – Plan
Motivation

- Efficient data management and fast access to information becomes more and more important in today’s information technology
- It needs many different applications to manage different types of information
- More and more data is stored “in the cloud“, which requires reliable tooling to access information while not connected to any remote system without duplicating information (offline availability)
- Missing “searchability” over multiple information repositories
- Today it is still very difficult to exchange data between applications which store information
Goals

• Tooling for information management
  – Standalone RCP/Set of bundles which provides a complete tooling environment for information management
  – Tools for project management, document management, multimedia and social services
  – Connectors to various remote repositories (e.g. Amazon EC2, Alfresco, Mediawiki, etc)

• Framework for building information-centric Eclipse based applications
  – Different integration levels for building custom applications
  – Remus provides APIs that can be reused in different contexts
    • Approximately 20 different extension points
    • Widget APIs
Concepts (1)

- Every information can be described via a structure definition of the data that represents an information.
- Example: Informationtype Book

  Book
  - Author: String
  - Year: DateTime
  - Title: String
  - ...

Information type is described declaratively and made consumable via an Extension-Point.
Concepts(2)

- Remus components are working with the information unit as data input and the referenced structure definition.
- All components can handle any type of input data as long as the input data can be referenced by a structure definition.

```
Information-Unit  What?  Indexing
                   Edit  Synchronization
                   Visualization
                   ... Rule-Engine
                    
How?  Structure-Definition
```
Proposed Components

• Core Components
  – Information Structure Definition
    • Provides a Framework and APIs to define an information structure and manipulating data on a high level API
  – Virtual File System
    • Provides API (based on the Eclipse File System) to create a custom persistence-layer, e.g. for encrypted data

• General UI Components
  – Viewer Component
    • Provides API for visualizing information structures in JFace Viewers including Context-Menues, Drag’n’Drop, Decorations, etc
  – Editor Components
    • High Level API for creating User Interfaces based on information structure definition, including a HTML Template Engine for visualizing information structures in an embedded browser
Proposed Components (2)

• Synchronization Components
  – Framework for synchronizing local data with a remote information repository, including user interfaces for visualizing change sets, updated items, etc.

• Rule Components
  – API for executing custom rules based on user-input (Groovy based). E.g. the user drops a string into the application and can select a rule (predefined or custom) which does something with the input, e.g. creates new data in the application.

• Search Components
  – Bundles to define declaratively which content nodes of an information unit needs to be indexed, API for searching references, etc.
Scope

• Maintain and improve the technology
  – Provide detailed user documentation
    • User scenarios/Screencasts
  – Developer documentation
    • Extension Point Description
    • JavaDoc of Service Interfaces
    • PDE Template Wizards
    • Examples

• Within scope if interest is expressed
  – New information types
    • Especially Information types with attached documents are highly requested
  – New repository connectors
    • Focus on generic connectors
  – RAP compatibility
    • Basic Framework functionalities, no plan to port 100% of all components
Relationship with other Eclipse Projects

• **EMF/EMFT**
  – Remus uses a generated EMF Model for manipulating data. All manipulations are executed with the EMF-Edit Framework on an EditingDomain
  – The Synchronization components are using EMF Compare to detect changes between remote and local information structures
  – EMFQuery is used to access the underlying EMF Model

• **ECF**
  – Remus makes use of the ECF Filetransfer API to download and upload files over several protocols

• **BIRT/DTP**
  – Remus includes a full BIRT Runtime, a specific ODA Driver to access local data for visualizing them in BIRT Reports and offers an integration of the BIRT Designer.

• **Mylyn**
  – Remus reuses some UI Components from Mylyn, e.g. the notification window, Date-Widgets and the Screenshot function
Community Feedback

• Remus already has a growing user community, which is not part of the Eclipse Community
  – We receive consistently positive feedback
  – Users are very interested in document management
  – Surprisingly many downloads for Linux based operating systems.

• Potential Trademark Issues
  – Janet Campbell raised concerns regarding the project name [1]
  – Alternative names are:
    • Rhea Information Management
    • Aenas Information Management
Committers/Mentors

• Committers
  
  – Tom Seidel (proposed Project Lead)
    Tom is an independent Software Architect and Developer for Eclipse Technologies with over 6 years experience in building applications based on RCP and OSGi. Prior to this he has worked as technical lead in a developer team which primarily built RCP applications for the medical and pharmacy industry.

  – Andreas Deinlein (proposed committer)
    Andreas serves as Software Architect for Siemens AG, Sector Healthcare where he develops and manages embedded software components for medical devices. He is also an enthusiastic C++ developer and has over 4 years experience in building plugins for the Eclipse Platform.

• Mentors
  
  – Ed Merks
  – Tom Schindl
Code Contribution

- Current Project is hosted at sourceforge.net
  - All code is licensed under the Eclipse Public License
  - Initial committers own all copyrights
  - Current bugtracker is JIRA [2]
  - Nightly builds are available via Bamboo Build System [3]
  - Code statistics are available via Ohloh [4]
Plan

- **Releases**
  - Current release is an unofficial Beta Version
  - 0.1 Release scheduled for Q2 2010

- **Plan**
  - Milestone release every 3 Month
  - 1.0 scheduled for Q2 2011
References

• [1] Eclipse Community Forums

• [2] JIRA System Dashboard
  http://remus-software.org/jira/secure/Dashboard.jspa

• [3] Bamboo Remus Dashboard
  http://remus-software.org/bamboo/

• [4] Remus Information Management – Ohloh
  http://www.ohloh.net/p/remus/analyses/latest