Agent Modeling Platform (AMP) Incubation Initial Release: 0.8.0

Review Materials June 15, 2010 Prepared by Miles Parker

Introduction and Purpose

This document is to fulfill the requirements of the Eclipse Release Review for the Agent Modeling Platform (AMP) Incubation project planned for release at the end of June or beginning of July, 2010.

This will be the first official release of AMP, though many of the underlying tools have been released in previous incarnations outside of Eclipse, including the bundled Escape API which is now at 5.5.0 and has been under active development for 12 years and “MetaABM” which began development in 2006. AMP was approved as an Eclipse incubation project as a sub-project of the Eclipse Modeling Project in April of 2009.

Component Overview and Status

AMP project provides extensible frameworks and exemplary tools for representing, editing, generating, executing and visualizing agent-based models (ABMs) and any other domain requiring spatial, behavioral and functional features. It includes:

AMF Meta-model and Infrastructure

Overview

AMF provides an ABM meta-model representation, editor, generator and development environment. The AMF Acore meta-model is similar to EMF Ecore and defined in Ecore, but provides high-level support for complex agents. AMF generates complete executable models for a number of Agent-Based modeling APIs as well as Java Skeletons and Interfaces, JUnit test cases and documentation and is easily extensible to support additional targets.

Status

The Meta-Model actually has two planned incarnations. The current meta-model “metaabm” was originally slated to be replaced immediately with “acore” but rather than deal with migration issues twice, we’ve elected to keep the metaabm name for the time being while making forward compatible changes to metaabm, and differ the acore transition to the 0.9.0 release. The meta-model is quite stable now and we don’t anticipate making any breaking changes to it. We are adding new features through the “acore” release that will not be backward compatible.

The meta-model is also suited to other modeling and simulation domains (here we use modeling in the sense of models of real world entities) and we are working with potential contributors to support Systems Dynamics representations which may appear as early as the 0.9.0 release.

AGF and AXF: APIs and Runtime

Overview

The Agent eXecution Framework provides services and UI for model management, execution, and views. Arbitrary toolkits can easily integrate with Eclipse and AXF by implementing pluggable providers like engines, agents and view parts. These are suitable to a wide variety of domains and use cases outside of the Agent Modeling target and a near term goal is to evangelize these uses outside for projects outside of AMP and Modeling.

The Agent Graphics Framework extends GEF, GEF3D, Zest, and the BIRT charting engine to support real-time visualization of and interaction with agent models. AGF currently provides support for 2D, 2 1/2 D. and graph structures, and will be extended to 3-D, GIS and others. As with other AMP components, the AGF design focus is to provide an extensible infrastructure so that platform adopters can easily create their own view and editor parts and we would also like to broaden usage.

Both components have significant runtime components -- including UI support for specialized views and controls -- as well as fully abstracted and extensible APIs.

Status

Both AXF and AGF are more mature, capable and generalized than we had expected them to be at this point. We’re especially happy with the support for extensible graphics and spatial abstractions using the common Eclipse provider pattern. As EMF has demonstrated this pattern works particularly well for generated code that can then be manually customized and extended. We are not ready to move from

platform API as significant changes may still be made, but users should be assured that any future updates will be manageable. The basic runtime environment is under unit testing and the overall environment has been used significantly and provides a solid overall user experience. There is extensive support for handling exceptions and other issues that come up during model execution. The major unknown aspect of AXF and AGF at present is how well it behaves under more generic use -- i.e. wether it might actually be too “tuned” to the Escape use. We need to find outside adopters of the technology in order to ensure that the API truly is general enough and works well enough for a broad set of users.

Escape: ABM exemplar environment

Overview

Escape is an exemplar ABM toolset and provides all of the tools needed to do complete model execution and visualization of ABM models. It allows modelers to code in Java and/or generate models with AMF and then execute those models within the same development environment.

Status

As stated above, the API that Escape is based on is very mature. It's based on Ascape, which has been in use for more than 10 years. The core API is very stable, and that should give users a way to explore the features of AMP without concerns about keeping in synch with the rapidly evolving AXF /AGF API. There remain some testing and minor UI issues but overall the release would make a good quality x.0.0 release within a non-incubation context.

Quality

Bugzilla

AMP has had 77 official bug reports total, of which 52 (67%) have been resolved. Many of these bugs are very high-level and have represented large chunks of functionality. All but five of these bugs have been self-reported. At this point in the development process the committer(s) have often resolved bugs on new features without reporting them to bugzilla and we need to do a better job of providing bugzilla traceability in the future and that will become more important and easier as a broader user base develops.

Standards

We’ve made a strong effort to conform to all Eclipse standards, conventions, practices and idioms. We believe that our current code base is at or near a 1.0.0 level of quality with respect to this and appreciate any feedback on areas where we do not fully conform.

Testing

Unit Testing

Unit Testing support is mixed. In some areas our support is very good. For example we have developed a testing framework for our EMF.Edit code that is general enough to be useful for other Eclipse EMF based projects. AXF and AGF unit testing coverage is too light in some areas. We have also been challenged by the lack of easy to integrate testing apparatus in two important areas. First, while we do have fairly extensive SWTBot based tests written for the AXF and AGF runtime they are not often run because they take too long on a local machine, and we have not had the time or resources to overcome issues with integrating them into our Hudson/Athena based build environment. As the Athena team has resolved a number of these issues we’ll be looking toward integrating this for the 0.9.0 release. Second a

long-standing issue has been testing the actual generated code. There are difficult environment boot-strapping issues here that we have attempted a number of solutions for but have not been able to find an adequate solution to. We need to develop a good strategy for this to have the confidence we need to make large scale changes in the meta-model such as required for the transition to Acore. Input from other teams would be greatly appreciated.

Blackbox Testing

We're aiming to integrate FindBugs coverage in our automated build soon, at least by our 0.9.0 release.

Non-Code

We've made an extensive effort at end-user focused documentation and user experience support, including roughly 150 pages (pdf formatted) of user documentation, intro page support cheat sheets and help integration.

Communities

Community building is currently the greatest challenge for our project. AMP is a relatively new project with a usage domain that seems complex and even esoteric even within our target domain. Still there is a very broad community of potential users. We've worked very hard to make the tools more immediately accessible and transparent and attended many domain conferences where the tools has been very well received. Now that we have a solid foundation, with good documentation and support, community building should be our strongest focus.

Committers

"We" still have only one committer, the author of this review. Finding other committers with the background in the ABM domain along with an interest and capability to work with complex Eclipse technologies has been difficult. However, we have had a lot of discussions with potential participants and as this is written we have a group who have decided to participate in developing the Systems Dynamics portion of the model. This is very exciting as we don't have a good project until I can stop writing code!

End-Users and Adopters

We have had many people use the tools and provide feedback in the last year. Our challenge is converting those users into active end-user developers who use the tool in their day to day work. There are a number of established tools in the domain and AMP provides a different paradigm for design that requires a break from the current way of doing things. Here, the leverage that the tool provides should win over users and the new documentation has already started to pay dividends. Also the diverse set of potential users many of whom are incidental programmers and some of whom don't write code at all means that the initial user experience is critical. Even the much improved update process is too much of a barrier of entry for these users, so we're going to be providing a full platform build. We also need to work at getting people to use the Eclipse community forum for communication rather than contacting the committers directly!

IP Issues

Our project has submitted our IP to Eclipse legal. See: http://www.eclipse.org/projects/ip_log.php?projectid=modeling.amp It includes:

- A list of third party software distributed with AMP, with a link

to the relevant CQ.

- The name of every committer for this release.

- No non-committer have provided code to AMP.

- There are no required third party components. (AMP does have a dependency on GEF3D, which in turn has a technically optional dependency on the LWJGL library but our understanding is that this is not a case where we need a CQ.)