Software Architecture Survey Form (CS 746)

Name of system: Eclipse

Reviewer: Xuan Choo

Date: 5 Nov 2011

Author of software: IBM Canada (Object Technology International)

Author of book chapter: Kim Moir

Five star rating of book chapter: *****

Purpose of system:

The Eclipse project is an open source integrated development environment implemented using the Java language. The main goal of the Eclipse project is to develop a generic environment in which program development could take place. Another goal of the project is to create a modular and fully customizable environment that could be used to develop programs not just with the Java language, but with other programming languages as well.

The main purpose of the Eclipse platform is to develop software. Interestingly, the Eclipse platform is used to develop future iterations of itself. Because of its architecture, the core of the Eclipse platform (known as the Rich Client Platform) can also be used as part of Java applications. This allows developers to utilize the core features of Eclipse (e.g. the UI elements, and the plug-in environment) in their own applications, which may not be development related.

In the Eclipse platform, a plug-in is defined as modular component written in the Java language that implements a particular sub-feature of the eventual platform. This plug-in architecture allows developers to extend the functionality of Eclipse to encompass multiple programming languages, allowing it to become a truly generic integrated development environment.

Basic metrics -

KLOC: The Eclipse project has three major sub-projects: the Eclipse platform, the JDT (java development tools), and the PDE (plugin development environment). The lines of code count is listed for each of these sub-projects.

Eclipse platform

Source: Ohloh project page for Eclipse [http://www.ohloh.net/p/eclipse/analyses/latest]

Language	Code Lines	Comment Lines	Blank Lines	Total Lines
Java	1,705,860	1,005,031	303,193	3,014,084
XML	236,170	5,553	5,584	247,307
C	138,994	3,776	9,750	152,520
HTML	54,303	726	10,676	65,705

C++	31,094	1,020	2,402	34,516
CSS	4,752	1,103	1,102	6,957
MetaFont	3,623	0	9	3,632
JavaScript	2,374	447	342	3,163
shell script	742	107	44	893
XML Schema	374	4	64	442
DOS batch script	242	115	60	417
Make	124	85	42	251
SUM:	2,178,652	1,017,967	333,268	3,529,887

JDT

Source: Ohloh project page for the JDT [http://www.ohloh.net/p/eclipse-jdt/analyses/latest]

Language	Code Lines	Comment Lines	Blank Lines	Total Lines
Java	1,990,890	456,147	248,679	2,695,716
HTML	105,033	823	8,090	113,946
XML	37,244	1,883	2,962	42,089
MetaFont	1,215	0	2	1,217
JavaScript	735	121	35	891
CSS	246	29	30	305
shell script	1	0	1	2
SUM:	2,135,364	459,003	259,799	2,854,166

PDE

Source: Ohloh project page for the PDE
[http://www.ohloh.net/p/eclipse-pde/analyses/latest]

Language	Code Lines	Comment Lines	Blank Lines	Total Lines
Java	358,615	154,786	61,131	574,532
HTML	29,814	820	2,976	33,610
XML	27,337	2,033	1,862	31,232
MetaFont	3,904	0	164	4,068
CSS	627	69	120	816
XSL Transformation	358	10	3	371
XML Schema	112	15	3	130
SUM:	420,767	157,733	66,259	644,759

Project start-up: 7 Nov 2001 (v1.0 release)
Number of major releases: 11 (v1.0, v2.0, v2.1, v3.0-3.7)
Number of developers:

Active commiters: 592
Inactive commiters: 867
Total: 1,459
Source: Eclipse company / project commit details - [http://dash.eclipse.org/dash/commits/web-app/commit-count-loc.php]

Size of user community or number of installations:

Unknown, most probably in the millions.
Source: 2011 Eclipse community survey - [http://www.eclipse.org/org/community_survey/Eclipse_Survey_2011_Report.pdf]

Major stakeholders:

Many companies have a vested interested in the Eclipse project. Some of these companies were part of the original development team of Eclipse (indicated with a solid bullet point, italics indicate company is currently not actively making commits to Eclipse). All the remaining listed companies all have at least 1 active committer (to date).
Source: Eclipse commits history - [http://dash.eclipse.org/dash/commits/web-app/commit-count-loc.php?&sortBy=&show=]

- *Borland*
- IBM
- Merant
- QNX Software Systems Co.
- Rational Software (now IBM)
- Red Hat, Inc.
- *SuSE (now Novell)*
- TogetherSoft (now Borland)
  - Actuate Corporation
  - Adobe Systems
  - AGETO Service
  - Atos Origin
  - Business Systems Integration AG
  - CEA List
  - Cisco Systems, Inc.
  - Cloudsmith Inc.
  - Compeople AG
  - Continental AG
  - Eclipse Foundation
  - Engineering Group
  - Ericsson AB
  - Freescale Semiconductor
  - Google Inc.
  - Intalio Inc.
  - Intel Corporation
  - Itemis AG
  - Mentor Graphics
  - Mia-Software
  - Motorola
  - Nokia
  - Nuxeo
  - Obeo

- Open Canarias S.L.
- OpenMethods LLC
- Oracel
- Prosyst Software
- Remain BV
- SAP AG
- SAS
- Siemens AG
- Sierra Wireless
- Sonatype
- Soyatec
- SpringSource, Inc
- Tasktop
- Texas Instruments
- Thales
- Tieto
- Xored Software, Inc.
- Zend Technologies

Use of concurrency: No
Implementation language: Primarily Java.
Supporting software: Eclipse requires the Java runtime environment to be installed to run.

High level architecture -

Diagram of software architecture:

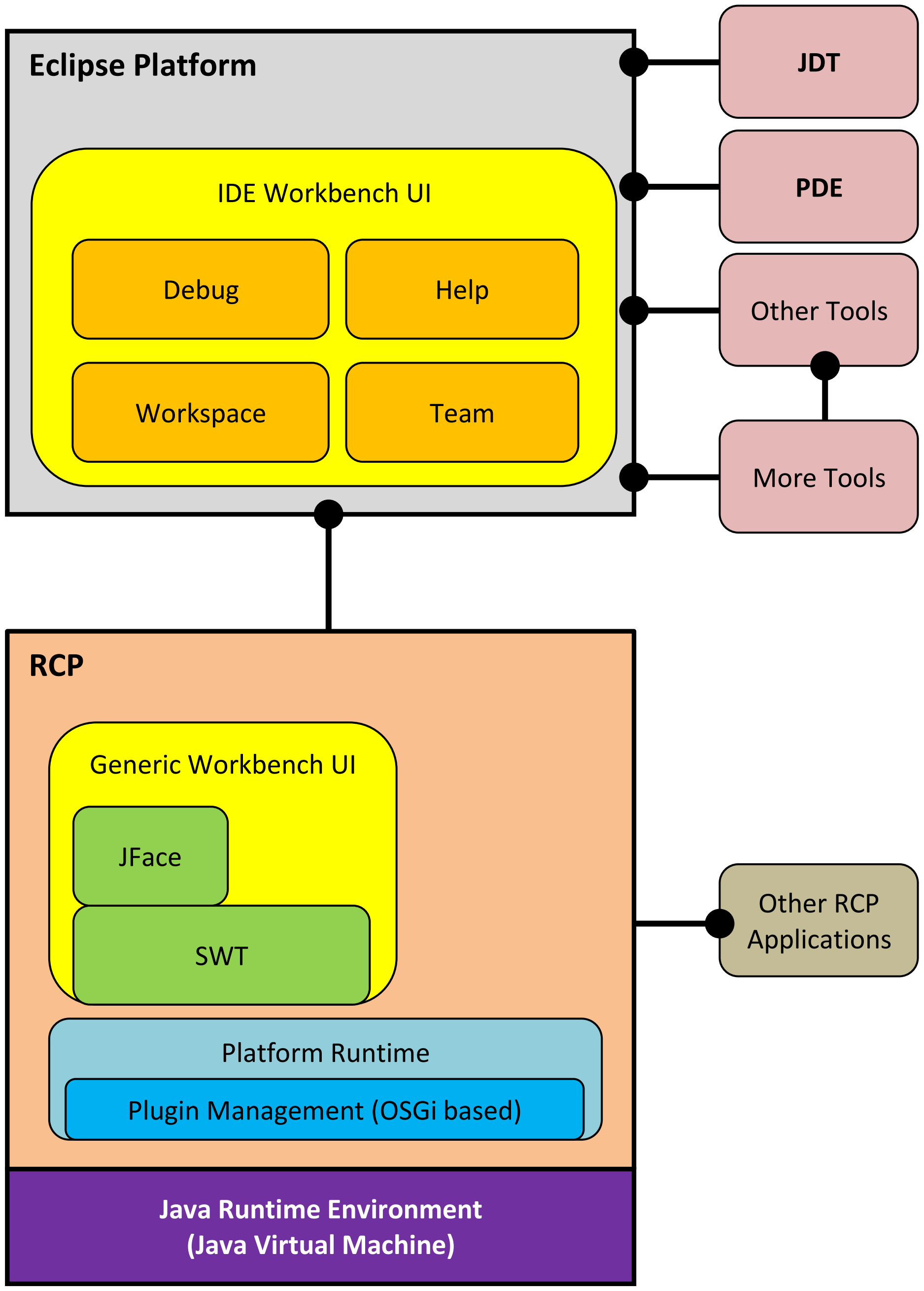

Figure 1 - Current Eclipse architecture. Lines with rounded terminations indicate plug-in implementation.

High level scenarios:

The following is a high-level scenario demonstrating the "lazy activation" principle of the Eclipse system. Note that in this scenario, the plug-ins are added to the plug-in registry but is not activated (i.e. the class file is loaded) until the plug-in is required.

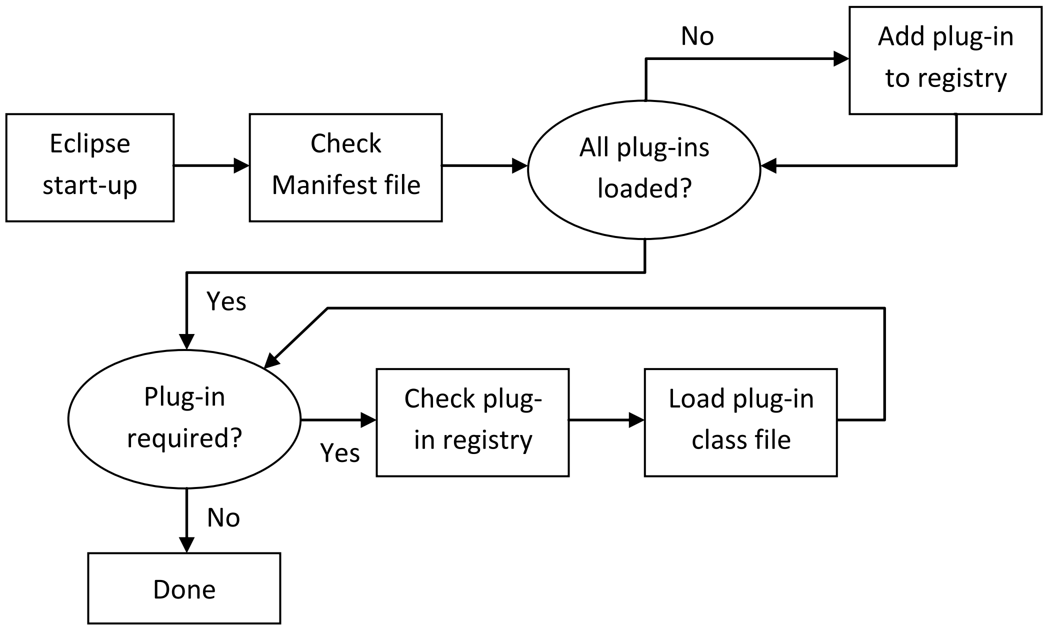

Figure 2 - High-level scenario of "lazy activation"

Data structures or algorithms:

Every feature in the Eclipse system is implemented as a plug-in. This way, the system is extremely modular, and can be customized to any working style. Interestingly, Eclipse utilizes a "lazy activation" approach to activating the installed plug-ins. With this method, plug-ins are only loaded when they need to be used.

Control flow and/or data key to the architecture (if any):

Eclipse utilizes manifest files to store information such as required runtime environments, plug-in version and plug-in dependency information. Eclipse also utilizes workspaces, which are used to contain files and metadata used by the user in their projects.

Architectural style:

Although Eclipse is composed entirely of plug-ins (even the core components are also considered plug-ins), the core of the Eclipse platform has a layered architecture. Outside the core of the Eclipse platform, Eclipse takes on an abstract data types and objects architecture, with each plug-in defining one abstract object.

Major evolutionary changes:

There have been a couple of evolutionary changes to the architecture of Eclipse. The first major architecture change is the adoption of the OSGi standard for managing the myriad of plug-ins for Eclipse. The switch to OSGi did not affect the visual layout of the architecture, it did alter the way in which the plug-in dependencies were handled (e.g. it is now possible to support dynamic loading and unloading of plug-ins).

Another major change to the architecture involved the refactoring of the platform to allow Rich Client Platforms (RCPs) to be developed. RCPs are essentially Java based applications that utilize the Eclipse platform, but do not require the IDE-related components to function. This moved allowed organizations to utilize the core features of Eclipse (SWT, JFace, a generic workbench UI, and the OSGi plug-in management system) in other applications not related to program development.

The following is a diagram showing the early architecture of Eclipse, with the changes indicated above shown. (Contrast this to Figure 1)

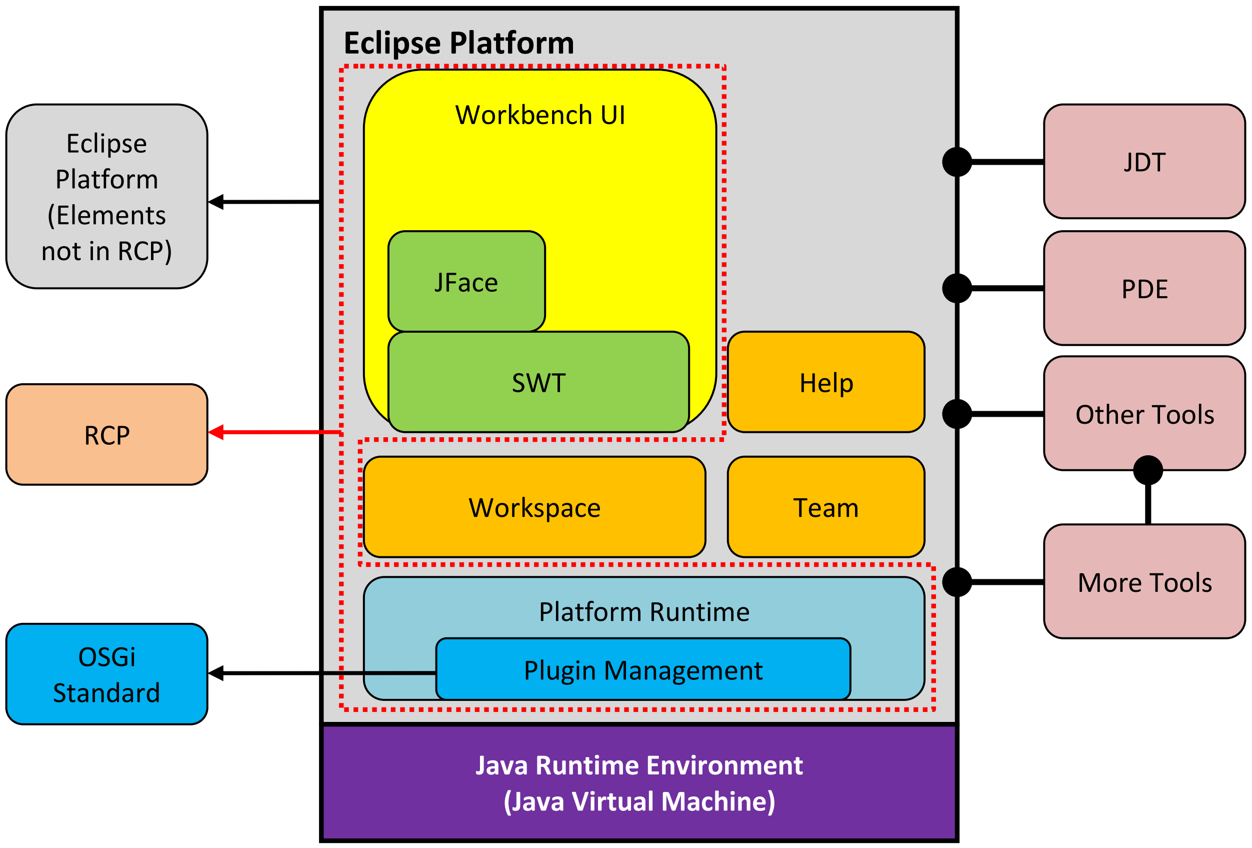

Figure 3 - Early Eclipse architecture with evolutionary changes indicated.

Performance bottlenecks:

The primary performance bottleneck is the Java runtime environment (JRE). Since the entire project is Java based, without the runtime environment, nothing will be executed. Additionally, performance hits to the runtime environment (e.g. CPU utilization slowing down the JRE) will affect the entire system.

Real time:

Real time support is not implemented in the Eclipse system by default. However, it is conceivable that a real-time plug-in could be developed for Eclipse.

Notation for architecture:

- IDE - Integrated development environment
- JDT - Java development tools
- PDE - Plug-in development environment
- SWT - Standard widget toolkit
- RCP - Rich client platform

Methodology:

Because of the large and wide spread nature of the development team, the development methodology is not easily definable. Each developer (committer) to the Eclipse project is allowed to make their own changes to the project as they see fit. There is however, a core team of Eclipse developers that work on "incubations" of new releases of the system. With this method, minor improvements are made to the released software, while the team makes the major changes and testing to the "incubated" project.

Appendix:

Kruchten's eight context attributes applied to Brown/Wilson systems -

1. **Size**: L (~7MLOC)
2. **Criticality**: Lo - Med (depends on implementation of platform)
3. **Age of system**: L (10 yrs)
4. **Rate of change**: Med (There have been 11 major releases thus far)
5. **Business model**: Open source
6. **Stable architecture**: Med (Within the 11 major releases, the architecture has evolved at least 2 times, with another major evolution expected for the upcoming v4.0 release)
7. **Team distribution**: Hi (Development team meets entirely on the internet)
8. **Goverance**: XHi (Although developers and testers are spread out globally and across multiple companies, there is a board of directors to oversee the project. A full-time staff is also employed to handle matters like marketing, IT support, system administration, and so forth)