# MDD Maturity Model: A Roadmap for Introducing Model-Driven Development

Erkuden Rios[1], Teodora Bozheva[1],
Aitor Bediaga[1], and Nathalie Guilloreau[2]

[1] European Software Institute, Parque Tecnológico de Zamudio, #204
E-48170 Zamudio Spain
{Erkuden.Rios, Teodora.Bozheva, Aitor.Bediaga}@esi.es
http://www.esi.es
[2] Thales Research & Technology, RD128
E-91767 Palaiseau cedex France
nathalie.guilloreau@thalesgroup.com
http://www.thalesgroup.com

**Abstract.** Experience reports show that MDD reduces time-to-market and increases productivity by means of platform independent business logic modelling and automation. Achieving these two concepts in the organisation is not a one step process. This paper explains the MDD Maturity Model developed to drive this task in a structured way. The MDD Maturity Model establishes five capability levels towards the progressive adoption of MDD within an organisation. Each level describes a coherent set of engineering, management and support practices involved in the MDD approach, and characterizes the MDD artefacts, called MDD elements, used in or resulted from those practices. The paper presents also the validation process that the model will undergo in two large organisations and two SMEs.

## 1 Introduction

Several examples can be found of satisfactory MDD introduction in organisations, such as Interactive Objects' report on MDA experimentation in DainmlerChrysler TSS and M1 Global's own case study report, both available at Object Management Group's (OMG) MDA web site (www.omg.org/mda).

As seen in experiences of the like, successfully introducing MDD methods and tools in a project is not simple, and obviously deploying them throughout the organisation is much more complex because it implies serious changes in the organisation's culture and processes: start treating models as first class citizens (which means keeping them updated and on-track), adapt the roles, provide staff with the necessary tooling and methodological training, and so on.

Maximising the benefits of MDD for time-to-market reduction and productivity increment is achieved through two key factors: abstracting from platform specificities when modelling business logic and exploiting automation possibilities.

A. Rensink and J. Warmer (Eds.): ECMDA-FA 2006, LNCS 4066, pp. 78 – 89, 2006.

In this paper we explain the MDD Maturity Model developed within the MODELWARE project[1] aimed to help organisations in the MDD approach adoption, until the whole process automation is reached, and organisation acquires the sufficient capability for business knowledge capitalisation in reusable models.

The Model has been developed to be used as reference model for identifying and appraising the level of maturity of a given organisation with respect to MDD technology implementation. The validation of the model will be done through using the model in assessments of MDD implementations by different companies.

The remainder of this paper is organised as follows. The next section explains the concepts used in the MDD Maturity Model, and Section 3 explains the Model itself. Section 4 summarises its major contributions for the industry and Section 5 describes the validation process the model will undergo. In section 6 we deal with some related works and finally, we present our conclusions and future work.

## 2 The MDD Maturity Model Concepts

The MDD Maturity Model consists of five *maturity levels*. The maturity levels provide a general characterization of the organisations with respect to the degree of adoption and implementation of MDD; this means that each maturity level indicates a step forward in the MDD improvement path of the organisation. For each maturity level goals associated to both MDD practices and MDD elements status are defined.

### 2.1 MDD Practices

MDD practices describe only activities specific for the model-driven development and typical practices in traditional software development are deliberately excluded from this Model.

Three categories of MDD practices are defined in the MDD Maturity Model:

- *Engineering practices* (ENG) cover development activities in the model-driven software engineering discipline.
- *Project management practices* (PJM) address activities that are directly related to management decisions absolutely necessary to setup and manage an MDD project. The typical practices such as planning a project, milestone definition and resource assignment are not considered.
- *Support practices* (SUP) cover activities that support the implementation of the engineering and the project management practices.

### 2.2 MDD Elements

MDD elements are the basic artefacts used in the MDD technology such as models, transformations, MDD tools and so on. The following MDD elements are identified:

[1] The work presented here has been developed within the MODELWARE project. MODELWARE is a project co-funded by the European Commission under the "Information Society Technologies" Sixth Framework Programme (2002-2006). Information included in this document reflects only the author's views. The European Community is not liable for any use that may be made of the information contained herein.

**Table 1.** MDD Elements and associated attributes

| MDD Element: | Attribute: | Attribute description: |
|---|---|---|
| **Models** | Model purpose | The extent to which the model is defined according to established organisational policies and standards. |
| | Adherence to organisational policies and standards | The objective for which the model is defined. |
| | Scope of the model | The extent of the matters defined in the model. |
| | Integration degree | The extent to which the model is integrated in the development process, if the model is defined in isolation or it is linked to other MDD elements by means of formal and consistent relationships. |
| | Verification degree | To which extent the verification activities are focused on this model. |
| | Traceability depth | Extent of details addressing the traceability of the model to other MDD elements. |
| | Simulability | Ability of being simulated by means of a model simulator. |
| | Executability | Ability of being executed by means of a model executor or virtual machine. |
| **Transformations and code generation mechanisms** | Transformation type | Horizontal (generation of another model view at same level of abstraction) or Vertical (generation of another model or artefact at another level of abstraction). |
| | Round-trip engineering support | Degree of support for round-trip engineering (forward and backward transformation). The implementation of this aspect supports synchronisation among models. |
| | Platform dependency | Degree of dependency with the specific target platform of the system. |
| **Tools** | Integration facility | Capability of the tool to be integrated with other tools supporting the MDD process. |
| **Documentation** | Automation extent | Average ratio of automatically generated to manually written part in documentation. |

- *Models*: A model represents an abstraction (simplification) of something in the real world and captures its essential characteristics. The following types of models are distinguished:
  - *Domain metamodel*: is the metamodel or language that captures the abstract structure of the business domain identifying fundamental domain entity types and the relationships between them.
  - *Architecture-centric metamodel*: is the metamodel that captures the concepts of the technical platform.

  - *Domain model*: is the model that defines how a business works without reference to software systems, similarly to OMG's Computation Independent Model.
  - *Business model*: is the model that resolves business requirements through purely problem-space terms and it does not include platform specific concepts, as the OMG's Platform Independent Model.
  - *Technical model*: is a solution model that resolves both functional and non-functional requirements through the use of platform specific concepts. This model is equivalent to the OMG's Platform Specific Model.
  - *Code*: is the final asset in the development, which can be considered as a model because it conforms to a specific metamodel, the programming language.
- *Model transformations and code generation mechanisms*: are mechanisms for converting a model to another model of the same system. Model to model, Model to text and Model to code transformations are examples of this MDD element type.
- *Modelling tools*: are tools that are used in modelling activities, e.g. model editors, model simulators, model executors, model repositories, transformation editors, transformation repositories, transformers…
- *Documentation*: is the set of text documents which describe all the development process and/or the assets generated and, thus, is linked to other MDD elements.

While MDD practices do not, MDD elements do have *MDD attributes* associated to them. Each attribute describes an essential characteristic of the MDD element. The next table summarizes the attributes identified in the Model.

The maturity level of an organisation is given by the assessment of two factors:

- whether the MDD practices and MDD elements corresponding to that maturity level exist or not and
- whether those MDD elements' attributes take the appropriate values corresponding to that maturity level.

## 3 The MDD Maturity Model

One of the major requirements of the MDD Maturity Model developed inside MODELWARE project is to be compliant with the Capability Maturity Model® Integration (CMMI®), which is a recognised and widely spread model, implemented in lots of software intensive organisations.

One approach to developing the MDD Maturity Model is to define how the MDD activities amplify the CMMI® specific practices. This approach could be useful for organisations that have experience and knowledge in applying CMMI®. However, organisations interested in adopting MDD without implementing CMMI® will get little benefit from an MDD Maturity Model represented as an amplification of CMMI®. Besides, lots of Small and Medium Size (SME) companies do not apply CMMI®, yet are interested in increasing the effectiveness of their software engineering processes by means of MDD. The MDD Maturity Model is developed as an independent model, which, however, complements CMMI.

Additionally, the MDD Maturity Model is aligned with the model developed within the FAMILIES (IP02009) project, with respect to the domain capitalization dimension, because the goals of the two upper levels in the MDD Maturity Model fit very well in it.

To define the MDD Maturity Model, literature and early adopters' MDD processes has been studied and the following approach was adopted:

- Analysis of the MDD practices and grouping them in levels representing different degree of profundity of the implementation of MDD
- Analysis of technical means: how they can be characterised, what are the different possible extents for using and deploying these means in a development process
- Study of the dependencies between the MDD practices and the technical means
- Identification of discrete levels of the MDD adoption that combine MDD practices and relevant technical means.

As shown in Figure 1, the MDD Maturity Model defines five maturity levels distributed bottom up, from less mature to more mature MDD adoption. The lower level MDD practices and elements are a basis for the implementation of the activities on the upper levels.

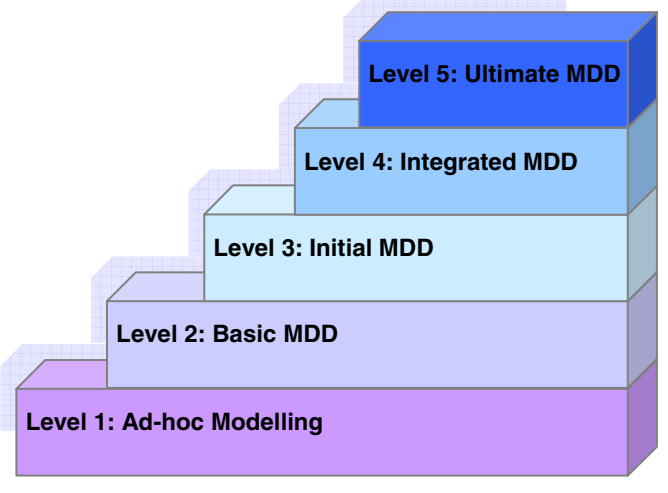


**Fig. 1.** MDD Maturity Model levels

### 3.1 Maturity Level 1: Ad-Hoc Modelling

The Ad-hoc modelling level corresponds to situations where modelling practices are sporadically used or not used at all in the organisation. This means that the organisation is performing traditional software development, and individuals may use some models for their own help, but no policy or common understanding applies to those scarce models. Obviously, the organization has no specific goals on modelling activities or artefacts.

### 3.2 Maturity Level 2: Basic MDD

In this level of maturity, the organisation is more mature in modelling and in each project developed in the organisation a Technical model is created with which the final code and system documentation have to be in line. In this level, the Technical model combines business and technical aspects of the system to be developed, with no distinction between them.

The final code and documentation shall comply with the system specification modelled. This alignment is done by means of basic automatic code generation and documentation generation mechanisms which generate (parts of) them from the Technical model.

In Level 2, the fact that models are used for guiding implementation and production of documentation is an organisational premise and not an individual initiative. In the projects, it is necessary to take decisions upon the modelling tools and techniques that will be used in the development, in accordance with project objectives.

The next table defines the goals in this level and the MDD practices aimed to achieve them.

**Table 2.** MDD Maturity Level 2 goals and practices

| **Goals:** | |
|---|---|
| Goal 1 | Develop technical model and use it to build up software |
| Goal 2 | Include all business and technical requirements in models |
| Goal 3 | Select MDD tools aligned to project objectives |
| **MDD Practices:** | |
| **Engineering** | ENG 1 Identify modelling techniques<br>ENG 2 Define Technical model<br>ENG 3 Generate code from the Technical model<br>ENG 4 Generate documentation from the Technical model<br>ENG 5 Complete code to comply with all req. |
| **Project Management** | PJM 1 Decide upon modelling tools |
| **Support** | N/A |

Figure 2 shows the key elements in the MDD maturity level 2.

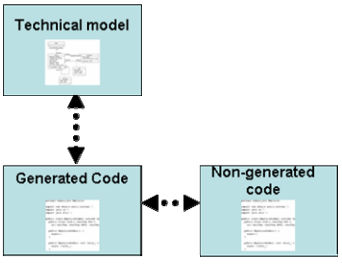


**Fig. 2.** MDD elements in MDD Maturity Level 2

Note that in all figures, a thick dashed arrow stands for “manual or automatic transformation”, whereas a thick continue arrow means “automatic transformation”.

### 3.3 Maturity Level 3: Initial MDD

The organisation starts developing systems in a more model-driven approach when, besides aligning the code and the models, it develops business models which address the business logic of the system separately from the technical models which cover the technical requirements. This is done for capitalising the business knowledge over all the projects.

Business models are then manually converted to technical models, but these technical models are represented by means of a tool and are converted to code automatically. The Business models can be directly converted to code also, which means that the Technical model with platform specifics resides implicit in this direct transformation.

In addition to business logic and platform specifics differentiation, in this level of maturity, the models are exchanged between different stakeholders for communication, which implies the need of models are checked with respect to well-formedness rules, and metrics on modelling activities are consistently defined, collected and analysed.

The next table defines the goals in this level and the MDD practices aimed to achieve them.

**Table 3.** MDD Maturity Level 3 goals and practices

| **Goals:** | |
|---|---|
| Goal 1 | Separate business and technical aspects in MDD elements |
| Goal 2 | Define rules for modelling linked to organisation’s strategy |
| Goal 3 | Exchange system knowledge with other stakeholders through models |
| **MDD Practices:** | |
| **Engineering** | ENG 6 Define Business model<br>ENG 7 Define transformations from Technical model to text<br>ENG 8 Separate generated from non-generated code<br>ENG 9 Check models |
| **Project Management** | PJM 2 Define MDD-project workflow<br>PJM 3 Decide upon coverage of modelling activities |
| **Support** | SUP 1 Establish and maintain repositories for models and transformations<br>SUP 2 Define, collect and analyze measures with respect to the modelling activities |

The next figure depicts the MDD elements of the level 3 and their relationships.

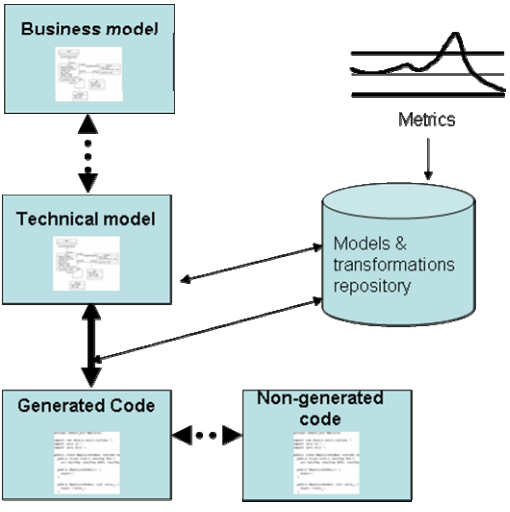


**Fig. 3.** MDD elements in MDD Maturity Level 3

### 3.4 Maturity Level 4: Integrated MDD

The organisation begins integrating the models when domain modelling is performed. This means that the domain concepts are represented by means of a domain model. Business models are derived from the domain models and are developed by means of a tool. Then, they are automatically transformed to technical models and these technical models into code. Domain, business and technical concepts are separated.

In this maturity level, two types of technical models are developed: the ones that model the core infrastructure shared by all products in a product family, and the technical models for a specific application development. This ensures reusability of infrastructure models.

At Level 4, the organisations are more mature in modelling and they simulate the models created with a tool, in order to verify them for early correcting possible design errors.

The next table defines the goals in this level and the MDD practices aimed to achieve them.

Figure 4 shows the MDD elements involved in the level 4 and the relationships among them.

### 3.5 Maturity Level 5: Ultimate MDD

To achieve a complete MDD adoption and reap its benefits, there is a need to have a system family engineering mindset, which means to have a common set of MDD assets (transformations, domain models, metamodels,...) that are reusable organisation-wide. Therefore, the ultimate maturity level is reached when the transformations between all the models are made automatically and models are fully integrated between them and with code. Executable models are developed so the focus of the organisation efforts is on the models and not on code programming. The whole life cycle becomes model-driven.

**Table 4.** MDD Maturity Level 4 goals and practices

| **Goals:** | |
|---|---|
| Goal 1 | Separate domain, business and technical aspects in MDD elements |
| Goal 2 | Ensure efficient modelling performance |
| Goal 3 | Share integrated development environment |
| **MDD Practices:** | |
| **Engineering** | ENG 10 Define architecture centric metamodel<br>ENG 11 Define domain model<br>ENG 12 Define transformations from Business model to Technical model<br>ENG 13 Simulate models<br>ENG 14 Separate the technical models of the product and the system family infrastructure |
| **Project Management** | PJM 4 Manage common infrastructure development. |
| **Support** | N/A |

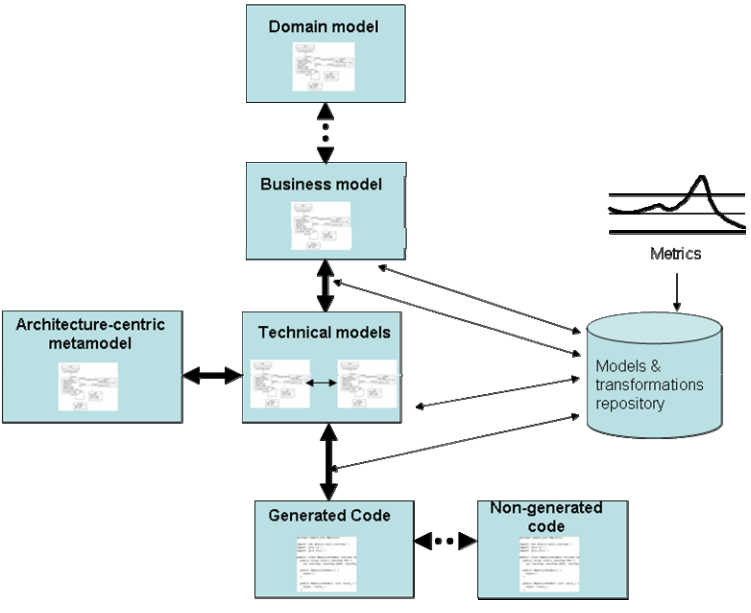


**Fig. 4.** MDD elements in MDD Maturity Level 4

Hence, the main characteristic of the ultimate MDD level is that the entire organisation's know-how is capitalised in models and transformations. The domain engineering practices are put in place and Domain Specific Languages (DSL) are created in order to make strategic assets reusable. Even the system verification and validation (V&V) knowledge is stored in models that are used for V&V of the implementation.

The next table defines the goals in this level and the MDD practices aimed to achieve them.

**Table 5.** MDD Maturity Level 5 goals and practices

| **Goals:** | |
|---|---|
| Goal 1 | Ensure complete model-centric development |
| Goal 2 | Ensure organisation's knowledge is capitalised in models and transformations |
| **MDD Practices:** | |
| **Engineering** | ENG 15 Define domain specific languages<br>ENG 16 Continuously improve and validate the meta-models<br>ENG 17 Define transformations from Domain model to Business model<br>ENG 18 Model-based V&V |
| **Project Management** | PJM 5 Establish and maintain strategic MDD elements |
| **Support** | N/A |

Figure 5 shows the MDD elements involved in the level 5.

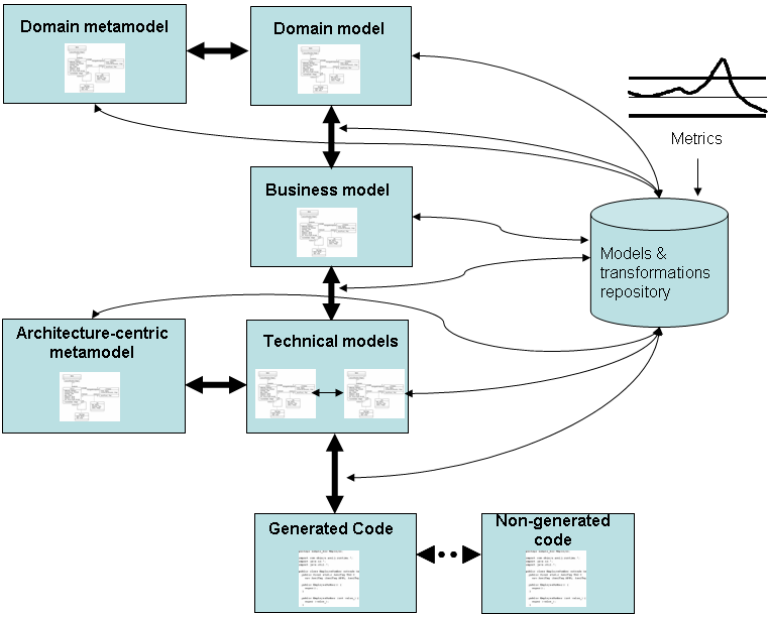


**Fig. 5.** MDD elements in MDD Maturity Level 5

## 4 MDD Maturity Model Benefits for the Industry

The main benefits that the MDD Maturity Model can offer to the industry are:

- Provides understanding of the steps towards a complete and efficient MDD adoption.

- Makes easier to further improve MDD practices and work products in the organisations.
- Establishes a common integrated vision of all MDD dimensions to improve in the organisation.
- Facilitates to accomplish the cultural and organisational changes that MDD implies simultaneously with the learning of a common language on modelling activities understood by all process participants.

The MDD Maturity Model complements other maturity models, such as the CMMI®, allowing the adoption of MDD specific practices within the CMMI® improvement initiative.

The MDD Maturity Model is the first step for building the standardized framework for categorizing the organisations' capabilities on MDD, for either internally identify or externally claim their maturity level.

## 5 Validation Process

The MDD Maturity Model described above is going to be used as the reference model for performing assessments of companies or project teams with respect to MDD. In particular, the model will be used to assess the MDD implementation by four leading companies: Thales ATM (France), France Telecom (France), WM-data (Estonia) and Enabler (Portugal). The first two are well known large businesses, WM-data is a SME branch of the leading supplier of design and IT services in the Nordic regions and Enabler is the SME branch of an international IT solution provider for retailing. These four companies are industrial partners in MODELWARE and the assessments' work is also part of the project results.

The assessment consists in rating the capability in each of the MDD practices and evaluating the status of the MDD elements in the Model, and therefore highlighting both strengths and areas candidate for improvement.

The MDD Maturity Model will be valid if it enables to distinctively characterize the maturity level of each organisation and if it helps organisations in effectively implementing MDD and improving its weak areas in MDD. Besides, the terminology used in the model shall be understandable for all these companies and it should embrace all the key modelling practices experimented by them.

After the validation process a refined version of the MDD Maturity Model will be issued. The improvement brought will mainly consist in refining the goals and practices in each of the levels and integrate them with appropriate MDD metrics to collect in each case.

## 6 Related Work

Assessing the capability of an organisation with regards to MDD technology is a relatively new subject, with limited material available and experimented in the MDD community.

Some partial attempts have been made in MDD maturity degrees definition, which focus on specific aspects of MDD. This is the case of Kleppe and Warmer's

Modelling Maturity Levels [2] or the IBM approach [3], which uses some form of a MDD technological capability model as commercial support for their proprietary tools. Whereas neither of these models has formal specification of the MDD practices and assets inside each maturity level, nor is validated by the industry yet, our Model makes a formal definition of both MDD practices and elements for unambiguously characterising the maturity levels. Besides, our model will undergo a validation process by the industry in near future.

## 7 Conclusion and Future Work

The MDD Maturity Model described has been developed inside the MODELWARE project to complement the existing models for quality and process improvements by putting the focus on how to execute software engineering activities applying the MDD technology.

The Model describes five maturity levels in the roadmap for improving MDD practices and MDD artefacts, from the lowest level (Ad-hoc modelling level) to the highest level-5 (Ultimate MDD level). Each level describes a consistent set of engineering, management and support practices within the MDD approach. Additionally, it provides a characterization of the MDD elements created or used at each level.

The MDD Maturity Model is the tool for organisations to establish the correct roadmap for the adoption of MDD. It provides them a means for identifying their strengths and weaknesses with respect to MDD. Therefore, the MDD Maturity Model serves to support industry in improving their MDD development processes, technology and organisation. A final, refined iteration of the model will follow after the model is validated in the industrial partners in MODELWARE in June 2006.

## Acknowledgments

We would like to thank the contributions to the development and refinement of the MDD Maturity Model: as co-developers Veronique Normand (Thales Research & Technology) and Jason Xabier Mansell (European Software Institute); and as main reviewers Asier Azaceta (European Software Institute) and Robert Pastor (Thales Research & Technology).

## References

1. Bettin Jorn, Patterns for Model Driven Software Development, (2004),
2. Kleppe, Aneke, Warmer, Jos: Getting started with Modeling Maturity Levels, (2004), http://www.devx.com/enterprise/Article/26664
3. A Roadmap for Agile MDA, http://www.agilemodeling.com/essays/agileMDA.htm
4. Family Evaluation Framework overview & introduction, Families project http://www.esi.es/en/ Projects/Families/
5. CMU/SEI-2002-TR-012 Capability Maturity Model Integration ver. 1.1
6. Kleppe, Aneke, Warmer, Jos, Bast, Wim: MDA Explained, ISBN 032119442X (2004)
7. Bézivin, Jean: In Search of a Basic Principle for Model Driven Engineering. Upgrade. Vol V, No. 2. (2004)