Planning Releases of The ASF+SDF Meta-Environment

Paul Klint
Jurgen Vinju

2010-02-08 09:17:12 +0100 (Mon, 08 Feb 2010)

Table of Contents

Introduction and discussion ................................................................. 1
   Risks ................................................................................................. 2
   Efficiency ......................................................................................... 2
   Deliverables .................................................................................... 2
   Versions .......................................................................................... 3
Planning ................................................................. 3
   ASF+SDF Meta-Environment 2.0 - August 2008 .................................. 4
   ASF+SDF Meta-Environment Servicing Stage (starting 2009) ............ 4

This document is being written at this time: work in progress.

Introduction and discussion

Since The Meta-Environment is a product family, of which also sub-components are released, this document describes the planning for components as well as products in The Meta-Environment. Work on The Meta-Environment can be categorized:

• Programming - Implementing (new) features
• Refactoring - Improving non-functional quality attributes
• Fixing Bugs - Fixing known issues that are registered the Bugzilla system
• Documenting - Producing user-documentation and other documents
• Building - Work on the build and configuration environment
• Deployment - Work on continuous integration and deployment, this includes manual testing

The work on The Meta-Environment, and thus its releases, are driven mostly by research. Deadlines for papers about components of The Meta-Environment, or using components of The Meta-Environment, provide the basic structure for this planning. These deadlines may be self-imposed or provided from outside. In any case, the above categories of work organize naturally around these kinds of deadlines. For example: when writing a research paper there is no time for Refactoring, other kinds of Documentation, the Build environment, or Deployment. The focus is then simply on Programming and Fixing Bugs. On the other hand, Refactoring and Documenting are done during explorative research and Friday afternoons, when time seems more abundant.

Since research deadlines are induced on a personal level (of the researchers/programmers), they can be very dynamic (read irregular). One could say that this makes planning impossible, but we still want to get things done. Therefore, the contents of this planning are the responsibility of all programmers: please check it, execute it, adapt it.
Risks

There are two risks that this document intends to mitigate:

• There will never be a release, because research always gets higher priority than the software. The effect is a large disinvestment in software that is written, debugged and maintained but never used outside of the development team. This will also harm users of the software that either use The Meta-Environment for their teaching, research or commercial activities.

• Research deadlines are not met, because the software prohibits easy experimentation. This may be caused by badly maintained, buggy or undocumented software. The effect is missed deadlines, and a possible disinvestment in a laboratory designed for experimental research in language analysis and transformation that can not be used for experimental research.

These two risks influence each other. I think it is safe to assume that software quality generally decreases when working on research deadlines, and it generally improves when working towards releasing the software. Still, for doing good research, you need good software, and for good software you need good research. Finding the proper balance is a dynamic process that needs constant monitoring and assessment.

Efficiency

To walk the line, we have invested and keep investing in making our software process more efficient. Standardization on the ATerm data-type is one trick that worked. Code generation has also been a successful method in increasing efficiency. Another success is the continous integration and release system Sisyphus.

However, the most important method to make releases of The Meta-Environment possible is version management. Please read this document: version management. By keeping low code quality experiments on separate branches we facilitate experimentation without jeopardizing the quality of the trunk.

Deliverables

We deliver the following products, which are all compositions of one or several of our components, including the respective documentation:

• ATerm library in C
• ATerm library in Java
• ApiGen code generator
• SDF ApiGen: SDF front-end for ApiGen
• JJTraveler: tree traversal library for Java
• ToolBus in C, including adapters
• ToolBus in Java, including adapters
• SDF: parser generator and parser libraries and tools, including the sdf-library of grammars for programming languages
• Asc-support: the run-time environment for compiled ASF+SDF specifications
• ASF+SDF Meta-Environment: the programming environment for ASF+SDF
• Sisyphus continuous integration system
Know that all releases of The Meta-Environment are accompanied by at least a full source code distribution. Binary distributions are complementary, and based on the availability of platforms to generate the distributions on. A delivered source code distribution has:

- a tested build environment (configure script and Makefiles)
- the full source code of the product
- documentation online at www.meta-environment.org
- and a link to the tarball on the www.meta-environment.org

**Versions**

Each released product has a human readable version number: `<major>.<minor>.<patchlevel>`. These numbers are maintained with the top-level component of each product. This implies that the version of the top-level component identifies the version number of the product, and that each product must have a single top-level component.

Release Candidates are releases of products that are incomplete, but testable and possibly usable by end-users. The intention is to get early feedback from a less prejudices audience than ourselves. The version scheme is `<major>.<minor>-RC<candidate-number>`.

Continuous releases (See Sisyphus) have generated version numbers based on the Subversion repository version and the Sisyphus Build identifier.

**Planning**

Each release has a section, in order of planned release date. Please remember that we do not promise to do any of these tasks. This is just our planning. It may change.
ASF+SDF Meta-Environment 2.0 - August 2008

Table 1.1. Planning the work of the Meta-Environment

<table>
<thead>
<tr>
<th>Activity</th>
<th>Category</th>
<th>Team</th>
<th>Deadline</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>First class layout supported by ASF compiler</td>
<td>Programming</td>
<td>Jurgen</td>
<td></td>
<td>done</td>
</tr>
<tr>
<td>Fix race condition in GUI</td>
<td>Programming</td>
<td>Jurgen</td>
<td>August 2008</td>
<td>In progress</td>
</tr>
<tr>
<td>Fix documentation Sisyphus</td>
<td>Documenting</td>
<td>Tijs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross linking documentation</td>
<td>Documenting</td>
<td>Paul</td>
<td>August 2008</td>
<td></td>
</tr>
<tr>
<td>Meta on rails documentation</td>
<td>Documenting</td>
<td>Paul</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document meta.actions</td>
<td>Documenting</td>
<td>Jurgen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>Deployment</td>
<td>Everybody</td>
<td>August 2008</td>
<td></td>
</tr>
<tr>
<td>Documentation commandline tools SDF</td>
<td>Documenting</td>
<td>Jurgen, Paul, Tijs</td>
<td>August 2008</td>
<td></td>
</tr>
<tr>
<td>Documentation commandline tools ASF</td>
<td>Documenting</td>
<td>Jurgen, Paul, Tijs</td>
<td>August 2008</td>
<td></td>
</tr>
<tr>
<td>Release 2.0</td>
<td>Project</td>
<td>Jurgen, Tijs</td>
<td>August 2008</td>
<td></td>
</tr>
</tbody>
</table>

ASF+SDF Meta-Environment Servicing Stage (starting 2009)

In 2009 development started on the Rascal DSL, which put the ASF+SDF Meta-Environment in servicing stage due to manpower constraints. Only a limited amount of bugs will be fixed. See also http://blog.meta-environment.org

Table 1.2. Planning the work of the Meta-Environment

<table>
<thead>
<tr>
<th>Activity</th>
<th>Category</th>
<th>Team</th>
<th>Deadline</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNGLR merged into the trunk</td>
<td>Refactoring</td>
<td>Rob</td>
<td>Postponed</td>
<td></td>
</tr>
<tr>
<td>Migrate to Eclipse IMP platform</td>
<td>Programming</td>
<td>Jurgen, Magiel, Tijs, Arnold, Bas</td>
<td>Postponed</td>
<td></td>
</tr>
<tr>
<td>Setup Eclipse Update Site</td>
<td>Building</td>
<td>Tijs, Jurgen</td>
<td>Postponed</td>
<td></td>
</tr>
</tbody>
</table>