Vision on the future of Java-OpenDA and Python-OpenDA.

This document describes the vision of the OpenDA association on the future of the Java- and Python-based OpenDA. We publish this vision so that both users and developers of OpenDA can take this into account for future developments using OpenDA. We begin with a brief pointwise statement of the vision, followed by a more detailed explanation of the rationale for this vision.

1. Current operational systems should not be affected by any changes proposed to OpenDA.
2. The Java version of OpenDA will remain the main version for operational use for the foreseeable future. New developments, outside of those required for operational systems, are not expected.
3. We will use the new experimental Python version of OpenDA (pyOpenDA) for new developments and research.
4. Once pyOpenDA becomes stable, performant, and suitable for operational use, we will work on a smooth transition to pyOpenDA for operational systems.
5. There is no deadline for this transition, as OpenDA developments are dependent on projects for development and there is no funding available for core development of OpenDA in isolation from usage.
6. To assist in the long-term maintenance of OpenDA, a number of classes will be deprecated (by clearly marking these @deprecated in the Java code) and eventually removed from the Java-based OpenDA once they are no longer used by any operational system.
7. This vision will be revised on a yearly basis.

Rationale

OpenDA is used in many operational systems, where the stability of the implementation is very important. For this, the Java-based OpenDA implementation is very suitable. The core data structures and building blocks are very stable and hardly change. Most recent developments of the Java version focus on coupling new models to the framework.

Unfortunately, Java is not a widely-used programming language in academia. It is rarely used in teaching, nor do many researchers use it. On the other hand, Python is a popular language for research, especially for toolboxes such as OpenDA that interface to many existing models written in different programming languages. There is an experimental version of OpenDA implemented in Python (pyOpenDA). Going forward, pyOpenDA will be the primary version used for research and other experimental work on OpenDA.

Having a separate codebase from the operational Java code allows us to experiment with changes to OpenDA, for instance adding new classes of algorithms to the framework for machine-learning techniques, without disturbing the operational work done with the Java-based version of OpenDA. For the time being, we advise not to use pyOpenDA for

operational systems since new developments will cause existing applications of pyOpenDA to break from time to time, requiring perhaps significant effort to update.

The current Java version will be maintained and will remain the main version for operational purposes for the foreseeable future. Bugs and performance issues will be resolved, as well as limited new features when required for an operations system.

Once pyOpenDA stabilizes to the point that it could potentially be a replacement for the Java version of OpenDA, we will start creating stable releases of it. From that moment, we will work on a smooth migration for operational systems currently based on the Java-based version of OpenDA to pyOpenDA. As funding for pyOpenDA is largely dependent on projects, there is no deadline for creating the stable pyOpenDA, nor the transition to it for operational systems.

As a result of all the developments over the years, Java-based OpenDA has collected a fair collection of implementations of more or less equivalent functionality. Some performance-critical parts of OpenDA were initially written in C. This used to be beneficial for performance, but it has been obsolete after performance improvements to Java. In general, we see that new models are mostly coupled to OpenDA using the black-box coupling, with a number of other more specialized couplings based on ZeroMQ and sockets. To assist in the long-term maintenance of OpenDA, we will begin marking some of the older implementations as deprecated and will remove these once no operational systems make use of them.