

Experience Report:
Building an Eclipse-based IDE for Haskell

Leif Frenzel

himself@leiffrenzel.de

ICFP 2007, October 1-3, 2007, Freiburg, Germany

Where the experience comes from:

The EclipseFP project

- An Open Source Project at SourceForge
- All volunteer work
- Founded 2004 by Leif Frenzel
- Since 2006 maintained by Thiago Arrais
- 5 Developers, 49 mailing list subscribers
- 10 preview releases so far (version 0.1 to 0.10)

Where the experience comes from:

The EclipseFP project

- Full support for Haskell development in Eclipse - comparable to the Eclipse Java IDE (JDT).
- Builds on generic IDE Platform (Eclipse Platform)
 - Get language-neutral functionality for free.
- Don't re-implement existing Haskell tools – integrate!
 - Get language-specific functionality for free.
 - Users already familiar with these tools.

Getting things for free (I) – language-neutral functionality

The screenshot shows the Eclipse IDE interface with several annotations in orange text and arrows pointing to specific features:

- text search**: Points to the search icon in the toolbar.
- plain-text editing**: Points to the text editor window titled "text-file.txt".
- pluggable RCS support (here: Darcs)**: Points to the "Darcs Reposit..." icon in the top right of the IDE.
- files organized in projects**: Points to the Project Explorer on the left, which shows a tree structure of files and folders.
- task items**: Points to the "Tasks" tab in the bottom right, which contains a table with columns for Description, Resource, Path, and Location.

The text editor window displays the following content:

```
1 Plain text file editing functionality is included
2 with the generic IDE Platform ("Eclipse Platform").
3
4 Provides many text-editor features, such as
5
6 - Copy/Cut/Paste
7 - Line numbers
8 - Search-as-you-type
9 - Spell checking
10 - etc.
```

✓	!	Description	Resource	Path	Location

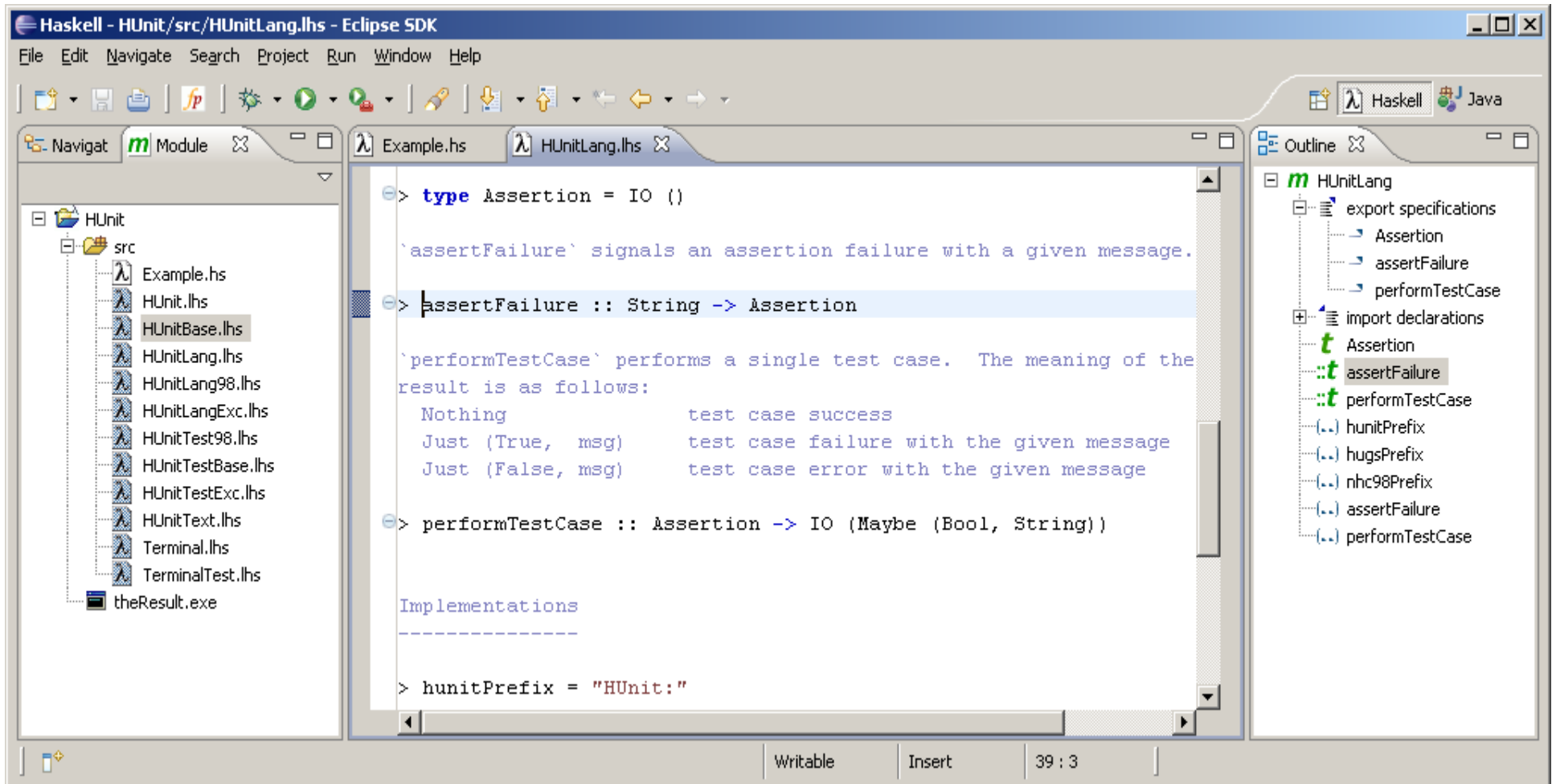
At the bottom of the IDE, the status bar shows "Writable", "Insert", and "7:15".

Getting things for free (II) – language-specific functionality

- Tools and libraries from the Haskell community:
 - Haskell parser (language-src)
 - Type checker and type inference engine (GHC API)
 - GHCi debugger
 - Haskell refactorer (HaRe)
 - Haskell API search (Hoogole)
 - Build infrastructure (Cabal)
 - Documentation generator (Haddock)
 - Unit test runner (HUnit, QuickCheck)
 - ... and more

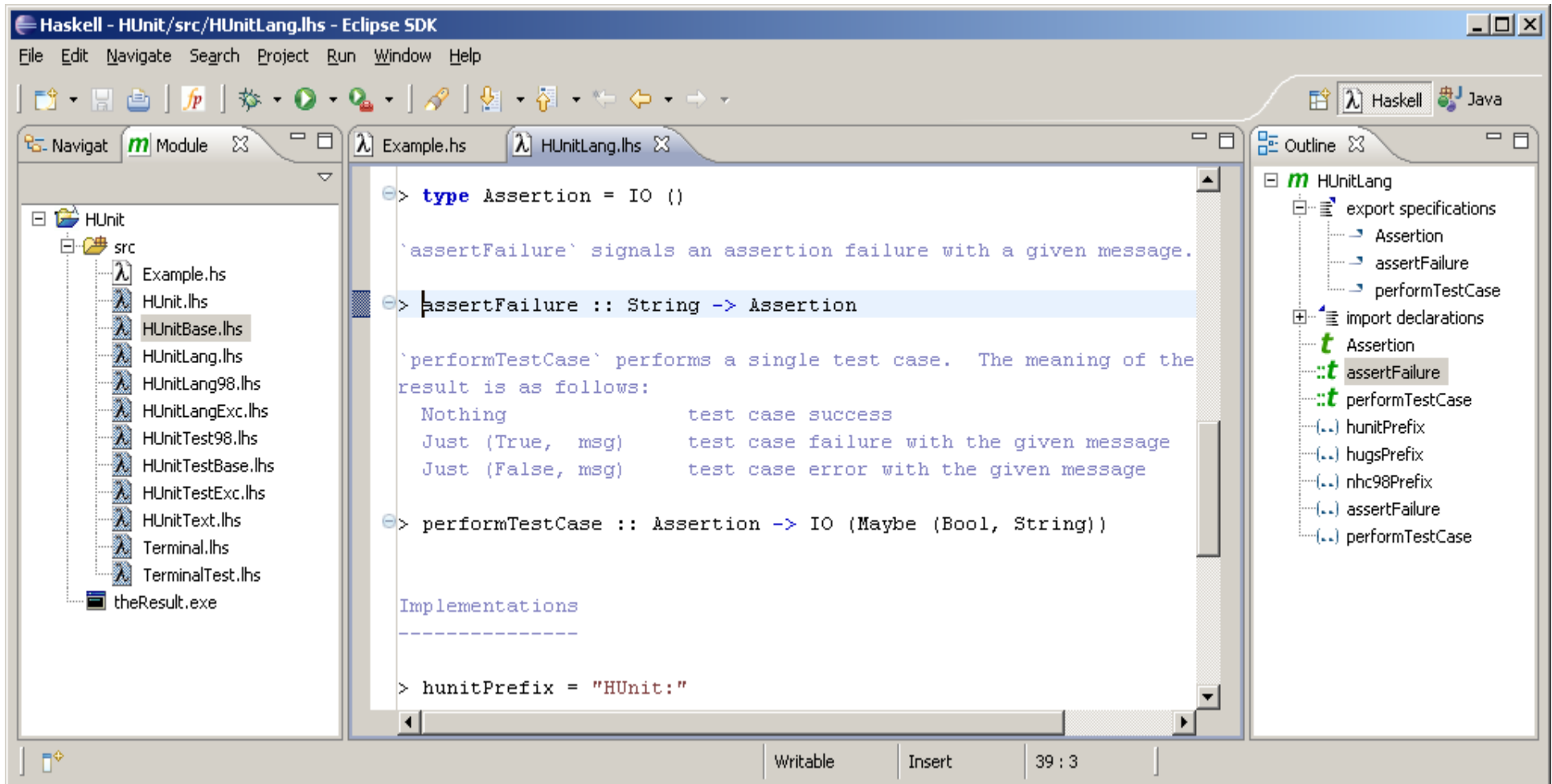
- Eclipse extensions must be written in Java.
- The Eclipse runtime architecture is aggressively dynamic and modular.

Example: Outline View functionality



- for overview and navigation of the source file
- connects to the active editor
- shows top-level elements from the source

Example: Outline View functionality



- parse the editor content (not the file content)
- build a list of elements to display
- include positions of the elements in the source code

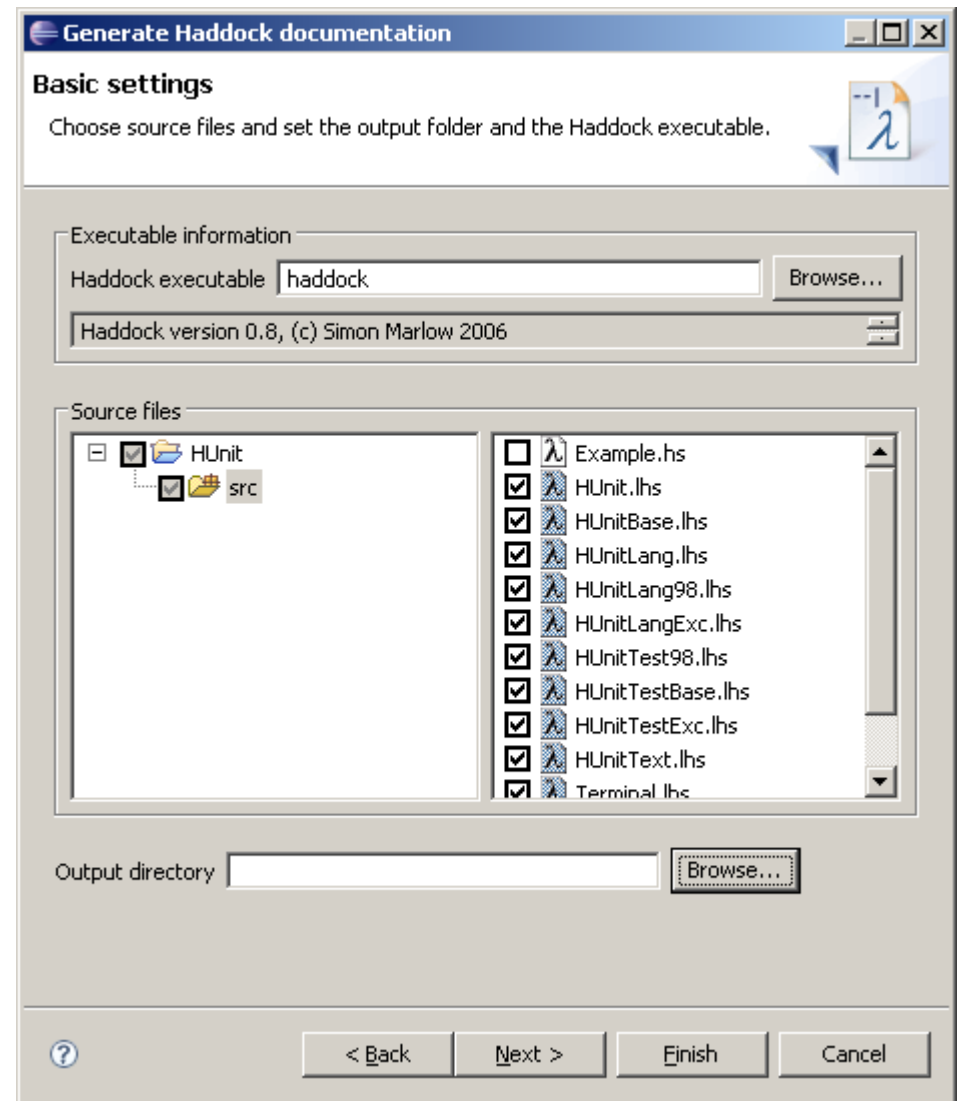
Option I: Call as external program

- write a Haskell mini-program and link into an executable
- call from an Eclipse extension (i.e. from Java code) in a separate process

Option I: Call as external program

Makes sense in some cases:
Docs generation via Haddock

- show a wizard
- collect settings
- build a command line and run Haddock
- make generated HTML docs visible in the project tree



Option II: Go via native interfaces

- put Haskell code in a dynamically linked library (included with the Eclipse extension)
- export Haskell functions via FFI and import them on the Java side via JNI (with additional bridging code written in C)
- re-use `haskell-src` library and write additional logic in Haskell
- works, but is not really satisfying:
 - didn't manage to build suitable dynamically linked lib for Linux (no PIC support in GHC)
 - stability issues: problems in the native code always crashed the JVM
 - implementation via JNI is complicated (and ugly)

Option III: Parser implementation in Java

- write a grammar for a parser generator that generates Java code (ANTLR)
- walk the AST, collect information etc., all in Java
- integration is much easier, but there's no future:
 - ignores existing Haskell tools entirely and re-invents
 - How could we convince potential contributors?

Solution: The Cohatoc approach

- spread Haskell code (compiled object code or source code) over Eclipse extensions
- register it (similar to Eclipse extensions written in Java)
- there is an additional support layer on top of the Eclipse runtime that
 - knows the structure of the Eclipse installation
 - can locate and prepare the registered Haskell code
 - maintains a separate process ('function evaluation server') for the execution of native code
 - dynamically loads (if necessary, compiles) Haskell code using the hs-plugins library
 - provides utility functionality to support marshalling data

Summary

Building an IDE is a huge effort – build on a generic platform and concentrate on integrating existing functionality.

To integrate Haskell code in a dynamic and modular host platform, we had to find a way to modularize executable Haskell code (hs-plugins).

Eclipse extensions can now be written partly in Haskell (language-sensitive logic) and partly in Java (UI and IDE integration) – and it feels natural on both sides.

Thanks!

EclipseFP homepage

<http://eclipsefp.sourceforge.net/>