Escaping the Java Trap (05 June 2005)

Written for SouJava during FISL 6.0 by Bruno Souza, Dalibor Topic, David Wheeler and Mark Wielaard

For the last couple of years the community has been working to ensure that developers can create applications using the java programming language without having to depend on proprietary software, and we have already succeeded for many developers. Unfortunately, many people don't understand how successful we've been, and where we're going; this document gives a road map showing how the various projects are working together, where they are, where they're going, and how we make sure that they work well.

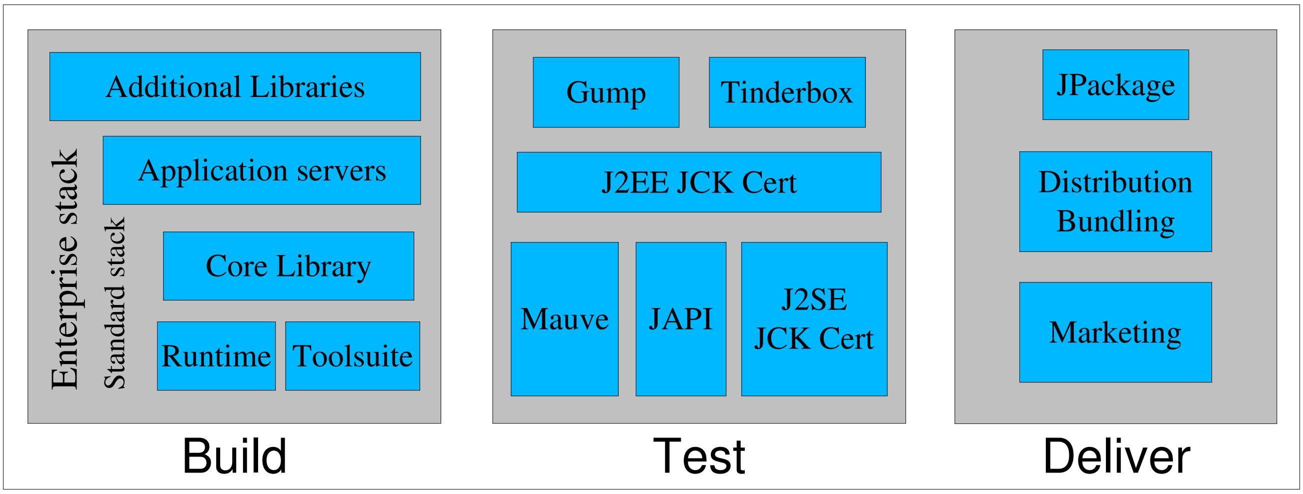
Today, the free implementations are already very capable; they can run important large applications like OpenOffice.org 2, Eclipse 3, Tomcat 5, and implement a vast amount of functionality that developers expect from a java-like environment.

Our strategy has been "user-driven development": we prioritize developing the capabilities most needed by real applications. In some areas we do "collaborative competition": we have multiple projects that share ideas and sometimes code so people can choose the best implementation for their purposes. To promote adoption and participation in the free implementations core components are distributed under terms that allow you to develop and distribute software under any license, though we hope you'll decide to release your programs as free, libre, open source software (FLOSS).

We recommend that developers use the FLOSS implementations right now — they are ready for use, and you'll avoid accidentally using a capability not currently implemented in them. Use 1.4 or lower features, not 1.5. GUI developers must choose a GUI toolkit; for now, they should avoid Swing (unless they wish to help complete our implementation) and choose an alternative toolkit: AWT for portability, java-gnome for GNOME integration, or Eclipse SWT for Eclipse integration.

Currently the primary challenge is remaining unimplemented facilities in the core library; we urge developers to help in this area.

The picture below shows how the community builds, tests, and delivers the full free java-like stack.



Build

Component	*Current Status*	*Future*
Toolsuite and runtime: Includes the tools (e.g., compiler, jar creator and appletviewer) to develop and deploy applications. The core component for running applications written in java is the runtime.	Many implementations. GCJ can compile to fast (production quality) machine code, but as part of the critically important GCC suite, releases are not as frequent to allow more time for testing. Kaffe is a more traditional (interpreter/jit) implementation that is developed and released more rapidly; IKVM runs applications on Mono. There are many other implementations where innovations occur and actively compete with them; their best features get merged into Kaffe, and later GCJ. Gcjwebplugin provides a (pluggable) appletviewer and GNU Classpath Tools provides various additional tools. Native GCJ applications use GDB for debugging.	Implementations are working to complete 1.5 features. GCJX is the next generation GCC frontend that will replace the current GCJ and will add 1.5 language features. Jarsigner, key management and corba tools are in development. Recently the Apache Harmony project was announced, it is researching a pluggable component model for runtimes using existing parts of the existing free stack already available. GNU Classpath will add generic JDWP debugging support.
Core Libraries: The essential core libraries required to compile and run applications.	GNU Classpath is the shared library that includes most 1.3 and 1.4 features, including essentially all of the core java.* and javax.* packages; it currently lacks full support for Swing, CORBA and some of the newer 1.5 packages.	Enhanced AWT integration with GTK+, cairo and pango, add full Swing support, accessibility, JNDI providers, CORBA and kerberos. Where possible import existing libraries to provide additional core packages. Classes that use the new 1.5 language capabilities are developed on a separate branch.
Application Servers: Extended set of enterprise libraries and programs.	Jonas and JBoss (both LGPL) both provide J2EE certified application servers.	Get Jonas working on GCJ. Apache Geronimo is being finalized to be J2EE certified.
Extra libraries and languages: java-gnome, Apache libraries (Jakarta), ObjectWeb, SouJava extras, and many more.	Java-gnome enables access GNOME features. java-gnome is also ported to MS Windows. GNU KAWA provides Scheme, Common Lisp, and other scripting support. Other languages are provided by Jython, JRuby and Rhino (javascript).	SouJava working on libraries for mobility, etc. Jakarta libraries keep expanding.

- GCJ: http://gcc.gnu.org/java/
- Kaffe: http://www.kaffe.org/
- IKVM: http://www.ikvm.net/
- Overview of runtimes: http://www.gnu.org/software/classpath/stories.html
- GNU Classpath: http://www.gnu.org/software/classpath/
- GNU Classpath Tools: http://www.gnu.org/software/classpath/cp-tools/
- GCJWebPlugin: http://www.nongnu.org/gcjwebplugin/
- Jonas: http://jonas.objectweb.org/
- Jboss: http://www.jboss.org/
- java-gnome: http://java-gnome.sf.net/
- Jakarta: http://jakarta.apache.org/
- GNU Kawa: http://www.gnu.org/software/kawa/

Test

Component	***Current Status***	***Future***
Mauve: Tests core library, JVM, compiler.	Over 28,000 tests of the core library, plus Jacks (which tests the compiler specification) and AWT interactive test suite.	Improve bytecode verification tests, add automated GUI tests, add Swing tests (inc. from NetBeans)
JAPI: Determines % symbols complete.	Works well, graphically displaying which facilities are available.	Extend to cover 1.5 (e.g. Enumerations, generics).
Gump: Integration stack testing.	Continuously builds applications on permutations of full stack to detect any regressions, including Kaffe.	Add gcj.
Tinderbox: Platform portability testing.	Continuously builds on different platforms to make sure applications can run anywhere.	Add gcj, Mauve testing as part of Tinderbox testing.
Major application testing.	OpenOffice.org 2, Eclipse, and TomCat running today. Jonas close to completion (demonstrated).	Azureus, More Jakarta & Geronimo to work on Free implementation. Long-term: Netbeans (requires Swing).
J2SE and J2EE JCK Certification.	• SouJava full Java community Process (JCP) member, and will submit stacks for certification. • Apache Foundation JCP member and on executive committee; submit stacks for Apache, and vote on specifications.	• SouJava to submit stacks for certification; first Kaffe plus GNU Classpath (javali and roxo projects). • Apache Foundation to submit Apache Harmony for certification.

- Mauve: http://www.sourceware.org/mauve/
- JAPI: http://www.kaffe.org/~stuart/japi/
- Gump: http://gump.apache.org/
- Tinderbox: http://tinderbox.anholt.net/tinderbox3/
- SouJava Javali project: http://javali.soujava.org.br/

Deliver

Component	***Current Status***	***Future***
JPackage	Packaged over 1500 Java FLOSS packages as rpm.	Support more systems (.deb and source based); unify deployment approach on Fedora, Debian and Gentoo.
Distributions and Bundles: Distribution and packaging of the full free enterprise stacks.	Red Hat Fedora Core 4 includes GCJ 4, GNU Classpath, Eclipse, OOo2, Tomcat 5, plus many of the extra libraries. Debian Sarge includes Kaffe, GCJ (3.4), and others. FreeBSD, NetBSD and OpenBSD support Kaffe but don't use GCC 4 yet.	Debian/Ubuntu: GCC 4 migration, then OOo2 and Eclipse. FreeBSD, NetBSD: GCC 4. Kaffe will be used as a bridge where GCC 4 is not currently available.
Marketing: Explaining how it all works together	SouJava released this document.	Core developers will give presentations at international conferences. Apache Harmony will create awareness.

- JPackage: http://www.jpackage.org/
- Distributions: http://java.debian.net/, http://www.ubuntulinux.org/wiki/JavaIntegration, https://www.redhat.com/archives/fedora-devel-java-list/, http://www.gentoo-wiki.com/Java.
- GNU Classpath events: http://www.gnu.org/software/classpath/events/