The Eclipse IMP: An IDE Meta-tooling Platform

Eclipse Technology Project
Proposal Creation Review
7 Nov, 2007

https://bugs.eclipse.org/bugs/show_bug.cgi?id=207597

IMP Project Motivation

- Eclipse offers:
  - first-class, extensible application platform
  - some excellent language-specific IDEs
- But: much less extensible as IDE platform
- Many existing programming languages do not yet enjoy a full-featured Eclipse IDE
- Related problems:
  - lack of full-featured IDE support for a language inhibits language adoption
  - lack of full-featured language support inhibits Eclipse adoption
- Prohibitively high cost of building Eclipse IDEs:
  - learning curve
  - lack of reusable components

IMP Project Goal

- Greatly reduce the complexity and cost of building Eclipse IDE support for new programming languages by:
  - Encapsulating common language/IDE idioms in framework components
  - Reducing burden of learning curve through frameworks and meta-tooling
  - Use of declarative languages to specify IDE appearance & behavior
  - Provide appropriate hooks to permit language-specific customization

IMP Project Scope

- No restrictions as to type of language
- Full-featured ("JDT-like") IDE functionality:
  - editor affordances:
    - syntax highlighting, folding, source formatting, quick outline, hyperlinking, content assist, "mark occurrences", etc.
  - other views:
    - single compilation-unit (e.g. outline) and project/workspace-level views (e.g. Package Explorer)
    - type hierarchy (for languages w/ inheritance), call graph views
  - preferences
  - incremental build support
  - indexed search
  - refactoring and underlying analyses
  - debugging

IMP Current Status

- Prototype w/ support for several IDE services
- Currently: used by several groups both inside and outside IBM
- Hosted on SourceForge since late August
  http://eclipse-imp.sourceforge.net
  - CVS
  - Eclipse Update Site
- Steady stream of traffic on IMP news group at Eclipse.org

Project Mentors

- Richard Gronback, Borland: GMF lead, co-lead Modeling PMC
- Ed Merks, IBM: EMF lead, co-lead Modeling PMC

IMP Community

- Many known potential users
- Current (active) collaborators:
  - Oxford University - Oege.de.Moor@comlab.ox.ac.uk, Torbjorn Ekman, Oleg Murk
  - Centrum voor Wiskunde en Informatica (CWI) - Jurgen.Vinju@cwi.nl, Paul.Klint@cwi.nl
  - IBM Rational - Chris Laffra
- Potential collaborators (have expressed significant interest, but have not yet begun actively working):
  - Borland, Inc. - Artem.Tikhomirov@borland.com
  - Eclipse JDT team - Martin Aeschlimann, Philippe Mulet
  - Eclipse DLTK team
  - Eclipse EMF team – Krzysztof Czarnecki, Ed Merks
  - Cornell University - Andrew Myers
  - Delft University - Eelco Visser
  - Eclipse CDT team
  - EPFL, Switzerland - Martin Odersky

Proposed components

- **Service Generators.** Generate stub/initial implementations for language-specific IDE services.
- **IMP Runtime.** Encapsulates common IDE behavior and appearance, with hooks for providing language-specific customizations.
- **Language Service Extension Points.** Encapsulate language-specific IDE service implementations (e.g. parsers, syntax highlighters, outliners, etc.).
- **Declarative Languages.** DSLs that specify language-specific service implementations. IMP will provide IDE support for the declarative languages themselves.
- **Parser Generator IDEs.** IDEs for existing parser generators, e.g. LPG, Antlr, or JavaCC.
- **Compiler/Parser Adapters.** Permits use of arbitrary scanning, parsing, AST & compiler technology.
- **AST-based Tools.** Includes languages for specifying AST patterns and transformations, search index creation, content assistance, "quick fixes" and AST-based search.
- **Static Analysis.** Reusable analysis components (type inference, pointer analysis, call-graphs, etc.) for tools relying on deep semantic knowledge.
  - AST-based analyses, similar to the JDT's type analysis infrastructure
  - Support for integrating with existing static analysis frameworks such as WALA.
- **Debug Framework.** Permits specification of debug-time presentation of the runtime platform (e.g. backtraces, variables, breakpoints, etc).
- **Process Documentation.** Cheat sheets & other on-line documentation
- **IDE-building Perspective.**

Community Response

- "Cool!" * 1000
- "When can we have it?" * 100
- "Wish this had been around N months ago!" * 10
- "Do you need to use LPG?" * 10
  - Answer: no!
- "Can this do something for the modeling community?" * 5
  - Answer: yes, models ~= grammars; we're talking to Ed Merks & K. Czarnecki
- "Will meta-tooling really work?" * 4
  - Answer: so far, it seems to work quite well (leverage, encapsulation)
  - Answer: we're adopting a more customizable, hands-on approach than previous meta-tooling environments (not merely "one-button" synthesis)
  - Answer: just ignore the meta-tooling and write code/extensions yourself
  - Answer: we've got a lot of exploring to do to find what works best (hence incubation-style Technology Project)
- "How does this relate to DLTK?" * 2
  - Answer: broader scope, but possible overlap in runtime components (TBD)
- "What about other kinds of IDE functionality?" * 2
  - Answer: we will explore how to provide support for new kinds of views, and new editor functionality
- "What about testing IMP-based IDEs?" * 2
  - Answer: we have nothing specific at this point, but this is a great area for community contributions
- "Why extension points instead of plain old inheritance?" * 1
  - Answer: inheritance still possible for implementation ease; meta-data natural way to bind services to languages; permit multiple contributors for services for which that makes sense (e.g. content assist), etc.

Relationship with Eclipse Projects

- JDT
  - work with JDT team to identify existing JDT components to be generalized and made part of the IMP, mine JDT for "best practices"
- LTK (Language ToolKit)
  - investigate IMP components to be moved into the LTK
  - such components can be used by IDE developers who don't wish to use IMP meta-tooling
- Web Tools Platform
  - Examine Structured Source Editor (SSE) as another potential alternative to current source editor in IMP runtime
- DLTK (Dynamic Languages ToolKit)
  - DLTK project focus is on dynamic languages, IMP has broader scope
  - investigate portions of DLTK framework to be reused or adapted
  - investigate using IMP as "front-end" to DLTK runtime framework
- EMF
  - investigate enhancing IMP to permit design/implementation of concrete textual language syntax from existing EMF models (based on notion that an EMF model can be viewed as a grammar)

Proposed Committers

- Robert M. Fuhrer - IBM Watson Research
- Philippe Charles - IBM Watson Research
- Stanley Sutton - IBM Watson Research
- Jurgen Vinju - CWI, Amsterdam
- Torbjorn Ekman - Oxford University, England

Committer Bio – Robert M. Fuhrer

- 20+ years' experience in software tool development and software/hardware development environments
- Ph.D. in Computer Science, Columbia Univ. 1999
- Spent last 5 years working on static analysis & refactoring for Java and Java-derived languages in Eclipse
- Worked closely with Eclipse JDT/UI team to incorporate advanced refactorings into Eclipse 3.1 & 3.2
- IMP project lead, principal architect and developer
  - Principal developer of IMP-based IDEs for LPG, X10 and a declarative language for specifying aspects of IMP IDE appearance & behavior
- Core team member in development of open-source compiler & runtime for X10 (http://x10.sourceforge.net)
- System administrator for IMP servers (web, update site, CVS, wiki, bugzilla, cruisecontrol)

Committer Bio – Philippe Charles

- 15 years' experience developing parser generators, compiler front ends, and associated tooling
- Architect and principal developer of open-source LPG parser generator (http://www.sourceforge.net/project/lpg) used by Eclipse JDT/Core
- Principal developer of Jikes open-source java compiler (http://jikes.sourceforge.net), part of many Linux distributions
- Core developer of X10 compiler (http://x10.sf.net)
- IMP core team member since inception, contributed to parser-related components, architecture, LPG IDE, X10 IDE

Committer Bio – Stan Sutton

- Ph. D. in Computer Science in 1990 from the Univ. of Colorado, Boulder, with a specialization in the area of software engineering.
- Career-long emphasis on software development environments, including object management, transaction management, software-process modeling, process-support environments, and aspect-oriented software development.
- Played active role in the Concern Manipulation Environment, an Eclipse Technology Project, on which he served as a committer.
- Core IMP project member since early 2006.
- IMP contributions:
  - development and maintenance of IMP runtime components, meta-tooling components, IDE service implementations, and elements of the IMP-based IDEs for LPG and X10.
  - Designed and implemented support for IDE preferences, including DSL for specifying preferences and preference UI, and associated IMP-based IDE.
  - Lead maintainer and lead author of IMP web site.
- Continuing interests: further applications of DSLs to IDE specification and generation, investigation of the extensibility of the IMP architecture, and applications of IMP to IT governance and of IT governance to IMP.

Committer Bio – Jurgen Vinju

- Research project leader at CWI in Amsterdam, The Netherlands
- Team member for over eight years, and now team lead for The Meta-Environment, an open-source meta programming framework based conceptually on formal language specifications. See http://www.meta-environment.org.
- Designed and developed the architecture of The Meta-Environment framework and many of its components, in close collaboration with many contributors.
- Specialist in source-to-source transformations, parser generation, disambiguation of context-free grammars, term rewriting and pretty printing.
- Active contributor to the Eclipse-IMP project; currently working on source formatting and source-to-source transformations

Committer Bio – Torbjorn Ekman

- Ph.D. from Department of Computer Science, Lund University in 2006. Thesis subject: Extensible Compiler Construction, which is highly relevant to IMP's goals.
- Current work extends those results to extensible development environments based on Eclipse, and in particular to composable refactorings, also highly synergistic with IMP.
- Research builds on the meta-compiler system JastAdd (http://jastadd.org) as an experimental platform.
- Designed and implemented the JastAdd Extensible Java Compiler, a declarative object-oriented framework used to implement a complete Java 1.4 compiler with Java 5 as modular language extensions.
- Resulting compiler passes 99% of Jacks/Mauve Java test suite, slightly more than the Eclipse JDT java compiler and significantly more than Polyglot.
- Currently overseeing effort to adapt JastAdd to IMP.

Code Contributions

- IBM Research is offering to contribute its Eclipse IMP software as the initial codebase
  - previously released as open-source at

    http://eclipse-imp.sourceforge.net
- Components
  - IMP Runtime
  - Support for several language services:
    - syntax highlighting, parsing, outlining, folding, hover help, hyperlinking, incremental building
  - Meta-tooling
  - Cheat sheets

Tentative Near-term Plan

- Initial Eclipse.org presence in Nov./Dec. 2007
  - website, newsgroup, CVS repository seeded from SourceForge contents (if IBM source code contributions are accepted)
  - Bugzilla repository, seeded with bug reports from current non-public Bugzilla repository (if IBM's code contributions are accepted)
- Initial Eclipse.org release, v1.0: Mar. 2008, including:
  - IMP Runtime library
  - Parser-generator-neutral APIs
  - Reworked (standard Eclipse) feature/site build/deployment process
  - Initial meta-tooling support for scanning, parsing, syntax highlighting, outlining, source folding, hover help, hyperlinking, content assist, incremental building
  - Meta-tooling and IDE support for LPG-based grammar and scanner specifications
  - Exemplar IMP-based IDEs for X10, the LPG grammar specification language, Java
  - Cheat sheets guiding/describing IDE development process
  - Initial installation and use documentation, taken from existing (non-public) web pages
  - Possibly remove dependence on JDT Package Explorer
- Second Eclipse.org release, v1.1: Oct. 2008, including some of:
  - User-customizable source code formatting
  - Support for language embedding
  - Integration with the JAstAdd compiler framework
  - Integration with other parser generators such as ANTLR and/or JavaCup
  - Improved support in meta-tooling, framework classes and code templates for language inheritance

Future Directions

- Refine existing declarative specification languages, explore similar support for other language services
  - reduce burden of writing boilerplate Java code and understanding IMP/Eclipse APIs
- Explore tooling to help in implementing new static program analyses
- Explore frameworks for cross-language analysis & refactoring
- Explore support for building views not present in existing IDEs
  - e.g. meta-tooling to assist in building views from analysis queries (a la JQuery, but not targeted at end users)