jddd migration to OpenJDK11+: benefits and pitfalls

E. Sombrowski, K. Rehlich, G. Schlesselmann, DESY, Hamburg, Germany

Abstract

The Java Doocs Data Display (jddd) [1,2] is a Java-based tool for creating and running graphical user interfaces for accelerator control systems. It is the standard graphical user interface for operating the European XFEL accelerator. Since Java8 Oracle introduced a number of major changes in the Java ecosystem's legal and technical contexts that significantly impact Java developers and users. The most impactful changes for our software were the removal of Java Web Start, Oracles new licensing model and shorter release cycles. To keep jddd up to date, the source code had to be refactored and new distribution concepts for the different operating systems had to be developed. In this paper the benefits and pitfalls of the jddd migration from Oracle Java8 to OpenJDK11+ will be described.

Changes in Java ecosystem since Java8

Oracles new licensing model

Starting with Java 11 Oracle offers two distinct Java releases with different license models:

- Oracle JDK under commercial OTN License Agreement for Java SE [3]: with Long Term Support (LTS), but only free of charge for development and tests.
- Oracle OpenJDK under the open source GNU General Public License v2 with Classpath Exception (GPLv2+CPE): free, but no LTS.

New Oracle Java release cycles

Oracle changed from a feature-based to a time-based release cycle:

- Major releases for Oracle JDK and OpenJDK every 6 months, which are only supported until the next release.
- Commercial Oracle JDK LTS releases every 3 years: The most recent LTS release is Java 11, which came out in September 2018 and will be supported until 2026.

Removal of Java Web Start

Java Web Start is not included any more in Java SE11 and later [4].

Our way from Java 8 to OpenJDK 11+

Choosing a Java distribution

Oracle has been and will stay the reference implementation for Java . We have decided to use Oracle OpenJDK, because the license model fits our needs and environment best.

Dealing with shorter release cycles

We have started the jddd migration from Java 8 to OpenJDK11+ in September 2018 with OpenJDK 11, which turned out to be quite easy. No major code changes had to be done. Since OpenJDK11 jddd application are always compiled with the latest OpenJDK release without any problems.

Replacement of Java Web Start

Java Web Start was an easy and uncomplicated way to distribute a completely configured application to everyone's desktop independent of the operating system. As Java Web Start is not included in OpenJDK11+ any more, another deployment technology had to be found.

Distributing only the jddd.jar to the users is not an option, because it requires a standalone JRE to be installed and maintained on every user desktop.

Using jlink for the creation of a custom Java runtime image is also not possible, because jddd depends on many old non-modularized libraries.

Our solution is to deploy jddd packages including all jddd jars plus the appropriate JDK. Four different packages are needed at DESY (see screenshots below):

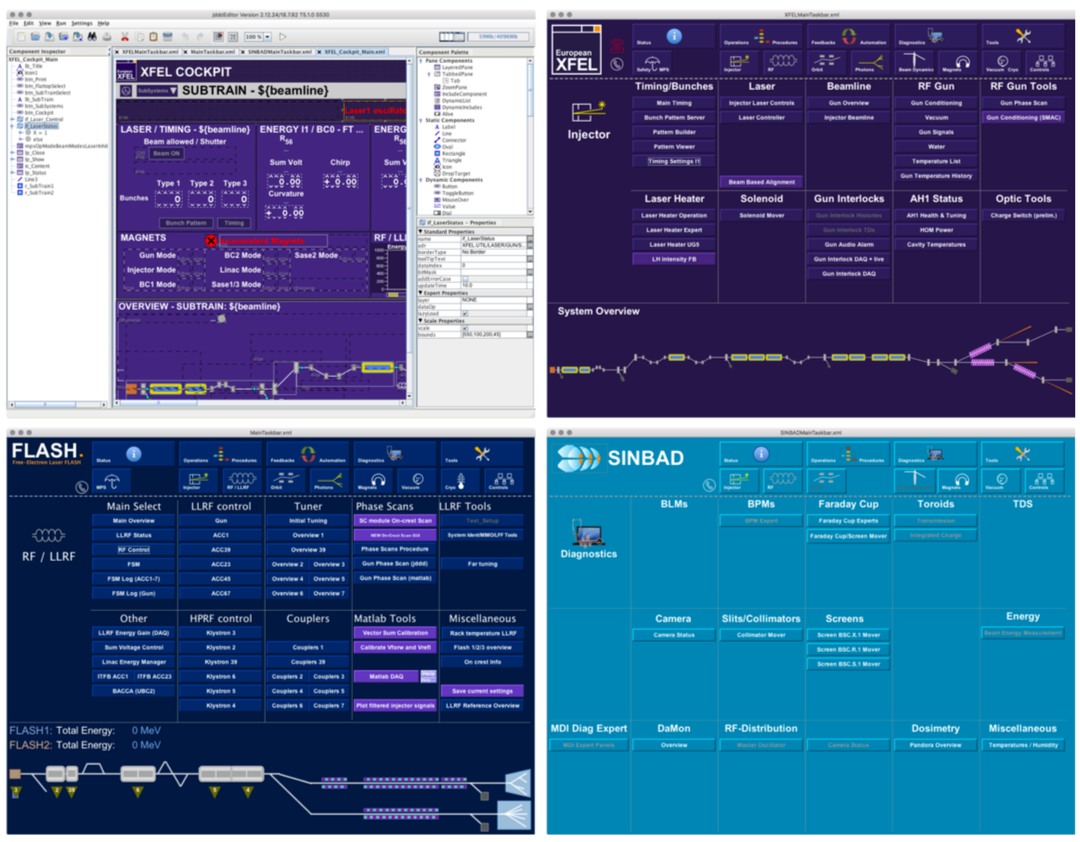
- The jddd editor
- XFEL MainTaskbar
- FLASH MainTaskbar
- SINBAD MainTaskbar

Starting old Java applications from jddd

The screenshots on this poster shows the jddd based control windows for XFEL, FLASH and SINBAD (called MainTaskbar), which are used as entry point to start all needed graphical user interfaces (GUIs). These are mainly jddd panels, but can also be Matlab, Python or other Java applications.
Because it would be too much effort to upgrade all old Java applications, we found a simple solution to start these GUIs without any code and JNLP start script modification: We added jaws, a Java Web Start replacement by Cosylab [5], to our jddd packages. The jddd application now internally replaces all javaws calls by jaws. Because the Java Virtual Machine is backward compatible and can run older bytecode, most old Java applications (except JavaFX applications) can be started with the current OpenJDK from the jddd package.

Using jaws is a suitable intermediate solution until the transition from Java 8 to OpenJDK11+ will be finished for all applications.

Screenshots of the jddd Editor (top left) and XFEL, FLASH and SINBAD MainTaskbar



Upgrade of the messaging system – from JMS to MQTT

The jddd software utilizes a messaging system for:

- collecting runtime statistics
- evaluating failures/exceptions
- sending panel update notifications within running jddd applications at DESY

With the Java upgrade we decided to switch the messaging system from JMS to the more popular MQTT protocol. The Eclipse Paho Java Client library [6] provides an open-source implementation of the MQTT messaging protocol. It is well documented and fits our needs perfectly. As broker we decided to use the open-source message broker Eclipse Mosquitto [7] because it is lightweight and available as Debian package.

Upgrade of the jddd Web Interface

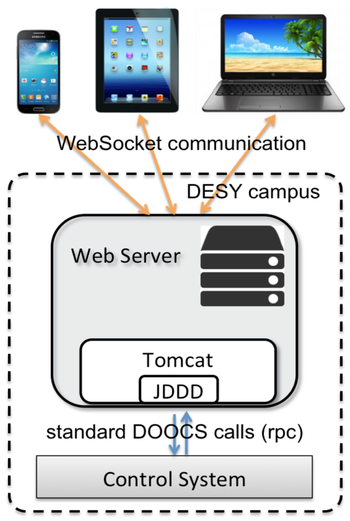
For remote monitoring and expert assistance an HTML5 version of jddd, running in a Tomcat application server, is available [8].

Functionality

On the server side the JDDD application is running in a Tomcat web server. All panels are started in headless mode. A buffered image of each panel is created with an update rate of 0.5 Hz.
For client /server communication the WebSocket protocol is used. Images are sent from the server to the client, mouse events are sent in the opposite direction.

Upgrade

The web interface has been successfully tested with OpenJDK11+ and Tomcat 9. No major problems have been observed. The upgrade is scheduled for the end of this year.



Conclusion

For the Java update of our jddd software, the following decisions were made:

- We use Oracle OpenJDK11+.
- We stick to Oracles short release cycles and compile and distribute jddd always with the current OpenJDK version.
- We deploy jddd packages including all jars, the complete JDK and the jaws software for starting old Java applications via jddd.

The migration from Java 8 to OpenJDK11+ was successful. Unfortunately it has not been possible to switch completely to the new Java module system, due to many old libraries. The plan is to replace these old libraries to be able to use jlink in future.
The concept of jddd packages containing the current JDK has proven itself. Even though the packaging and deployment procedure is more complicated now compared to Java Web Start before, the benefit is that jddd no longer depends on the appropriate preinstalled Java version.

References

[1] doocs/jddd website: https://doocs-web.desy.de
[2] E. Sombrowski. A. Petrosyan, K. Rehlich, W. Schütte, "jddd: a tool for operators and experts to design control system panels", ICALEPCS'13, San Francisco, USA, October 2013.
[3] https://www.oracle.com/downloads/licenses/javase-license1.html
[4] Oracles Java client roadmap: https://www.oracle.com/technetwork/java/javase/javaclientroadmapupdate2018mar-4414431.pdf
[5] COSYLAB website: https://www.cosylab.com
[6] Eclipse Paho MQTT Java library at GitHub: https://github.com/eclipse/paho.mqtt.java
[7] Eclipse Mosquitto MQTT broker homepage: https://mosquitto.org
[8] E. Sombrowski. R. Kammering, K. Rehlich, "A Html5 Web Interface for Java Doocs Data Display", ICALEPCS'15, Melbourne, Australia, October 2015.