Roadmap of Eclipse Modeling Platform

October 29th 2010

Report prepared for Alcatel-Lucent, Ericsson, Swift, UBS

Authors

Dr. Martin Mandischer (itemis), Dr. Stephan Eberle (Geensys)

Note:

The overall project plan consists of 2 parts whereas this document is the second part that delivers the roadmap and process definition. The first part, a gap analysis, was delivered on October 1st, 2010.

Genesys 6 avenue de Verdun, 92250 La Garenne Colombes, France **T** +33 (0)1 73 54 00 36 **F** +33 (0)1 73 54 00 32 http://www.geensys.com
Itemis Am Brambusch 15–24, 44536 Lünen Germany **T** +49 231 9860-210 http://www.itemis.com

Content

1 Executive Summary

The present document delivers an executable plan for realizing an open source Modeling Platform on Eclipse. The starting point was a document describing the requirements for such a Modeling Platform from an end-user perspective. In a gap analysis these requirements were expanded into more detailed capabilities and initial priorities for each of them has be defined. The gasp with respect to existing eclipse project have been analyzed and effort estimates for missing features have been established.

The ultimate goal is to fully bridge the gap between the existing Eclipse modeling projects and the requirements in order to provide an open source Modeling Platform that meets the end-user companies' needs. However the investment implied by the initial top priority items was to high for being addressed in a first step.

The initial priorities have been refined in order to determine which requirements and detailed capabilities can be realistically implemented in a first development phase. The presented roadmap is based on these refined priorities.

In addition to the roadmap a process for the development of the modeling platform is proposed.

To summarize, the contributions of this report are:

1. A feasible executable project plan with budgets and milestones based on top priority requirements.
2. Identification of the potential solution provider.
3. A process definition for the development.
4. An update of the gap analysis spreadsheets reflecting the refined priorities and some additional requirements and capabilities.
5. A number of suggestions of how the IWG's future collaboration could be organized.

2 Roadmap

This section presents a roadmap for a first development phase of the Eclipse Modeling Platform. The roadmap was produced by itemis and Geensys on behalf of the MPIWG.

The roadmap includes:

- Staffing and budgeting plan.
- 4 Milestones with delivery dates and estimated budgets.
- Themes and established priorities, allocation to milestones and resulting milestone efforts are given in the updated gap analysis document.
- Risks involved with respect to activity/maturity of underlying Eclipse projects and leeway of Modeling Platform requirements have not been treated explicitly. Instead risks have been addressed in effort estimates and project selection.
- Potential responsible and involved companies and/or individuals.

2.1 Refined Priorities

The following table yields the effort estimates resulting from the gap analysis that has been conducted prior to establishing the roadmap. It indicates the total effort required to implement all requirements of the modeling platform and the total effort for all initial priority 1 features that ideally ought to be realized in the 2011 development phase.

		2011 Initial Prio 1	***Total***
Total Effort in Persons/Year		***37***	***101***
Management	25 %	72	195
Platform Integration	5 %	14	39
Integration Testing	25 %	72	195
Development effort		288	781

The user companies' feedback on the gap analysis gave evidence to that it is not likely to get enough funding to cover all priority 1 features in 2011. Therefore a further refinement of the priorities has been achieved within the MPIWG.

2.2 Assumptions

The following additional assumptions had to be made to produce a realistic roadmap:

1. It is crucial to have a small but solid start and a long-term perspective for the modeling platform project.
2. To produce a realistic roadmap we start with a feasible team setup.
3. As the requirements are not detailed enough for direct implementation an iterative process like Scrum will be used to further refine requirements and achieve regular deliverables.
4. Milestones are quarterly delivered feature sets. Depending on the still to be produced more detailed definitions of requirements and the actual project progress the features sets may change throughout the project.
5. The development will take place as a single project with a dedicated and distributed project team.
6. The potential team can only start developing the features after funding has been confirmed. This remains open.
7. Instead of padding the project with extensive buffers we consider features with lower priorities as optional in case of unexpected problems or scope changes.

2.3 Staffing and Budgeting

This section proposes a feasible team setup to realize the features addressed in the 2011 roadmap. The following staffing plan depicts suggested resource allocation over time. In combination with an average labor rate it also gives us budget estimates for the quarterly milestones.

Staffing assumptions:

1. We assume that it is feasible to have a distributed technical team of 8 – 10 people working on the project; 3 – 5 additional persons would be required for management, integration and testing purposes.
2. We need a ramp-up phase starting with 5 people and grow to 12-15 people within the first 3 months. The ramp-up phase starts with architecture and a more detailed definition of the first requirements in parallel with the collection of real-world test cases as basis for system and integration testing.
3. This leaves us with approximately 8 PY for feature development plus approx 4 PY of integration testing and management.

The following tentative staffing plan is based on these assumptions:

		Q1			Q2			Q3			Q4		
Person	Category	January	February	March	April	Mai	June	July	August	Septembe	October	November	December
NN1	*Manag / Req.*	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN2	*Manag / Req.*		17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN3	*Integration / Test*			17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN4	*Integration / Test*			8,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN5	*Integration / Test*	8,0	8,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN6	*Dev*	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN7	*Dev*	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN8	*Dev*	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN9	*Dev*		17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN10	*Dev*			17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN11	*Dev*			17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN12	*Dev*				17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN13	*Dev*				17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0	17,0
NN14	*Dev*					17,0	17,0	17,0	17,0	17,0	17,0	17,0	
NN15	*Dev*					17,0	17,0	17,0	17,0	17,0	17,0		
	Days per month	76	110	178	221	255	255	255	255	255	255	238	221
	PM / Budgets		364.000,00 €			731.000,00 €			765.000,00 €			714.000,00 €	
	Man. / Dev / Total PM	8	13	21	15	28	43	15	30	45	15	27	42

As we have not done any recruiting the labor rates are based on average rates for experts in the modeling field.

Labor rates	
Min	800€
Max	1.200€
Average	1.000€

Estimated costs	
Average # Persons	12,5
Work days per Year	200
Average cost	**2.500.000€**
Budgeted cost	**2.548.000€**

The estimated cost will be approximately 2.5 Million € with varying quarterly budgets through the year.

2.4 Milestone Plan

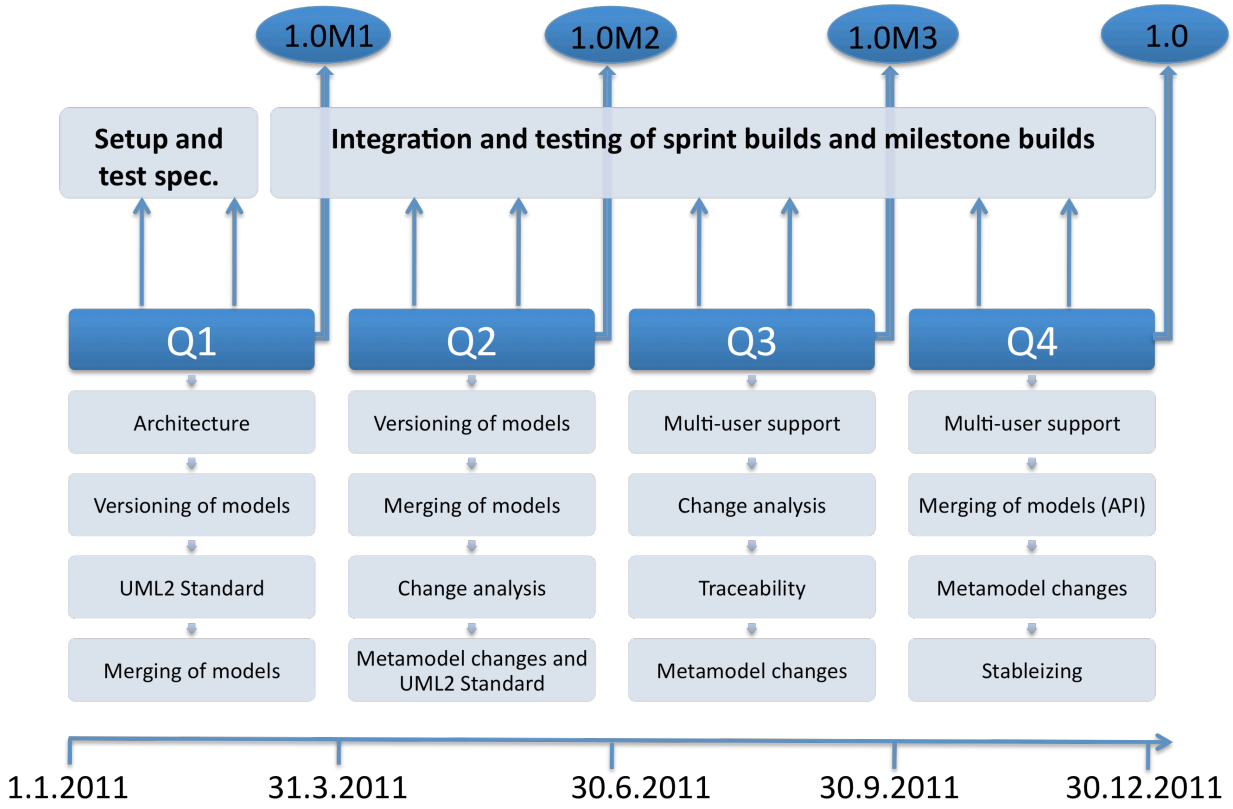
The selection of the features to be included in the milestone plan for 2011 was done in 3 steps:

1. Refinement of priorities by end-user companies, at least at a per-requirement level
2. Selection of requirements and detailed capabilities to be realized according to refined priorities, dependencies and relevance in terms of architecture.
3. Elimination of redundancies, i.e., decision on which technology or approach is to be used in case that several alternatives are available.

The new priorities and milestones are incorporated in an updated version of the gap analysis spreadsheets. The resulting new “Roadmap 2011” spreadsheet replaces the former gap analysis spreadsheet and includes all previously available information. The new priorities derived from user company input are expressed in priorities 1,2,3,4,5. For the remaining former priorities we added a 0 as postfix to the priority, i.e., P1 became P10, P2 became P20 etc..

We also added target milestones on the level of detailed capabilities as well as summed efforts for the milestones.

We plan to deliver regular increments on a monthly basis with the milestone deliveries every 3 months. Testing and integration is performed on the monthly builds. The following figure gives a high level overview of themes and topics addressed in the different milestones.



The following sections describe the planned milestone deliveries with their total *development* efforts. Setup, testing and management effort is not included in theses figures. The budgets for the milestones are given in section 2.3Staffing and Budgeting.

2.4.1 Q1 Milestone 1.0M1 – Architecture

Release date: 31. March 2011

Development effort for milestone: 9 PM

Deliverables:

1. Architecture of modeling platform. Concepts and first implementations
2. Setup of team and technical infrastructure

3. Theme: A1. Managing versions on various model granularity, meta-model and instances
   {1.1 a)+E-Config-1} Versioning of metamodels and instances

   a. Versioning of a model. Supported in CDO, but capacity to label/tag committed changes still missing (currently only commit number is exposed)

4. Theme: A3. Merging of different models and versions of models
   {3.1 a) + E-Comp-1} Parallel work on different parts of a model and merge of resulting changes into a common model

   a. Support for 2-way merge

5. Theme: A6. Support for meta-model change and update to appropriate instances
   {6 c} Support for different versions of a metamodel in the same environment

   a. Capacity to accommodate, distinguish and access different model instances that coexist in the Eclipse workspace; support for models with different scopes of included projects and files (e.g., models that are mapped to a project and include all resources with a given type inside that project, models that are mapped to a folder in a project and include all resources with a given type inside that folder); support for adding/removing dependencies between such model instances

   b. Capacity to handle such model instances being based on the same metamodel

   c. Capacity to handle such model instances being based on different versions the same metamodel

   d. Capacity to handle such model instances being based on different metamodels

   e. Ability to configure the version of a given metamodel to be used in a project (via project properties) and the default version of that metamodel in the workspace (via workspace preferences)

6. Theme: D1. General purpose models based on industry standards like UML, BPMN and SysML
   {E-Auth-1} Support for the full UML 2 standard

   a. UML2 metamodel support

2.4.2 Q2 Milestone 1.0M2

Release date: 30. June 2011

Development effort for milestone: 33,5 PM

Deliverables:

1. Theme: A1. Managing versions on various model granularity, meta-model and instances
   {1.1 a)+E-Config-1} Versioning of meta-models and instances

   a. User interface for mapping models from one or several CDO repositories to projects in the workspace and navigating through them (includes CDO Repository Exploring perspective with CDO Repositories

view and other related views, Project Explorer displaying workspace projects and mapped model resources and elements inside, actions for creating new model resources and elements, opening model resources and elements in editors, etc.)

   b. User interface for inspecting and managing model versions (includes History view, action contributions to Eclipse "Replace With" context menu, etc.)

   c. User interface for inspecting differences between 2 model versions on same branch (includes 2-point Compare view/editor operating on a single change set, corresponding action contributions to Eclipse "Compare With" context menu, etc.)

2. Theme: A3. Merging of different models and versions of models
   {3.1 a) + E-Comp-1} Parallel work on different parts of a model and merge of resulting changes into a common model

   a. Support of model repository backends, in particular CDO

   b. Comparison/merge of model elements or fragments instead of entire resources/files

   c. Support for logical model merging

   d. Support for loading minimal set of resources/files

   e. Operation on shared model instances in the workspace

   f. Model-oriented dirty state management

   g. Editor-relative undo/redo contexts

3. Theme: A4. Change analysis to identify root cause and impact analysis for changes to all levels of model content
   {4.2} Link of change request to resulting model changes

   a. Support for Mylyn task-based change set management in model repositories, in particular CDO (i.e., sort of Mylyn SCM connector for CDO)

4. Theme: A6. Support for meta-model change and update to appropriate instances
   {6 a} Support for automatic application of metamodel changes to model instances

   a. Extensible model converter service enabling model instances to be migrated forth and back between different metamodel versions; capacity to implement and contribute user-defined model converters for specific metamodel version combinations; ability to provide a human readable description of the conversion strategy for each contributed model converter

   b. Automatic application of appropriate/selected model converter when loading/saving model instances

5. Theme: A6. Support for meta-model change and update to appropriate instances
   {6 c} Support for different versions of a metamodel in the same environment

a. Generic wizard for creating projects supporting a given metamodel with a user-selectable version of that metamodel

b. Generic wizard for creating model resources corresponding to the metamodel and version which are configured on a project

6. Theme: D1. General purpose models based on industry standards like UML, BPMN and SysML
{E-Auth-1} Support for the full UML 2 standard

a. Model repository support, in particular CDO

b. Compare and merge support for UML including the ability to see changes to stereotypes applied to an element

2.4.3 Q3 Milestone 1.0M3

Release date: 30. September 2011

Development effort for milestone: 39 PM

Deliverables:

1. Theme: A1. Managing versions on various model granularity, meta-model and instances
{1.1 a)+E-Config-1} Versioning of meta-models and instances

a. User interface for inspecting and managing model versions (includes History view, action contributions to Eclipse "Replace With" context menu, etc.)

2. Theme: A1. Managing versions on various model granularity, meta-model and instances
{1.2 a)} Support of multi-user and distributed development teams

a. Capability to define user profiles

b. Capability to define users and associate them with profiles

c. Capability to associate user profiles with access privileges, i.e., to define which kind of changes on which parts of a model the users of each profile are allowed to do

d. Capability to authenticate users when they connect to model repository, i.e., to identify them through an interactive or automated login process

e. Capability to authorize users when they read/write the repository, i.e., to approve or deny the part of the model they attempt to access and the kind of change they are about to do according to their associated user profile

3. Theme: A1. Managing versions on various model granularity, meta-model and instances
{ Use Case + E-Collab-1} Support for tracking, describing, annotating, reporting of model changes

a. Traceability of the actions of users on the models

4. Theme: A3. Merging of different models and versions of models
   {3.1 a) + E-Comp-1} Parallel work on different parts of a model and merge of resulting changes into a common model

   a. Integrated with Eclipse team component

   b. Editor support for presenting changes in a tree or flat structure. Editor support for grouping changes by EClass, physical file or repository and change type

5. Theme: A5. Traceability to identify connections between model elements
   {5.1 a)} Creation of relationships between model elements that are independent of their metamodel(s)

   a. Creation of traceability relationships between two elements in a same model instance

   b. Creation of traceability relationships between two elements in different model instances that are based on the same metamodel

   c. Creation of traceability relationships between two elements in different model instances that are based on different metamodels

6. Theme: A5. Traceability to identify connections between model elements
   {5.1 b)} Visualization of these relationships and traceability between model elements

   a. Visualization of traceability relationships in same model instance

   b. Visualization of traceability relationships between different model instances that are based on the same metamodel

   c. Visualization of traceability relationships between different model instances that are based on different metamodels

   d. Support for different kinds of visualization (e.g., tree-based browser view, table-based view, graphical view, decoration/highlighting of related elements in explorer views, filtering of explorer view such that only related model elements are shown)

   e. Analysis of traceability relationships (e.g., listing of impacted model elements when a given model element is changed, listing of all covered/uncovered elements in a given model instance)

   f. Navigation between related model instances and elements (i.e. opening related model elements in editors, selecting related model elements in explorer view)

7. Theme: A6. Support for meta-model change and update to appropriate instances
   {6 a} Support for automatic application of metamodel changes to model instances

   a. Capacity to configure metamodel version to be used when model is saved/persisted (e.g. via workspace preferences or project properties)

2.4.4 Q4 Milestone 1.0 – Modeling Platform

Release date: 30. December 2011

Development effort for milestone: 20 PM

Deliverables:

1. Theme: A1. Managing versions on various model granularity, meta-model and instances
   {1.2 a)} Support of multi-user and distributed development teams
   a. Capability to retrieve existing user and user profile definitions from external user management systems (e.g., LDAP, Active Directory)
   b. Support of usual Eclipse team workflow operations (including Team Synchronizing perspective and Synchronize view, action contributions such as commit, update, override, etc.) when working in disconnected (offline) mode
2. Theme: A3. Merging of different models and versions of models
   {3.1 b) + E-Comp-1+E-Comp-2} Automatic detection of inconsistencies in merged model and decorations identifying those
   a. Automated detection of rule based conflicts
   b. Automated detection of basic conflicts
3. Theme: A3. Merging of different models and versions of models
   {E-Comp-1+E-Comp-2} Support for filtering changes
   a. User interface support within editor to select filters
   b. Filters for different atomic and "rule based" changes/differences
   c. Filters using string patterns
   d. Ability to extend/add available filters
   e. Filter selection and configuration through preferences
4. Theme: A3. Merging of different models and versions of models
   {E-Comp-1+E-Comp-2} API support for integrations
   a. API to create a viewer listing all structural changes resulting from a model comparison, which includes the filtering and grouping options as, mentioned in section above.
   b. API to query the viewer for the list of visible changes (e.g. not filtered out)
   c. API to launch an Eclipse Model Compare editor with two specified versions of model files and be able to programmatically request the selection of a structural change based on a previously obtained location.

5. Theme: A3. Merging of different models and versions of models
   {3.1 a) + E-Comp-1} Parallel work on different parts of a model and merge of resulting changes into a common model
   a. Identification of rule based changes
6. Theme: A3. Merging of different models and versions of models
   {Use Case} Support for automatic bulk merge operations
   a. Automated resolution of basic changes
7. Theme: A4. Change analysis to identify root cause and impact analysis for changes to all levels of model content
   {4.2} Link of change request to resulting model changes
   a. Link of Mylyn task context to relevant model repository revisions
   b. Capability to extract change description from differences between affected model repository revisions
   c. View that enables users to inspect model modifications behind repository revisions that are associated with active Mylyn task context
8. Theme: A6. Support for meta-model change and update to appropriate instances
   {6 a} Support for automatic application of metamodel changes to model instances
   a. Capacity to select a particular model converter based on its description in case that multiple model converters for the same metamodel version combination are available
9. Theme: B1. Overall end-to-end lifecycle management/model governance (elements can be in different 'review' states)
   {1.1. a) }Support of review cycles and approvals
   a. Ability for authors to commit a change and submit corresponding Mylyn task for review
   b. Automatic notification of reviewer(s) when a Mylyn task has been submitted for review
   c. Ability for reviewer to inspect latest change behind Mylyn task by comparing it to last approved version
   d. Ability for reviewer to approve change (e.g., by setting a flag on corresponding Mylyn task) or reject it and add some comments about the reason for that to corresponding Mylyn task
   e. Automatic notification of author(s) when a change has been approved/rejected
   f. Ability to store review related task data in underlying task repository (e.g., Bugzilla, Trac, JIRA, ClearQuest, Quality Center, etc.)
10. Buffer for final stabilizing, bug fixes and unexpected problems or changes

2.5 Potential Solution Providers

The potential developer groups, solution provider companies and individuals have been identified in accordance to the requirements and their history in contributing to Eclipse in the modeling field. The following list has been generated from the requirements addressed in the 2011 roadmap. Other developer groups are likely to be required in subsequent development phases. It is not meant to suggest that only listed developer groups could work on the modeling platform. Details on which group might implement which feature are given in the Roadmap 2011 Spreadsheet.

- CDO (Eike Stepper)
- Geensys
- itemis
- Obeo
- Tasktop
- Zeligsoft

Although we have mostly positive feedback of their potential contribution in case of funding this does not mean any formal commitment. It is likely that at project start some of the contributions must come from other resources or individuals as some groups or companies might no longer be able to provide qualified resources.

3 Process Definition

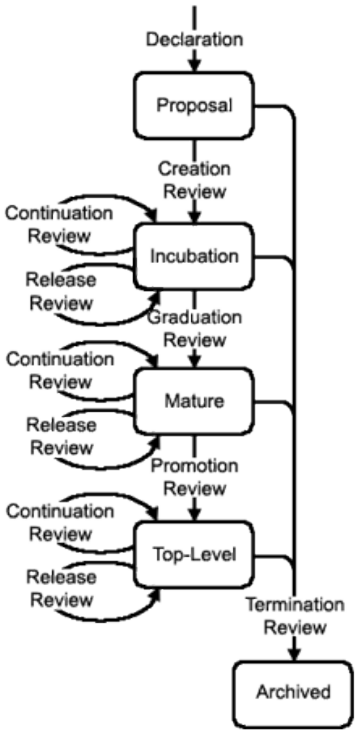
In section we will propose of a development process that fits into the EDP. Our process is closely related to the iterative SCRUM approach and tailored to the specific needs of the MPIWG. The process will define how the development progresses in iterations, how integrations and testing phases will be incorporated and in which way individuals interact with each other.

Main characteristics of the process definition are:

- Iterative, incremental and related to SCRUM
- Embedded into the EDP
- Definition of roles & responsibilities

The Eclipse Modeling Platform is not a typical eclipse project. It focuses on integration and enhancements of several existing Eclipse projects. In this sense it is similar to projects like Amalgam. It is also similar to Sphinx because Sphinx also aims at providing an integrated modeling tool platform. Whether Amalgam or the Sphinx project will be used as a common basis for the Modeling Platform or a new project will be created remains open.

3.1 Eclipse EDP



The Eclipse Development Process (EDP) does not provide detailed instructions as to how a project should be managed and executed.

The EDP is a formal process for the project lifecycle in an open source community. The EDP does not specify how the development team and individuals work together on a daily basis. The tailored Scrum process specifies the latter while the EDP is the formal process accompanying the development process. We assume that the management of the modeling platform project makes sure that the EDP is followed.

I.E. after project funding is given the management must formally bring the project from the Incubation Phase by means of a Graduation Review into the Mature Phase. All initial development takes place in the Incubation Phase while the first stable release of the modeling platform should take the project into the Mature Phase where all subsequent releases will be developed.

Further information on the EDP can be found at Eclipse Development Process.

3.2 Scrum-like Project Approach

We believe that a continuous monitoring of progress and results, frequent integration and testing, and regular feedback from user companies lead to a successful modeling platform project.

The control of a software development project focuses on one of the biggest risks: incompleteness and misunderstandings of the requirements as well as any form of change during the project. Therefore, we suggest the agile management process Scrum that formulates only a few rules, roles and principles, which, however, mean key advantages in the software development:

- Concentrating on the essentials – a consistent prioritization of the detailed capabilities.
- Transparency - deliverable software is presented in very short intervals (approximately 2 - 4 weeks) at the end of each iteration. Every 3 months a milestone build will be release. Thus the progress of the project is not purely based on abstract project plans, but on the executable software. The status of the development, progress and quality is at all times transparent.
- Short, team-oriented coordination - in the so-called Daily Scrums, the project team confers about upcoming work packages, problems etc. Misguided development and obstacles can be avoided or quickly resolved through these daily reviews. With appropriate tooling this is also valid for distributed teams.

- **Regular reviews and retrospectives** - at the end of each iteration (Sprint) a review and a reflection of the results takes place, assessing the critical questions as to what's good and what's gone wrong. The frequent feedback leads to a continuous improvement of the processes and thus results in an improvement of the quality.
- **Simple controlling mechanisms, without complex tools** - unwanted overhead is drastically reduced and thereby productivity is significantly increased.

An extensive description of the Scrum methodology has been purposely omitted. Further information about Scrum:

- http://en.wikipedia.org/wiki/Scrum_development
- http://www.scrumalliance.org
- http://www.itemis.com/scrum

3.2.1 General conditions for the modeling platform project

On the one hand on the project start date most of the requirements are only on a level of detailed capabilities and not precise enough for implementation. On the other hand there is a vision and an end date, which makes it necessary to start the project, in parallel with the completion of the remaining detailed requirement specification.

The following sections show how the agile development process based on Scrum can be implemented with the given time-related conditions.

3.2.2 Roles

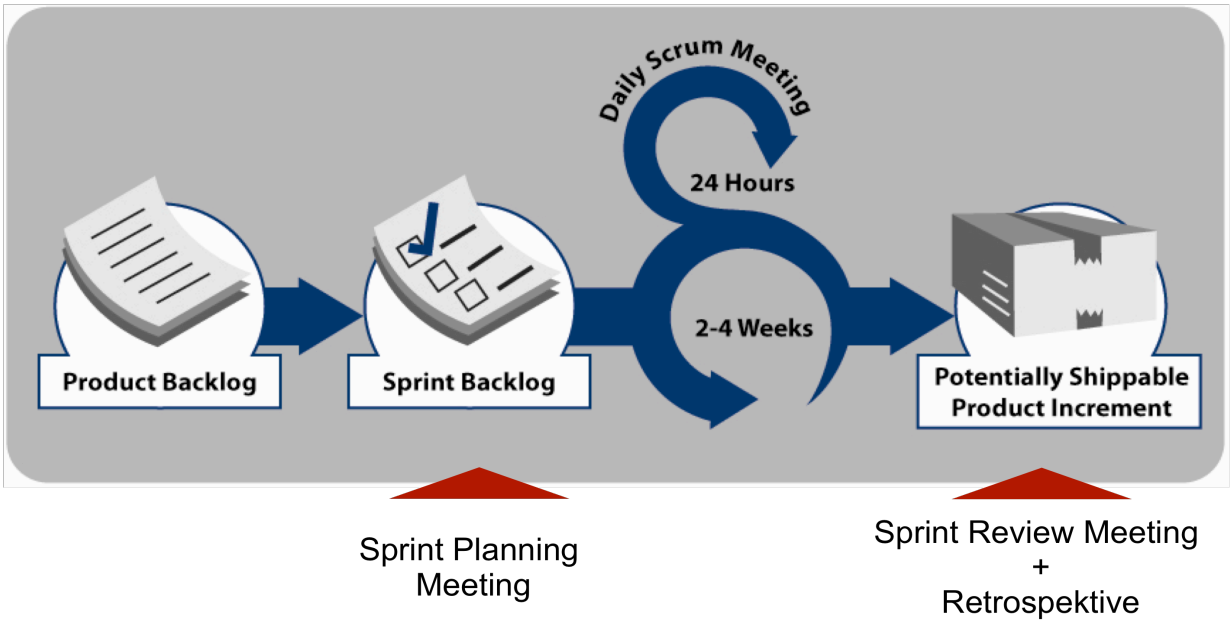
The **Product Owner** (PO) represents the end-users needs and controls the software development, in which he determines – by breaking the work down into more detailed features and prioritizing them – the order of implementation.

In addition, the product owner continuously completes the specification of the detailed capabilities with the help of the representatives from all user companies and the development team. For the MPIWG the PO must be an authorized representative of all user companies. As most requirements are of technical nature he must have sound technological background in modeling and the Eclipse projects. He is also responsible for the stakeholder management and communication

The **Development Team** carries out all work needed for the implementation of the requirements. The result is a potentially ship-able product increment, i.e., an executable piece of software. This development team should be staffed with committers from the respective Eclipse projects. They take up the typical project roles such as developer, architect, tester, database- and modeling experts.

A **Scrum Master**, who among other things coordinates the development process, supports the team and ensures the direct collaboration with the Product Owner.

3.2.3 Sprints (Iterations)



At the project start there must be a prioritized and estimated Product Backlog (list of requirements) and for the Sprint Planning Meeting the highest priority requirements must be described by further elaborated artifacts than only the detailed capabilities. The higher the rank of a Product Backlog item the better should be the understanding and specification of the underlying requirement.

This ensures that a sufficiently well specified and estimated set of requirements exists at the beginning of the Sprint Planning Meeting. In the Sprint Planning Meeting, the development team creates a realistic Sprint Backlog, which will determine which requirements get implemented during the sprint. Within the sprint it is not allowed to change the requirements or the scope of the requirements to be implemented.

A constant sprint length, a well-known team capacity (team hours per sprint), and the estimated expenditures shall solidly determine the amount of practical Sprint requirements. At the close of the Sprints the so-called Sprint Review Meeting is held, in which the Product Owner inspects and accepts the implemented requirements. Not fully implemented requirements shall be deemed as not accepted.

The Sprint Retrospective Meeting is a brief meeting that focuses on improvements of team performance. The development team and the Scrum Master discuss lessons-learned from the previous sprints and decide on improvements.

Through frequent feedback from the user companies a high transparency is achieved as a basis for controllability and goal attainment.

3.2.4 Test automation and continuous integration

To ensure the quality of the working process as well as the work results continuous integration and integration testing has to be performed.

To avoid scenarios with difficult integration phases, we rely on continuous integration and test automation. Because automated daily tests runs make the quality of the software measurable and show possible side effects of code changes directly and identifiably. They serve as a direct feedback for the developer, who perhaps is not capable of looking out over the entire platform at once, in order to detect side effects and follow-up-errors.

The test automation in particular comprises of Real World scenarios that must come from the user companies. The automated test suites must not be limited to testing functional aspects of the features to be realized but also include scalability and robustness (e.g., absence of deadlocks) checks.

Regular builds as well as quarterly milestone build can be released this way.

3.3 Team Setup and Ramp-up

For the initial phase we suggest a small team of 4-5 people as a ramp-up. As it is crucial for project success to quickly develop a sound understanding of underlying architecture and the detailed capabilities, the initial team should consist of persons that either have a strong requirement engineering and software architecture background or a requirement engineering and project management background. The Product Owner must work in very close cooperation with the architects or might even be an architect himself to steer this technology driven project.

The next 6-10 people joining the project must be senior developers familiar with the proposed technologies and Eclipse projects.

In parallel to the development team, a test and integration team will ensure a high quality of the achieved result. Both teams will work very close together to achieve this goal.

4 Summary

The gap analysis identified what functionality would need to be implemented and the effort it would take to achieve an integrated commercial grade modeling tool platform on Eclipse. This plan focuses on transforming parts of the gap analysis into a first version of a modeling platform.

By providing this plan and estimated costs, participating companies have the chance to share the effort and investment of realizing an industry quality open source modeling platform which each of them can use for their own model driven development projects and toolsets.

It needs to be emphasized that the effort estimates expressed in the gap analysis part of the planning are imprecise and are only a rough indicator of the effort that will really be required. Prior to committing development resources, a more detailed definition of the features to be realized in 2011 needs to be provided accompanied with a more advanced draft of the underlying architecture of the modeling platform. The Product Owner will refine requirements throughout the development process while the development team defines the architecture during the ramp-up phase.

As not all of the necessary information was available during planning, the proposed plan is based on several assumptions to make it feasible. Before the plan is put forward these assumptions need to be well understood and accepted.

We belief that a single project team of highly motivated individuals supported by a small test team and a dedicated Product Owner (PO) controlling the requirements and the prioritization will yield the highest chance for a successful realization of the Eclipse Modeling Platform.

The underlying iterative development process, which we propose, ensures that progress and changes are transparent to all participants at any time.

5 Appendix: Linked Documents

5.1 Roadmap 2011 Spreadsheet

The Roadmap 2011 spreadsheet replaces the former gap analysis spreadsheet. It can be downloaded from http://wiki.eclipse.org/ModelingPlatform

5.2 Staffing and Budgeting Plan

The Staffing and Budgeting Plan can be be downloaded from http://wiki.eclipse.org/ModelingPlatform

6 Appendix: Potential Ways of Cooperation

The following potential ways of cooperation are not much more than ideas. The participating companies of the MPIWG should discuss these different ideas and decide upon a way for cooperation.

6.1 Preliminary Considerations

- User companies must become solution members of Eclipse Foundation to establish the IWG
- The project itself has to be done within the Eclipse Foundation or in close cooperation with the Eclipse Foundation
- We need a secure legal setup for all participants
    - User Companies (UC)
    - Solution Provider (SP)
    - Eclipse Foundation (EF)
- Project boards should consider UC, SP and EF needs
- Project manager should also have modeling background
- We need clear rules how new user companies can join the IWG
    - Contribution of new requirements
    - Funding
    - Decision making
- We need clear rules how new solution provider can join the IWG
    - Favoring of such SP that have a history of active Eclipse support
    - Avoid freeloaders without interest in Eclipse technologies

- IP process should come from Eclipse

6.2 itemis and/or Geensys as Main Contractor / Subcontractor

Continuing the current situation with itemis as main contractor and others as sub contractor. Clear legal framework for contracts will be needed. This might only be an interims solution until a better way of funding is established.

6.3 Industry Consortium

Founding of an industry consortium with a charter, committees and well defined rules for money flow. The charter should be very closely related to Eclipse charter and rules and incorporate the IWG concept. Project management should be assigned by steering committee. Templates and examples exist from European Union.

6.4 Eclipse Foundation

Eclipse IWG concept will be extended to support money flow. Dedicated project management should come from steering committee members and know the modeling field.

6.5 Funding of a New Company dedicated to EMP

Members of the MPIWG found a new company to handle the legal stuff and money flow.

6.6 Direct Contracts

Direct contracts between user companies and solution providers.