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 the Indigo simultaneous release in June 2011.

This is the second AMP release and the first as part of a regular release train. A lot of project effort since the prior release has been related to getting up to speed on build, process and other infrastructure issues. As we mentioned in the prior review documents many aspects of the current toolset are quite mature and have been in active development since 2006, with some component dependencies in active development since 1998. AMP was approved as an Eclipse incubation project as a sub-project of the Eclipse Modeling Project in April of 2009. AMP is under active development and has a growing feature set and community.

Component Overview and Status

The AMP project provides extensible frameworks and exemplary tools for representing, editing, generating, executing and visualizing models of real world systems. This includes agent-based models (ABMs), Systems Dynamics models, and any other domain requiring spatial, behavioral and functional features. We are actively exploring expanding the overall scope of this project to include support for
more general science and math functionality including for example statistical, equation-based and other constructs. Currently the system is composed of:

**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 and takes a graph driven approach to representing relationships. 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” will be mature as of the 0.9.0 release. The upcoming “acore” release slated for 1.0.0 will not be backward compatible but we will provide a model-to-model transition path.

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 now have active contributors for Systems Dynamics functionality which will be part of this 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 capable, generalized and relatively mature. 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. The APIs are near platform API but may still some changes up to the 1.0.0 release. 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.
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 Escape API is completely platform API, 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 138 official bug reports total, of which 84 (60.8%) have been resolved. A significant number of remaining bugs will be resolved for Indigo release. Of the current bug list, more than 20% of the bugs have been assigned to or seen significant contributions from our new contributors and this number is expected to grow.

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. As always, more javadoc is needed and we’ll focus on this as we move toward a 1.0.0 release.

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 we still have challenges in the build infrastructure for setting up high-level SWTBot UI functionality tests. Because of the highly interactive nature of the environment and the complex visualization involved, this is a significant challenge. Our greatest challenge is still in the testing of generated code. There are difficult environment boot-strapping issues here, but our new contributors are also very interested in making this work well and we should be able to finally address this issue after we get through the current release.

Non-Code
We have extensive 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 but it is in strong need of updating and refinement. We’ve had two outside non-developers offer to make significant contributions to this effort which will be a significant benefit to the project.
Communities

In the last release we mentioned that community building was the greatest challenge for this project moving forward. That along with getting into the regular release train has been the most significant non-code related effort this year, and I’m happy to report that we’ve made significant progress. We have our first outside contributors, Jonas Ruettimann from the Institute for Modelling and Simulation, FHS St.Gallen in Switzerland. They have developed the Systems Dynamics portion of the model which is a key part of the functionality we can offer. We have also had a significant contribution from Oliver Mannion in New Zealand and have been discussion ways in which his efforts and those of his organization can become more deeply involved in the effort, perhaps as active contributors. Finally as mentioned above, our user community is growing and we’ve had two users step up to offer significant assistance in editing and developing new documentation. At this point, our challenge becomes demonstrating the relevance of the toolset to a much larger audience, in the Eclipse community in general, the Eclipse modeling community and also with the general science community. We’re currently in a niche within a niche in that we have to find users who are both somewhat familiar with Eclipse ecosystem, and that have an interest in Complexity science and modeling real world systems. To break out of these niches, I feel that the project must take a much broader view of its overall role in the open source science-software ecosystem, but discussion of that is outside of the scope of this document.

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.
- One non-committer has provided code to AMP.
- There are no required third party components for the components that will be released with Indigo.
- AMP does have an optional dependency on GEF3D that supports our 3D visualization capabilities. (We are currently hosting the GEF3D features while we work with the GEF3D team to develop their own hosted build.) GEF3D in turn has a (technically optional but required for functionality) dependency on the LWJGL library. LWJGL uses LGPL and has other IP issues and there is an active CQ related to this. The GEF3D dependency without the LWJGL will be hosted on the Eclipse AMP update site but that feature will not be part of the B3 aggregator build.