Code Recommenders 1.0
Release Review

Johannes Lerch (lerch@cs.tu-darmstadt.de)
Marcel Bruch (bruch@cs.tu-darmstadt.de)
Sebastian Proksch (proksch@cs.tu-darmstadt.de)

Mentors
Chris Aniszczyk & Jochen Krause
About

The Code Recommenders project develops tools that automatically analyze large-scale code repositories, extract interesting data from it and integrate this information back into the IDE where it is reused by developers on their daily work.

The vision of the project is to create a context-sensitive IDE that learns from what is relevant in a given code situation from its users and, in turn, give back this knowledge to other users. If you like, you may think of it like a collaborative way of sharing knowledge over the IDE – or as „IDE 2.0“ (in accordance to Web 2.0).
Introduction

• Code Recommenders is a Technology sub-project.
  o See http://eclipse.org/recommenders

• This release is its v1.0 release.

• The goal of this release is to provide a stable, scalable and fast version of Recommenders that also offers first support for Java Standard Library.
Committer Diversity

- The project is getting international. Germany, Canada, China.

**Committers:**
- Andreas Kaluza (individual)
- Chen Cheng (individual)
- Doug Wightman (Queens)
- Johannes Lerch (TUD)
- Marcel Bruch (TUD)
- Sebastian Proksch (TUD)
- Stefan Henss (individual)
Contributors

Lots of contributions on Snipmatch template completion engine.
A GSOC on Snipmatch is running.
Project History

Based on Ohloh.net from 01.01.2012 to 09.06.2012

• **LOC Java (blue line):**

67 klocs Java code in 22.01.2012
51 klocs Java code in 09.06.2012

Code consolidation, and cleanups

• **Number of commits:**

May 2012
Commits: 133

• **Number of active committers:**

May 2012
Committers: 6
Feature Overview 1.0 – What’s new?

• **Model Repository**
  - Server-side JAX-RS service replaced by maven-like artifact repository.
  - Model repository moved completely to eclipse.org.
  - Extensible for other recommendation model providers.

• **Call Completion Models**
  - Support for Java Standard Library (java.*) and many more Eclipse Plugins added (based on Juno M6 release train repository). Updates until Juno Release planned.
  - **Much** smaller memory footprint for call completion models (~1-2 MBs compared to 100+MBs for very large models).

• **Misc**
  - Various algorithmic improvements, for instance in chain, subwords, and templates completion.
Feature Overview 1.0 – What’s new?

Naïve template completion is back in train; based on call completion:

```java
private void parse() {
    ASTParser p = ASTParser.newParser(0);
    ...
    dynamic 'ASTParser' - VL-2 - 9 %
    dynamic 'ASTParser' - VL-10 - 8 %
    dynamic 'ASTParser' - VL-9 - 7 %
    dynamic 'ASTParser' - VL-0 - 6 %
    dynamic 'ASTParser' - VL-20 - 6 %
    dynamic 'ASTParser' - VL-35 - 6 %
    dynamic 'ASTParser' - VL-29 - 5 %
    dynamic 'ASTParser' - VL-19 - 4 %
    dynamic 'ASTParser' - VL-12 - 3 %
    dynamic 'ASTParser' - VL-17 - 3 %
    dynamic 'ASTParser' - VL-26 - 3 %
    Press 'Space' to show Chain Proposals (Recommended)
```

Note that templates don’t reflect call order-constraints.
Feature Overview 1.0 – What’s new?

New word-boundary aware fuzzy matching for Subwords:

See [http://blog.deepakazad.com/2012/05/](http://blog.deepakazad.com/2012/05/) for more examples.
Non-Code Aspects

• User Guide:
  o Evolving. More documentation is coming for Juno release.

• Low activity on forum yet. Reasons include:
  o Limited set of APIs covered → added support for Java standard library. More to come.
  o Weak documentation → User manual is improving.

• Scheduled Talks:
  o 06/2012 4 Democamps (Braunschweig, Frankfurt, Hamburg, Zurich)
  o 07/2012: RheinJUG Java User Group, DE
  o 09/2012: JavaOne
Architectural Issues

No severe issues. Using maven-like artifact repository for model delivery seems to work out well. Juno will show.

Repository approach is also easily extensible by 3\textsuperscript{rd} parties for own recommendation models.
Provisional APIs

Most parts of Code Recommenders are extensions of existing extension points and are yet not intended to be extended by clients.

For 1.0 no new APIs have been added.

Tool Usability

- **Chain** and **template completion** are computational intensive, and thus, should not be placed on the default (=first) content assist list. Actually, they disable themselves as soon they detect they were enabled there.

- **Subwords** disables itself on default content assist as soon as it detects JDT or Mylyn content assist is enabled. As of now, Subwords does not support the full HTML content assist functionality as JDT.
UI Usability

• No NLS support.
• Yet very limited configuration options to tweak extdoc
• Various configuration options for completion engines added.
Standards

Yet, there are no standards the Code Recommenders project defines. The HTTP based model repository may, however, become a standard for delivering recommenders-like recommendation models. Juno release will show.
Privacy

Recommenders v1.0 does not collect any usage data nor does it provide any infrastructure to do so.
Bugzilla (as of 09.06.2012)

Bugzilla interactions since inception:

• **42 bugs** newly reported
• **32 bugs** closed
• **23 bugs** open
IP Log

• [https://bugs.eclipse.org/bugs/show_bug.cgi?id=380885](https://bugs.eclipse.org/bugs/show_bug.cgi?id=380885)
Incubator Project

• With its 1.0 release, Code Recommenders requests an incubator sub-project in accordance to the Eclipse Development Process, Section 4.9

• This incubator aims to provide a home for innovative research ideas related to mining software repositories and recommender systems to develop their tools and user community with relatively low administrative overhead.

• This incubator welcomes research projects of universities, research organizations, and individuals that are committed to develop and maintain a innovative product with the ultimate goal to eventually leave the incubator to become an independent project or join the hosting Recommenders project.
Schedule

• **v1.0** – 27.06.2012 (Juno Release)

• **v1.1** – 28.09.2012 (SR1)
  o Get Snipmatch ready,
  o Improve library coverage,
  o Improved recommendation models for better recommendation
  o New model serialization for faster model loading,
  o NLS support

• **v1.2** – 22.02.2013 (SR2)
  o Provide code search Infrastructure
  o Create tools to enable developers to build own models from inside Eclipse
Project Plan

• **Available at:**

• **Themes**
  - for Kepler will be defined after Juno.
  - Theme candidates are
    - snipmatch,
    - code search,
    - more intelligent recommenders
    - performance
    - (stacktrace search)