

## Plan to Prepare ALF for Validation Review

### Background

#### Feedback from Eclipse

While the ALF project was guided by the documented criteria for exiting the Validation Phase – and it met those criteria – there were additional, less explicitly described criteria and processes considerations that had been overlooked or under emphasized. When the Eclipse community brought these to the ALF project's attention, the ALF committers decided to postpone the Validation review and address the concerns raised.

Some of these concerns were due to a difference in emphasis between a traditional IDE plug-in based Eclipse project and the more server-centric focus of ALF. Applying the expectations and criteria for plug-ins to a server-based project made ALF appear less prepared to exit the Validation Phase than it actually was. In addition, the ALF team, in its expectation of entering the Implementation Phase, had failed to appreciate how early in the Eclipse lifecycle project some of the less-well-documented criteria would be applied. Finally, also, a few points were either overlooked or not fully appreciated, for example the packaging of convenient executable downloads, or the desirability of unit tests for prototype, proof-of-concept grade code during the validation phase.

Given the breadth of vision and ensuing complexity of ALF, the team placed an emphasis on architecture and design and the community review of those documents over producing lines of code. In the judgment of the ALF team, this was a sensible tradeoff given the scope and fundamental nature of the project. The ALF team expects the volume of code produced to accelerate due to existence of a well considered and community-reviewed architecture, so the lines-of-code counts will take care of themselves naturally.

Armed now with a better understanding, ALF has been assembling this plan to address these items and prepare for a rescheduled Validation Review.

#### Current ALF Milestones

To date, ALF has had two "milestone" releases:

- 1.0M1 Event Manager and Configuration tool
- 1.0M2 Update and Sample

However, ALF has not properly packaged and promoted these sufficiently. For example, it is customary for Eclipse projects to provide a "build" for milestone releases. We have provided only a source download for a portion of the first Milestone.

ALF is aiming at providing two releases candidates: 0.7RC1 and 1.0 RC1

The 0.7 release contains the essence of the functionality for ALF. We have defined this release to satisfy two goals:

- Provide early preview to the Eclipse community
- Serve as a vehicle for performing a Release Review . This is largely to provide practice for the later 1.0 RC Validation Review, so we have the mechanics down pat prior to the 1.0 RC Validation Review

## Overall Approach and Plan

### Enhancements to the ALF Admin tool

While ALF provides a server-based framework, APIs and extensibility mechanism – largely in the form of extensible vocabularies and services – there is a fairly trivial ALF Administration component that is a plug-in. Because of its relative insignificance, the ALF team did not spend much time on polishing the plug-in during the incubation phase. This work was originally targeted for the “Implementation” phase. However, much external attention was focused on this relatively peripheral component during the PMC review. Even though the value provided by ALF lies squarely in the server-side components, the team will investigate whether defining an API for such a trivial model is warranted.

We will review and refine the ALF EventManager Admin tool for consistency with the quality goals expected of Eclipse plug-ins. Note that due to its inherent nature as a very simple configuration tool, we do not expect to define even a provisional API., Defining such an API, where one is not justified would be artificial. Having said that, we do plan to review the model and UI to see if an API is warranted, and if it is, to define a provisional API. In essence, we will do what makes the most sense for the community.

### Review ALF CVS repository for proper structuring, versions and labels

Review of the ALF code CVS repository to ensure that labels are applied, and making sure the naming matches Eclipse naming conventions. One difficulty here is that the conventions are not well described and the application of them vary widely across Eclipse projects.

### Better align Milestones with Eclipse numbering conventions

While ALF has followed the Eclipse development practice of milestones, we had not understood the preferred numbering scheme. This is a fairly mechanical change and will be fixed.

### Improve and increase use of Bugzilla

Entry of selected additional tasks into Bugzilla, and review of existing tasks for updates status, etc. Because the early code base has been remarkably stable and bug-free, there was little need for Bugzilla. Nevertheless, as a good practice, ALF will make more

extensive use of Bugzilla. This is a very natural development as the volume of contributed ALF code increases.

#### Improve community access to executable downloads

ALF has been cited for not providing convenient downloads of ALF executables. That criticism is valid and we will correct the situation..

#### Promote the notion that Eclipse is more than just an IDE

ALF has an education challenge regarding the definition of a community of ALF "framework" user and ALF "tool" users. For ALF, the framework is on the server side, where the equivalent of "plug-in providers" provide server-side tools that plug into the ALF framework. The traditional notion of Eclipse plug-ins largely does not apply. ALF does provide plug-ins for trivial administrative and configuration tasks where an extensible framework makes less sense.

ALF's goal is to build an extensible framework for process integration and tool interoperability. The extensibility focus is on tool vocabularies (e.g. web service interfaces). We have realized it is critical to proactively communicate the nature of ALF to the broader Eclipse community, beyond what was done at Creation Reviews and via WIKIs, Mailing lists, regular conference calls, and newsgroups, to avoid confusion and ensure ALF is viewed correctly. From a perspective centered around traditional Eclipse plug-ins, ALF will largely disappoint, since plug-ins are not emphasized. Instead, the focus on extensible frameworks has an emphasis on the server side aspects, so ALF embraces the Eclipse philosophy, but extends the physical forms of extensible frameworks.

#### Provide feedback to Eclipse community toward improving the process

Bringing the ALF project from concept to implementation has been a learning experience, mostly joyous and occasionally painful. We are interested in providing feedback to the community to improve the process, and are encouraged by the directions of the new Eclipse Development process, especially the idea of mentoring.

## Specific Action Plan

### **ALF 1.0M3 Milestone release.**

Package a build of the cumulative hard deliverables from Milestone 1.0M2 and call it the Milestone 1.0M3 download.

Subtasks:

- Create the download build .zip.
- Create/edit our download page to be more explicit. A good example can be found at: <http://www.eclipse.org/vtp/downloads.php>
- Describe milestone 1.0M1 and 1.0M2 as source code only milestones and refer people to CVS.
- We should remove the current event manager source download, as it has caused more confusion than good.
- Upload the build to the eclipse download site. For example, <http://www.eclipse.org/downloads/download.php?file=/technology/vtp/vtp-M1.zip>
- We should explicitly solicit feedback on the milestone build and encourage our audience to submit bugs in Bugzilla.

### **Plan for Release 0.7RC1**

There will be new code deliverables beyond those of the 1.0M3 release:

- SSO components
- New common services - Logging, Notification, Relationship.
- Refined Event manager (SSO)
- Refined Event Manager Admin tool., including a small provisional API and supporting specification and validation tests

Process and methodology oriented tasks:

- Create JUnit tests for components where appropriate.
- Daily builds established.
- A Test plan in place.
- Address bugs/issues found in Milestone .3.
- Create a Release 0.7RC1 build download package.
- Schedule 0.7RC1 as ALF's first Release review.

### **Plan for Release Candidate 1.0RC1**

Milestone 1.0 adds the following hard deliverables according to the current plan:

- Integration of ALF SSO components.
- Additional integration testing.
- Common Services/libraries - Batch, CommandLine EventGeneration Client

Process and methodology-oriented tasks:

- Schedule ALF's Validation Review (and second release review) to cover Release Candidate 1.0. Note that we could consider having this be the Validation review.

ALF has an education challenge regarding the definition of a community of ALF "framework" user and ALF "tool" users. For ALF, the framework is on the server

side, where the equivalent of "plugin providers" provide server-side tools that plug-into the ALF framework. The traditional notion of Eclipse plugins largely does not apply. ALF does provide plug-ins, but for trivial administrative and configuration tasks, where an extensible framework makes less sense.

Suggested optional tasks:

- Look for Eclipse plugins that would make good candidates to initiate ALF events. Note that a sample that ties CVS/Subversion with ANT and Bugzilla would be ideal, though realizing that will depend on community contribution, and is likely to occur after 1.0 RC

## Observations and Recommendations for Eclipse

- The original Eclipse process document is quite clear and very well aligns with our original interpretation. However the subsequent "clarification and guidance" has caused confusion and uncertainty, and based on our recent experience completely changed the meaning and intent of the validation checkpoint review.
- While the ALF team has religiously followed the description of the Validation phase, the Eclipse documentation for the Implementation phase contains additional guidance for exiting the Validation (aka Incubation) phase. All Eclipse projects can benefit from better organization and centralization of the relevant material.
- Consider whether the criteria for exiting the Validation phase should vary depending on whether a project starts with a large body of contributed code or simply a concept. For example, for ALF, its foundational nature and scope argued for a well-considered architecture and design, coupled with "proof of concept" demo to validate the core ideas. However, this resulted in less code, fewer bugs (the quality was high and the volume of code was low), and less of a perceived need for thorough testing of a "proof of concept". Design bugs were debated on the WIKI, mailing lists, a newsgroup, rather than in Bugzilla. A project that begins with a large volume of contributed code, naturally has larger LOC counts, a working code base against which bugs can be reported, etc.
- Eclipse follows a laudable model of allowing the initiation of many projects but raising the "quality criteria". While that general concept is easily understood, the specifics about what criteria are applied to judge quality at each phase could be much better defined.
- (Minor suggestions):
  - Standardize on one term (either "Validation" or "Incubation") to describe the phase
  - In describing the phases, place information in the phase where the information is relevant. For example, much of the useful information on Reviews is found in the description of the Implementation phase.
  - Reconcile the information on bug reporting between the Bugzilla Guide and the Eclipse Process Guidelines. For example the guidance for

formatting versions in the Bugzilla Guide does not match the Eclipse conventions for naming versions.