ITEA
INFORMATION TECHNOLOGY
FOR EUROPEAN ADVANCEMENT

MARTES

Model-Based Approach for Real-Time Embedded Systems development

Title:

Common evaluation criteria

Deliverable ID:	3.1
Version:	1.0
Date:	24/04/06
Editor:	Klaus Kronlöf
Status:	Final
Confidentiality:	Public

Authors:

Author	Partner	Email
Klaus Kronlöf	Nokia	klaus.kronlof@nokia.com
Kari Tiensyrjä	VTT	kari.tiensyrja@vtt.fi
Thierry Saunier	Thales TCF	Thierry.SAUNIER@fr.thalesgroup.com
Stefan Van Baelen	K.U.Leuven	Stefan.VanBaelen@cs.kuleuven.be

Document History:

Date	Version	Editor	Description
13/12/05	0.0	Klaus Kronlöf	Initial template
15/02/06	0.1	Klaus Kronlöf	First draft
01/03/06	0.12	Kari Tiensyrjä	VTT inputs to chapter 2.1
03/03/06	0.13	Kari Tiensyrjä	VTT inputs to rest of chapter 2, proposal to leave out original chapters 3 and 4, and raise the level 2 titles of chapter 2 to level 1.
27/03/06	0.12b	Thierry Saunier	TCF contribution
30/03/06	0.2	Klaus Kronlöf	Merging 0.13 and 0.12b
24/04/06	1.0	Klaus Kronlöf	Freezing the final version according to PCC decision

Filename: MARTES_D3.1_V1.0.doc

TABLE OF CONTENTS

LIST OF TABLES

Executive Summary

This document constitutes the MARTES project deliverable D3.1 “Common evaluation criteria”.

The purpose of the document is to support the assessment phase of the case studies. By having common guidelines and criteria for assessment we get more homogeneous evaluation reports. This helps providing comparisons and condensed presentations of the results.

Chapter 1 gives the motivation and background of the evaluation framework.

Chapter 2 provides guidelines for assessing the methodology in a tool independent manner.

Chapter 3 provides guidelines for assessing the tool support.

Chapter 4 defines quantitative metrics to be used in the assessment.

1 Introduction

The goal of this deliverable is to define common evaluation criteria and metrics for assessing the methodology. The evaluation criteria and metrics will serve as a framework for gathering of data and experiences from the design cases. They will also allow more systematic analysis of the results in the assessment phase.

A complete methodology is far more than a notation, process, and tools. It provides in addition information and techniques concerning is applicability such as cost estimation, software quality assurance policies and procedures, detailed role descriptions and training programs, worked examples etc…

Objectively evaluating methodologies is a difficult and complex task. This difficulty and complexity results from a number of factors, including, but not limited to:

- The use of terminology is often inconsistent. While the difference in terminology may seem academic at first glance, the appropriateness and applications of these methodologies may be significantly impacted by this.
- Many methodologies are targeted or strongly influenced by specific programming languages.

Any methodology comparison must evaluate a "snapshot" of methodologies. Any comparison must be restrictive in the information it reviews.

The evaluation guidelines sections are orientated to assess how well the methodology and tools fulfill the needs of the actual engineer that are using them to develop systems in her or his domain. These are obviously the most important aspects of the methodology. However, as the MARTES methodology is meant to be adaptable to different domains, there are also issues related to this adaptation. The evaluation criteria section addresses these aspects too.

The purpose of this document is to support the assessment and evaluation of methodology and tools in the design cases of MARTES. The design cases are different from each others, e.g. they address a bit different although related domains, and they will be executed independently, each by one or a few partners. Therefore the criteria framework is generic to be adaptable through instantiation to each design case.

The evaluation of MARTES methodology will be perform by the users whose can be classified in three categories [i]

- *Knowledge builders:* people who build knowledge to be used in multiple different MARTES MDA-based projects.
- *Knowledge facilitators*: people who assemble, combine, customize and deploy knowledge for each specific MARTES MDA-based project.
- *Knowledge users*: people who apply the knowledge built and facilitated by the other user categories, respectively.

The criteria defined for the MARTES evaluation address these three different kind of uses.

2 Methodology evaluation guidelines

This subsection concentrates on the principal properties of the methodology. The issues related to particular tool implementation are avoided as much as possible. We have divided the methodology assessment in to six main sections. It is not always easy to make a distinction between these, for example model and language are often quite intermixed, but nevertheless we feel that this division is helpful.

2.1 Design flow

Design flow criteria asses how well the MARTES design flow fit to company internal processes. As the MARTES methodology gives a general adaptable design flow framework, it is assumed that best effort has been made to adapt the flow to company and domain specific needs.

The MARTES design flow will be assess through the different steps from the specification to implementation covered by the following modeling activities [i]:

- Requirement and specification modeling
    - Abstract use case
    - Refined use case
- Application modeling
    - Functional modeling
    - Execution modeling
    - Workload modeling
- Execution platform modeling
- Allocation modeling

Implementation oriented modeling.

Table 1 exemplifies the assessment of design flow.

Table 1. Design flow assessment.

Main phases of your design flow	MARTES phases	Assessment
Phase 1	Requirements	- Match: 75% - Good points: NFP - Missing points: Validation
Phase 2	Specification	
…	…	
Phase N	Implementation-oriented modeling	

2.1.1 Activities

This criterion ranks activity by activity the degree of support/coverage the MARTES methodology gives to the activities of the company design process.

Table 2 exemplifies the assessment of activities.

Table 2. Activity assessment.

Main activities	MARTES activities	Assessment
Activity 1	Use case modeling	- Match: 75% - Good points: NFP - Missing points: Validation
Activity 2	Decomposition	
…	…	
Activity N	Performance evaluation	

2.1.2 Artifacts

In a model-based methodology, such as MARTES, the artifacts of the design process are models. This criterion ranks artifact by artifact the degree of support/coverage the MARTES methodology gives to the artifacts of the company design process.

Table 3 exemplifies the assessment of artifacts.

Table 3. Artifacts assessment.

Main artifacts	MARTES artifacts	Assessment
Artifact 1	Requirements model	- Match: 75% - Good points: NFP - Missing points: Validation
Artifact 2	Specification model	
…	…	
Artifact N	Implementation model	

2.1.3 Analysis methods

Analysis methods extract information from artifacts (models) to help engineer's decision making. The extracted information can be functional (e.g. correctness) or non-functional (e.g. power consumption). Analysis methods are normally supported by tools, but can be also manual. This criterion assesses how well the MARTES methodology provides the information that the engineer needs to do rational design decisions.

Table 4 exemplifies the assessment of analysis methods.

Table 4. Analysis methods assessment.

Analysis methods	MARTES analysis methods	Assessment

Analysis method 1	Use case validation	- Match: 75% - Good points: NFP - Missing points: Validation
Analysis method 2	Performance simulation	
…	…	
Analysis method N	Schedulability analysis	

2.1.4 Transformations

In a model-based methodology the design evolves through successive model transformations. Sometimes transformations can be automated or at least guided by tools. In a rigorous formal method the correctness (according to some useful definition) of transformations can be proved. It is desirable that the initial requirements can be traced through the transformations to the eventual implementation. This criterion assesses the degree of support that the MARTES method gives to the model transformations that are relevant in the company design process.

Table 5 exemplifies the assessment of transformation.

Table 5. Transformation assessment.

Transformation	**MARTES transformation**	**Assessment**
Transformation 1	Use case decomposition	- Match: 75% - Good points: NFP - Missing points: Validation
Transformation 2	Application refinement	
…	…	
Transformation N	Allocation mapping	

2.2 Artifact modeling capability

Artifact modeling capability criteria assesses how well the modeling approach of the MARTES methodology fits to describe the key artifacts of the company design process. In the MARTES methodology, the concepts available to the modeler are defined by a metamodel. This metamodel is normally a domain specific specialization of a more generic metamodel. It is assumed that the company method engineer (see next section) has done this specialization work and the possible difficulties of doing so are dealt in the next section. In this section we simply assume that the domain specific metamodel is available.

2.2.1 Suitability to the domain

This criterion assesses in general, how concisely, completely and easily the concepts of the domain can be modeled with the concepts of the (specialized) metamodel. Ideally there is a perfect match, so that there is a natural way to model the artifacts.

Table 6 exemplifies the assessment of domain concepts.

Table 6. Domain concepts assessment.

Domain concept	**MARTES concepts**	**Assessment**
Domain concept 1	One or more of application modeling concepts	- Match: 75% - Good points: NFP - Missing points: Validation
Domain concept 2	One or more of architecture modeling concepts	
…	…	
Domain concept N	One or more of allocation modeling concepts	

2.2.2 Application modeling

This criterion assesses how well the application modeling approach of the MARTES method applies to the modeling of the different parts of the application and their relationships. The term application refers here to the behavior of the target system. At different phases of the development process the application behavior is either required, specified or implemented. The application model contains both functional (logical) and non-functional (performance) aspects. Typically it also contains both control behavior and data streaming behavior. All these aspects and views are strongly inter-related.

Table 7 exemplifies the assessment of application modeling concepts.

Table 7. Application modeling assessment.

Application modeling aspect	**MARTES modeling aspects**	**Assessment**
Required behavior	A view of requirements model	- Match: 75% - Good points: NFP - Missing points: Validation
Specified behavior	A view of functional model	
…	…	
Implemented behavior	A view of implementation model	

2.2.3 Platform modeling

This criterion assesses how well the platform modeling approach of the MARTES method applies to the modeling of the different parts of the execution platform and their relationships. The execution platform provides the services that can be used to implement the required system behavior. Typically the execution platform consists of both software and hardware. The platform model contains both functional and non-functional aspects.

Table 8 exemplifies the assessment of platform modeling concepts.

Table 8. Platform modeling assessment.

Platform modeling aspect	MARTES modeling aspects	Assessment
Element	Component	- Match: 75% - Good points: NFP - Missing points: Validation
Architecture	HW architecture	
…	…	
Middleware	Platform architecture	

2.2.4 Allocation modeling

This criterion assesses the allocation modeling approach of the MARTES method. Allocation is a design activity that defines how the platform services are used to implement the required system behavior. Its essential part is a mapping between application model and platform model that specifies where the application components are located in the platform architecture. Typically this mapping is static but in some cases it can be dynamic.

Table 9 exemplifies the assessment of allocation modeling concepts.

Table 9. Allocation modeling assessment.

Allocation modeling aspect	MARTES modeling aspects	Assessment
	Allocation on design platform	- Match: 75% - Good points: NFP - Missing points: Validation
	Allocation on execution platform	
…	…	
Mapping	Allocation on implementation platform	

2.2.5 Constraints modeling

This criterion assesses the constraints modeling approach of the MARTES method. Design constraints are used for different purposes:

- Constraints expressed as end-user requirements or derived from them
- Constraints related to the proper usage of the execution platform
- Constraints defined by the system engineer as design choices to limit the design space.

Table 10 exemplifies the assessment of constraints modeling.

Table 10. Constraints modeling assessment.

Constraints modeling aspect	MARTES modeling aspects	Assessment
	End-user related	- Match: 75% - Good points: NFP - Missing points: Validation
	Design space reduction	
…	…	
	Platform usage	

2.3 Language

Language assessment criteria concentrate on the graphical and textual notations used in the MARTES approach as opposed to the modeling approach. In our terminology, models are written in a given language, for example UML. Theoretically one may mix languages in a single model, however we take the approach that a mixing of languages constitutes a new unified languages". Models may consist of other models, each of those expressed in their own language.

2.3.1 Expressiveness

The expressiveness criterion assesses the availability of appropriate language constructs to express the concepts of the artifact at hand (that has its underlying model). Given that there may not be a built in language construct that directly corresponds to a modeling concept, it is often necessary to represent the concept in a more complex way. If there is no obvious choice of how to express the concept, then we may end up in representing the same concept in different ways in different situations. This creates problems in human comprehension and particularly in machine interpretation. These issues have to be considered when assessing the expressiveness of the language.

Table 11 exemplifies the assessment of language expressiveness.

Table 11. Language expressiveness assessment.

Artifact concept	Language construct	Assessment
Behavior concept 1	State diagram	- Match: 75% - Good points: NFP - Missing points: Validation
Structure concept 1	Composite structure diagram	
…	…	
Implementation concept N	Communication protocol	

2.3.2 Convenience

The convenience criterion assesses the general usability aspects of the language, such as learning effort, readability, ease of comprehension, writing (or drawing) effort.

Table 12 exemplifies the assessment of language convenience.

Table 12. Language convenience assessment.

Usability aspect	Means given by language and/or support material	Assessment
Learning effort		- Match: 75% - Good points: NFP - Missing points: Validation
Readability		
…	…	
Ease of comprehension		

2.4 Support for validation and verification

This section concentrates on the validation and verification aspects of the MARTES methodology. These activities are normally supported by tools, but in order to be able to provide any tools support, the underlying models must contain the necessary information. The assessment should concentrate on the modeling aspect instead of the availability of tools. The best option of course is to have a concrete tool to be evaluated, but from methodology point of view it is enough to have an algorithm or manual procedure that can be automated.

Table 13 exemplifies the assessment of validation and verification support.

Table 13. Validation and verification support assessment.

Validation and verification aspect	Means of methodology	Assessment
	Use case simulation	- Match: 75% - Good points: NFP - Missing points: Validation
	System-wide test scenarios and data	
…	…	
	Performance simulation	

2.4.1 Logical correctness

This criteria assesses the ability to prove the logical correctness of a model or of a particular relationship between a set of models. Typically we may want to prove that a model is consistent (no internal conflicts), or that a model is a proper refinement (according to some definition of refinement) of another model, or that a model possesses some specific property (that can be expressed as another model).

Table 14 exemplifies the assessment of logical correctness.

Table 14. Logical correctness assessment.

Correctness aspect	Means of methodology	Assessment
Model consistency	Model checking ?	- Match: 75% - Good points: NFP - Missing points: Validation
Refinement	Functional simulation at various levels of abstraction	
…	…	
Property	NFP evaluation	

2.4.2 Performance evaluation

These criteria assess the ability to analyze and verify relevant non-functional properties of the modeled system. Typical properties of interest are cost, real-time performance, power consumption and memory demand. Also safety and security can be considered as non-functional properties.

Table 15 exemplifies the assessment of performance.

Table 15. Performance assessment.

Performance aspect	Means of methodology	Assessment
Real-time	Performance modeling and evaluation	- Match: 75% - Good points: NFP - Missing points: Validation
Power	Power estimation (system-level)	
…	…	
Cost	Cost model ?	

2.5 Support for design transformations

In the MARTES methodology the design evolves through successive model transformations. In each transformation we typically want to preserve some properties of the model, satisfy specific constraints, optimize specific quality measures, etc. Ideally we could have tools (or procedures that can be automated) that support all these objectives of model transformations. Normally even if we have tools, the support is partial, e.g. the tool may guarantee the preservation of given properties but optimization is left to the engineer.

Table 16 exemplifies the assessment of design transformations.

Table 16. Design transformations assessment.

Design transformation aspect	Means of methodology	Assessment
	Use case decomposition	- Match: 75% - Good points: NFP - Missing points: Validation
	Execution modeling	
…	…	
	Allocation	

2.6 Support for traceability

Traceability refers here to the ability to find out where and how a particular requirement is satisfied in the designed system. It should be possible to trace requirements at any point of development process, also with partial designs. Furthermore, due to the fact that requirements tend to change during the development, it should be possible to perform quick impact analysis and then update the requirements and make design changes in a consistent manner.

Table 11 exemplifies the assessment of support for traceability.

Table 17. Traceability support assessment.

Traceability aspect	Means of methodology	Assessment
Requirement decomposition		- Match: 75% - Good points: NFP - Missing points: Validation
…	…	
XXX tracing		

3 Tool evaluation guidelines

As opposed to the previous subsection, this subsection concentrates on the quality of tool implementations. The assessor should neglect the possible deficiencies of the methodology aspects as much as possible. For example, if a code generator generates class skeletons only while the user wants fully operational code, then one can either blame the method (the model lacks behavioral description) or the tool (it does not make use of the behavioral information of the model for code generation).

3.1 Interoperability

One expected benefit of the model-based approach is tool interoperability due to the fact that the meta-model is defined independently of tools and therefore we may have a common tool independent model repository. Tool interoperability is a direct consequence of this principle, that is, all tools should understand the same metamodel and use the same data formats to represent models. The criteria assess how well this principle works in practice.

Table 11 exemplifies the assessment of tool interoperability.

Table 18. Tool interoperability assessment.

Interoperability aspect	**Means given by tool**	**Assessment**
		- Match: 75% - Good points: NFP - Missing points: Validation
Model exchange		
…	…	

3.2 Usability

This criterion assesses the user interface of the tool and other usability aspects.

Table 19 exemplifies the assessment of tool usability.

Table 19. Tool usability assessment.

Usability aspect	**Means of tool**	**Assessment**
		- Match: 75% - Good points: NFP - Missing points: Validation
…	…	

3.3 Maturity

This criterion assesses the maturity of the tool, manifested by the absence of crashes and other erratic tool behavior.

Table 20 exemplifies the assessment of tool maturity.

Table 20. Tool maturity assessment.

Maturity aspect	Means of tool	Assessment
		- Match: 75% - Good points: NFP - Missing points: Validation
…	…	

3.4 Efficiency

This criterion assesses the efficiency of the tool, measured as simulation speed, compilation time, compactness of target code, etc.

Table 21 exemplifies the assessment of tool efficiency.

Table 21. Tool efficiency assessment.

Tool efficiency aspect	Means given by language	Assessment
Simulation speed		- Match: 75% - Good points: NFP - Missing points: Validation
Compilation time		
…	…	
Code compactness		

4 Quantitative evaluation criteria

This section defines an evaluation framework to derive quantitative information of the evaluation in order to facilitate comparing and summarizing the results of different case studies.

The criteria have been spitted in the following 6 categories:

- RTE methodology
- UML and modeling support
- Models approach
- Transformation approach
- Code generation approach
- Tool approach

4.1 RTE methodology criteria

N°	Criterion	Average pt (0.00 to 5.00)	Evaluated pt (-1 to 5)
m1	Completeness according your RTE domain	TBD	
m2	Process and notation of MARTES	TBD	
m3	Precise definition of concepts and terms	TBD	
m4	Project development role clearly defined	TBD	
m5	RTE System Development life cycle coverage (Analysis, Design, Implementation, Testing)	TBD	
m6	RTE domain applicability support (specify the dedicated context(s) assessed)	TBD	
m7	Required expertise for learning and apply the methodology	TBD	
m8	Compatibility and applicability according your current baseline	TBD	
m9	Available resources in support of the method	TBD	
m10	Innovation	TBD	
m11	Cost estimation	TBD	
m12	Measures and metrics	TBD	
m13	Support repeatability and flexibility through properties or attributes of the process (definition of the sequence steps, required inputs/outputs, involved roles, interactions with other steps, heuristics and mechanisms for traceability, verification, validation of the process)	TBD	
m14	Software quality assurance policies and procedures	TBD	

	(localization of the decisions made during the development)		
m15	Detailed role descriptions and training programs	TBD	
m16	Techniques for tailoring the method	TBD	
m17	Support for reuse (suited to creating, as well as incorporating, reusable components into its execution)	TBD	
m18	Address RTE safety critical developments that need to be certified using dedicated standards as DO178B	TBD	TBD
m19	Time development saving according your current practices	TBD	TBD

4.2 UML and modeling support criteria

N°	Criterion	Average pt (0.00 to 5.00)	Evaluated pt (-1 to 5)
u1	Expressiveness of the notations (including syntax and semantics) supported by MARTES	TBD	
u2	Support static concepts	TBD	
u3	Support dynamic concepts	TBD	
u4	Support partitioning mechanism	TBD	
u5	Support distributed environment	TBD	
u6	UML2 support	TBD	
u7	Support for creation, customization and extension of UML profile	TBD	
u8	Non functional requirements and properties support	TBD	
u9	Support creation and evaluation of model constraints	TBD	
u10	Object Constraint Language (OCL) support	TBD	
u11	Import and/or export of existing domain facilities models using some standard like XMI format.	TBD	
u12	Provide templates for existing domain facilities	TBD	
u13	Provide extension support for provided domain facility templates and/or imported models.	TBD	
u14	Support for one or more target platform	TBD	
u15	Support debugging of models	TBD	
u16	Support of legacy system.	TBD	
u17	Support validation and verification	TBD	

4.3 Models approach criteria

N°	Criterion	Average pt (0.00 to 5.00)	Evaluated pt (-1 to 5)
o1	Platform Independent Model (PIM) support	TBD	TBD
o2	Platform Specific Model (PSM) support	TBD	TBD
o3	Support for creation, customization and extension of Meta object Facility (MOF)	TBD	TBD
o4	Developed models should be MOF compliant	TBD	TBD
o5	Support specific language/library for model navigation and construction	TBD	TBD
o6	Support for XML Metadata Interchange (XMI) and accordingly import and export of UML and MOF using standard XMI format.	TBD	
o7	Support for design and/or development iteration of models *(Here iteration means if you make changes on certain level it should kept there. For example, you have developed some PIM and have generated based PSM and then have generated code. In this scenario if, you have done some manual/visual changes in that generate PSM and Code level. These changes should remain there accordingly or depending upon their effect in the next iteration when you have made changes at CIM or PIM level.)*	TBD	
o8	Support of reverse engineering or legacy system. (Here reverse engineering or legacy system means, you have some existing system developed and you want to generated PSM from this legacy and go back to PIM and/or PSM).	TBD	
o9	Support Martes methodology modeling process	TBD	

4.4 Transformation approach criteria

N°	Criterion	Average pt (0.00 to 5.00)	Evaluated pt (-1 to 5)
f1	Support for Mapping Language Portability	TBD	
f2	Query Views Transformation (QVT) support	TBD	
f3	Transformation from PIM to PSM to Code	TBD	
f4	Transformation from Code to PSM to PIM	TBD	
f5	Direct transformation from PIM to Code	TBD	
f6	Direct transformation from Code to PIM	TBD	
f7	Transformation from PSM to PSM Bridge to PSM	TBD	

f8	Transformation based on Meta-model	TBD	
f9	Transformation based on Models	TBD	
f10	Transformation based on UML Profile	TBD	
f11	Transformation based on Marking.	TBD	
f12	Record of Transformations. (All transformations are recorded/saved in some readable format such as XML)	TBD	
f13	Support debugging of the transformation process	TBD	
f14	Traceability of Transformations. (For example, one is able to find which element in PIM is transformed to other model).	TBD	
f15	Support for merging 2 or more PIM into 1	TBD	
f16	Support for merging 2 or more PSM into 1	TBD	

4.5 Code generation approach criteria

N°	Criterion	Average pt (0.00 to 5.00)	Evaluated pt (-1 to 5)
I1	Expressiveness	TBD	
I2	Convenience	TBD	
I3	Support for code templates like, user can define his own coding conventions, etc.	TBD	
I4	Quality of the code generation (readability, complexity, performance, respect domain coding rules…)	TBD	

4.6 Tool approach criteria

N°	Criterion	Average pt (0.00 to 5.00)	Evaluated pt (-1 to 5)
t1	Available on several operating systems	TBD	
t2	Support concurrent multi-user design	TBD	
t3	Support architectural exploration	TBD	
t4	Provides role based MDA tool environment roles such as builder, facilitator, user.	TBD	
t5	Support for integration of version control system	TBD	
t6	Support for visual comparison of existing and developed code	TBD	
t7	Support for visual comparison of existing and developed models	TBD	

t8	Support for execution of generated system	TBD	
t9	Support for generated system deployment	TBD	
t10	Support interoperability with other MDA based tool	TBD	
t11	Support customization for specific domain	TBD	
t12	Support requirement traceability	TBD	
t13	Support documentation generation	TBD	
t14	Usability	TBD	
t15	Efficiency	TBD	
t16	Maturity	TBD	

5 Comparison

The first important part for the evaluation criteria is the identification of a set of important features which should be supported by MDA Martes methodology.

The average rank for each criterium will be calculated by grading importance level according to the contributors of this deliverable subjective point of view.

Based on the assessment criteria proposed by [ii], one has to devise the scale of -1 to 5.

Where each scale from -1 to 5 represents a level of support for corresponding selected feature. Following is the scale table proposed by [ii] which we are going to use for evaluating the conformance or presence of features that we had identified in our Evaluation Criteria

Generic scale point	Decision of scale point	Scale point
Makes things worse/causes confusion	The way the feature is implemented makes it difficult to use and/or encouraged incorrect use of the feature	-1
No support	The feature is not supported nor referred to in the user manual	0
Little support	The feature is supported indirectly, for example by the use of other tool features in non standard combinations	1
Some support	The feature appears explicitly in the feature list of the tools	2
Strong support	The feature appears explicitly in the feature list of the tools and user manual. All aspects of the feature are covered but use of the feature depends on the expertise of the user.	3
Very strong support	The feature appears explicitly in the feature list of the tools and user manual. All aspects of the feature are covered and the tool provides tailored dialogue boxes to assist the user.	4
Full support	The feature appears explicitly in the feature list of the tools and user manual. All aspects of the feature are covered and the tool provides user scenarios to assist the user such as “Wizards”.	5

Figure 1 : Scale for assessment

There is a simple formula taken for calculating or evaluating the points for certain feature for selected item evaluated.

Criterium point = Average point for each criterium * user evaluated rank

Where

Criterium point is the final calculated point of each feature.

Average point for each criterium is the average point of the feature in above feature list.

User evaluated rank, this is the rank being evaluated or marked after evaluating selected feature by the user.

6 Glossary

- ***Abstract Platform***
  An abstract platform defines an acceptable or, to some extent, ideal platform from an application developer's point of view; it is an abstraction of infrastructure characteristics assumed for models of an application at some point of (the platform-independent phase of) the design process.
- ***Horizontal Transformation***
  A transformation in which both the source and the target artifacts are at the same level of abstraction. In practice this probably means that the source and target metamodel are the same.
- ***Methodology***
  The union of methods, notations, guidelines, strategies, processes for a specific application domain.
- ***Model Driven Architecture (MDA)***
  A specific approach toward model driven development (MDD) invented by the OMG. The focus of MDA is primarily the provision of standard specifications for useful technologies in the MDD space. Examples are the UML (modeling), MOF (metamodeling), transformations (QVT), platform specific annotations (set of UML profiles), etc.
- ***Model Driven Development (MDD)***
  A software development paradigm that promotes the use of models at different levels of abstraction and transformations between them to derive a concrete application implementation. A specific interpretation of this notion is worked out by OMG's MDA.
- ***MDD Infrastructure***
  The collection of metamodels, platforms, transformations, languages, tools, etc., organized in a structured way, that support the development of a specific application or group of related applications (product line).
- ***Platform-Independent Model (PIM)***
  The MDA model that defines an application independent of a specific platform. The PIM is applied to a more detailed model using a transformation mechanism.
- ***Platform-Specific Model (PSM)***
  The MDA model that includes elements from the implementation platform, often made more precise through the use of relevant stereotypes.
- ***Process***
  Definition of order of tasks and activities to be performed in order to obtain certain specific results. A process is usually divided in a number of (semi-ordered) phases.
- ***Process Activity***
  A concrete description of low level actions that need to be executed within a process phase.
- ***Process Phase***
  A distinct part within a process, consisting of several (semi-ordered) activities.
- ***Technology Domain***
  The domain in which we use the vocabulary of a certain technology platform which

offers standard solutions for some problems. This is very related to execution platforms such as J2EE or .NET but also SQL which represents a relational storage domain.

- ***Transformation***
  An model operation that takes one or more models as input and returns one or more models as output. The operation maps elements from the source model elements to the target model elements.
- ***Transformation Chain***
  If a sequence more than one transformations applied to an input model, the configuration of these transformations - including their sequence, possible branches and other information - is called transformation chain.
- ***Vertical Transformation***
  A transformation in which the source elements are at a different level of abstraction then the target elements. In practice this usually means that the source metamodel is different from the target metamodel.

7 References

[i] ITEA-MARTES, D1.1 Current limitations of best practices v1.0, 31 march 2006

[ii] University of Keele, A method for evaluating Software Engineering methods and tools (1996)