MODA-TEL

(Deliverable D3.add1)

MODA-TEL Model-Driven Methodology

Editor: Anastasius Gavras, Eurescom GmbH

Suggested readers

Information technology systems architects, senior information technology scientists, research staff especially in the telecommunications domain.

Research and development engineers involved in the OMG standardisation process around all issues relating to Model Driven Architecture

Abstract

Following the recommendation of the project technical reviewers, Mr. Rick Reed and Mr. Svein Hallsteinsen, to deliver a more prescriptive methodology as guidance to a development team that faces Model Drivel Architecture this document is a distillation of the information available in deliverables D3.1 and D3.2 of the MODA-TEL project. The information contained herein is strongly based on the contents of these deliverables that have been made publicly available through http://www.modatel.org It is unavoidable for the reader must consult the above mentioned deliverables.

The outlined methodology is effectively partitioning the MDA approach into several phases; each phase being partitioned in several activities. Roles and products for each activity are identified based on the terminology of the software process engineering meta-model (SPEM).

The MODA-TEL methodology is being applied inside the project for the development of the use cases, with the aim to verify the applicability and correctness of the methodology and the activities.

© MODA-TEL Consortium 2003

Disclaimer

This document contains material, which is the copyright of certain MODA-TEL consortium parties, and may not be reproduced or copied without permission.

All MODA-TEL consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the MODA-TEL consortium as a whole, nor a certain party of the MODA-TEL consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

 © MODA-TEL Consortium 2003

List of Authors

Name	Company	e-mail
Mariano Belaunde	France Télécom R&D	Mariano.belaunde@rd.francetelecom.com
Ronald Steinhau	Interactive Objects	Ronald.Steinhau@io-software.com
João Paulo Almeida	Univ Twente	almeida@cs.utwente.nl
Anastasius Gavras	Eurescom	gavras@eurescom.de

Project internal review by:

Name	Company	e-mail
Philippe Desfray	SOFTEAM	phd@softeam.fr
Diego Rapela	Interactive Objects	diego.rapela@io-software.com
Luís Ferreira Pires	Univ Twente	pires@cs.utwente.nl

© MODA-TEL Consortium 2003

Table of Contents

 © MODA-TEL Consortium 2003

List of Figures

© MODA-TEL Consortium 2003

1 Introduction

While the MDA describes methods and techniques for model driven software development, it does not define a methodology as *a body of interrelated methods and rules*. Furthermore the MDA methods and techniques are not explicitly related to identifiable activities within software development processes.

The purpose of this document is to outline the model driven methodology adopted in the MODA-TEL project [1]. This methodology provides definitions of the used concepts, the identification of phases and individual activities, and their interrelationships in a model driven software development process.

The proposed methodology herein can be seen as a framework for combining established software development processes with MDA and thus customise a certain software engineering process in an organisation. Moreover, this methodology can be adopted by software development organisations to define specific software engineering processes.

1.1 Concepts Definition

In this section, we briefly define important concepts that are used in the definition of the MODA-TEL methodology. These include concepts that are adopted in MDA documentation, such as, e.g., model, platform and model transformation, and concepts that are defined for the scope of this document. The concepts that are used to refer to software development processes and their elements are aligned with the definitions in the OMG Software Process Engineering Meta-model (SPEM) [2]. Several of these concepts have been identified and assessed in the MODA-TEL deliverables [5], [6], [7].

- The notion of *model* is central in MDA and in this methodology. A model of a system is a description or specification of that system and its environment for some certain purpose [3]. Examples of models are UML class diagrams, SDL specifications, IDL interfaces and Java source code.
- Models are expressed in a suitable *modelling language*. A modelling language may be defined by using meta-models that specify the abstract syntax of all possible models, or it may be defined by specializing an existing language through a profile.
- A *meta-model* is a model of models [3]. When a model *B* is used to describe a model *A*, *B* is said to be the meta-model of *A*.
- The term *profile* is used here to denote a specialization of a modelling language in general, and, in particular, a UML profile [4].
- The notion of platform also has a prominent role in this methodology. A *platform* is defined as "a set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented" [3]. A common pattern of application of MDA in a design trajectory is to define platform-independent models (PIMs) of a system, and to apply (parameterised) transformations to these PIMs to obtain platform-specific models (PSMs) of the system.
- *Model transformation* is the process of converting a source model into a target model [3]. Model transformation is often prescribed in a model, which we call model transformation specification.
- In our methodology, we also define the term *abstract platform* [7]. PIMs rely on an abstract platform in an analogous way as PSMs rely on a platform. An abstract platform is defined by the characteristics of the possible target concrete platforms that are relevant at a certain platform-independent level, and may be implied by the modelling languages used for platform-independent modelling.

In addition to the aforementioned concepts, we must also consider the concepts we use to discuss the procedures that are executed along a model-driven software engineering trajectory:

- A software engineering *process* is defined in term of activities, process roles, phases, work products, and associated guidance.

 © MODA-TEL Consortium 2003

- *Activities* are performed by *process roles*, and are the main elements of work.
- A *work product* is a description of a piece of information or physical entity produced or used by the activities of the software engineering process. Examples of work products include models, plans, code, executables, documents, databases, and so on. A *deliverable* is a formal work product of the process.
- A *phase* is a work definition that ultimately includes activities with a precondition that defines the phase's entry criteria and its goal. Phases are defined so that there is minimal overlap of their activities in time.
- All activities are performed with the use of some associated guidance. *Guidance* consists of techniques, modelling languages, guidelines, procedures, standards, templates of work products, examples of work products, definitions, etc.

In addition to the aforementioned terms defined in SPEM to describe work performed in a process, we define the following specializations of activities: identification, selection and specification of work products:

- The *identification of a work product* is the activity of recognizing the need for the work product.
- The *selection of a work product* is the activity of choosing the work product from an existing set of work products.
- The *specification of a work product* is the definition of the work product in detail, possibly using formalisms. The specification of a work product may also entail the specialization or refinement of previously specified work products.

Tools have an important role in model-driven software development. *Software development tools* are the software instruments used by process roles in performing activities of a software development process. Since tools are not used in isolation, one may also refer to the existence of a (MDA) *tool chain.*

2 Overview of methodology

2.1 Population of MDA users

In an organisation applying MDA technologies one can typically find three categories of users, partitioned according to the available or necessary expertise. Figure 1 illustrates the three groups of the population of MDA users.

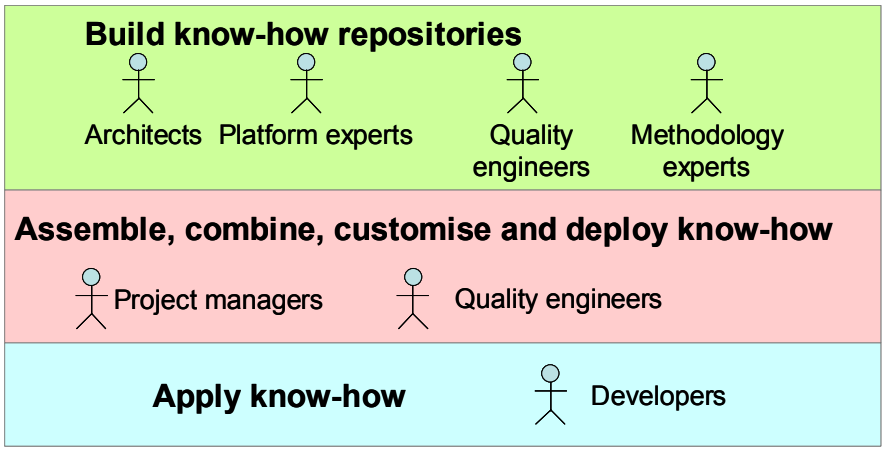

Figure 1 Population of MDA users

- The first group accounts typically for the expertise to *build know-how repositories*. This group includes systems architects, platform experts, quality engineers and methodology experts. We estimate that this group amounts approximately 5% of the total MDA users population.
- The second group accounts typically for the expertise to *assemble, combine, customise and deploy know-how*. This group includes project managers and quality engineers. We estimate that this group amounts approximately 5% of the total MDA users population.

© MODA-TEL Consortium 2003

- The third group accounts typically for the experts *applying know-how*, and includes the software developers. We estimate that this group amounts approximately 90% of the total MDA users population.

2.2 Phases

The distinction between preparation activities and execution activities is essential to manage a project applying a Model Driven Development (MDD) approach. As depicted in Figure 1, different roles are associated with different skills and different tools. Preparation activities are typically those that will structure and plan the work and, in the meantime, will make possible reuse of know-how. These activities are normally performed by people from the first (upper) group in Figure 1. Indeed, preparation activities are to be started before execution activities. However, especially in the MDD context, it is extremely important to allow switching between execution and preparation. Typical conditions for revisiting the preparation activities are: change of requirements (e.g. a new platform is introduced), detailing of requirements (e.g. because those details were simply omitted), feedback from execution (e.g. something is missing, or the modelling language is too poor to express real situations or it is too complex for expressing something that is simple).

The MODA-TEL methodology identifies the following phases:

1. Project management
2. Preliminary preparation
3. Detailed preparation
4. Infrastructure setup
5. Project execution

Figure 2 shows the five phases of the MODA-TEL methodology. The phases correspond to the available and required expertise and the activities identified in the MDD approach, being directly associated with the partitioning of the MDA users expertise as defined in section 2.1.

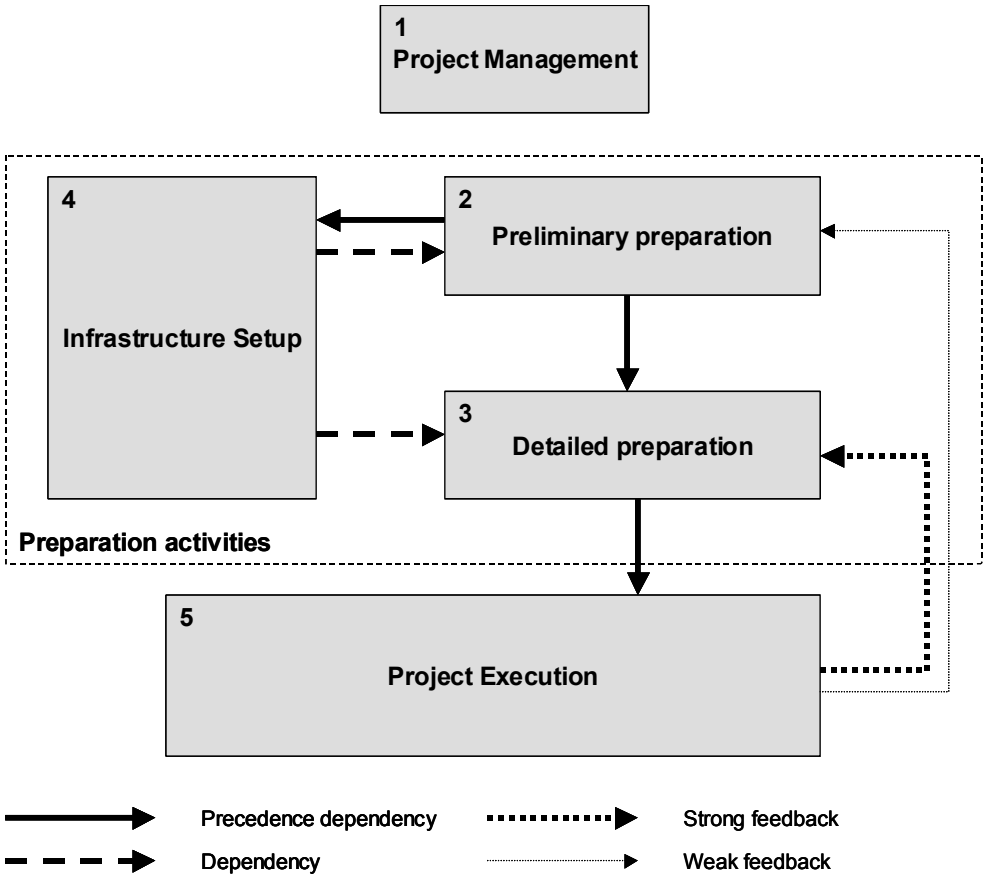

Figure 2 Basic phases of the MODA-TEL methodology

 © MODA-TEL Consortium 2003

Phase 1 corresponds to the second group of expertise (assemble, combine, customise and deploy know how). Phases 2, 3 and 4 correspond to the first group of expertise (build know how repositories). Finally phase 5 corresponds to the third group of expertise (apply the know how).

Figure 2 also shows how the preparation activities have been structured in different phases. These phases are useful to understand and to describe the dependencies between the activities. Project management activities have a direct impact on all the other activities; in particular, the activity that defines the whole software development process prescribes the list of the execution activities to be run, such as, e.g., the sequence of transformations to be implemented. Preliminary or detailed specifications activities, such as selecting a platform or deciding on the usage of a modelling language, are the key elements to enable reuse of know-how in the execution activities. Finally, the infrastructure set-up activities, such as tool selection, may influence the preliminary or detailed specification activities, even if ideally managers may desire to be as much tool-independent as possible.

As stated before, the development process should be iterative and incremental. In order to take efficiently the feedback from the execution activities into account, it is important to have a good level of automation, through model-to-model and code generation techniques, as well as a well-defined traceability strategy.

Each one of the phases identified above is discussed in the sequel.

3 Project management

In this document we distinguish between pure "process management" activities, such as the ones typically handled using MS-Project to manage milestones and resource consumption, and activities that are directly related to management decisions absolutely necessary to setup the project, such as an engineering process selection. Additional activities known and applied from "best practises" in project management can still be added to this phase, but are not covered by this methodology.

It is important to note that the management activities in the context of this document can be strongly influenced by preparation activities (such as SPEM process definition) and by execution activities (such as the requirements analysis). Figure 3 depicts the activities and their relationships in this phase.

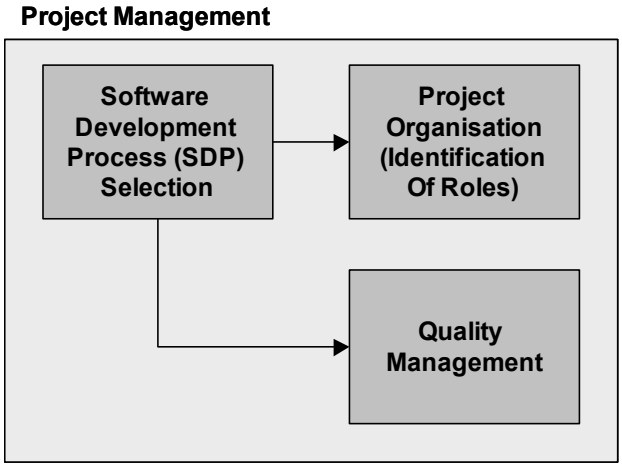

Figure 3 Project management phase and its activities

In the project management phase we identify three activities:

- Software Development Process (SDP) Selection, which results in the description of a software development process to be followed at the execution phase. A discussion on established software development processes can be found in [7].
- Project Organisation (Identification of Roles), which results in the allocation of activities to process roles.
- Quality Management, which is dependent on the SDP Selection. Specific sub-activities and the resulting work products are defined by the selected SDP. Some aspects of quality management can be orthogonal to the SDP for example the Capability Maturity Model (CMM) [8] as discussed in [7].

© MODA-TEL Consortium 2003

Because MDA is based on proven principles of object orientation and component based development, it fits well into established software development processes. If such processes exist in a corporation or are to be followed in a certain project, then the activities, as described in these processes, should be followed.

While the MDA standards focus mainly on the aspects of modelling and model transformations, it is important to base the methodology of developing MDA applications on a defined software development process. It is beyond the scope of the MODA-TEL project to introduce a complete, consistent software development process based on MDA. Instead MODA-TEL has highlighted (see section 2 in [7]) the principles of existing development processes and the specific aspects added or tailored by a strict model-driven approach.

The software development process related to MDA could also be seen as Model Driven Engineering (MDE). The engineering aspect – the process of designing, building and maintaining pieces of software – is more important than the static nature of a set of models. Although there will probably be no single way of engineering software, it is important to refer to established software development processes to give an orientation in the model driven development process.

Figure 4 shows the relationship between the SDP selection activity of the process management phase and the project execution phase.

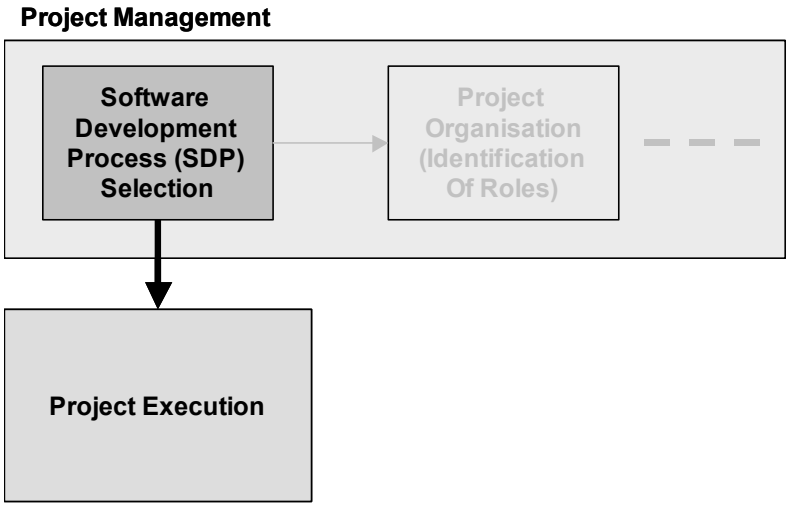

Figure 4 Dependency of the execution phase on the SDP selection

4 Preparation activities

The preparation activities have been grouped in three phases, namely preliminary preparation, detailed preparation and infrastructure setup. Each of the phases and their relationships with other phases are discussed below.

4.1 Preliminary preparation phase

Figure 5 shows the activities of the preliminary preparation phase.

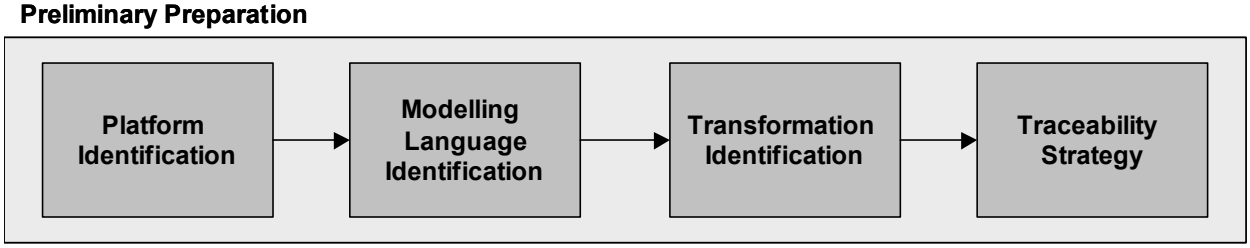

Figure 5 Preliminary preparation phase and its activities

In the preliminary preparation phase we identify four activities:

 © MODA-TEL Consortium 2003

- Platform Identification: A platform refers to technological and engineering details that are *irrelevant* to the fundamental functionality of a system (or system part). What is irrelevant and what is fundamental with respect to a design depends on particular design goals in different stages of a design trajectory. Therefore, in order to refer to platform-independent or platform-specific models, one must define what a platform is, i.e., which technological and engineering details are irrelevant, *in a particular context* with respect to particular design goals. These platforms include legacy platforms. This activity includes abstract platform identification. For a discussion on *abstract platform* refer to [7], section 4.1.
- Modelling Language Identification: Models must be specified in a suitable modelling language capable of expressing relevant domain knowledge. This activity identifies the specific needs for modelling languages. Since models are used for various different purposes, such as data representation, business process specification, user requirements capturing, etc. there exist different types of modelling languages. Process roles for performing this activity include the domain experts.
- Transformation Identification: Once concrete and abstract platforms have been identified, as well as suitable modelling languages, this activity identifies the possible and/or necessary transformation trajectories. For a discussion on model transformations see [6], section 6, and [5], section 2.2.
- Traceability Strategy Definition: Traceability in model transformation refers to the ability to establish a relationship between model elements or sets of model elements that represent the same concept in different models. Traces are mainly used for tracking requirements and changes across models. Traceability is discussed in [6] (section 8.2) and [7].

The preliminary preparation activities often follow the informal requirement analysis activity as depicted in Figure 6. The requirement analysis activity itself is part of the execution phase and the work products and sub-activities are defined by the selected SDP.

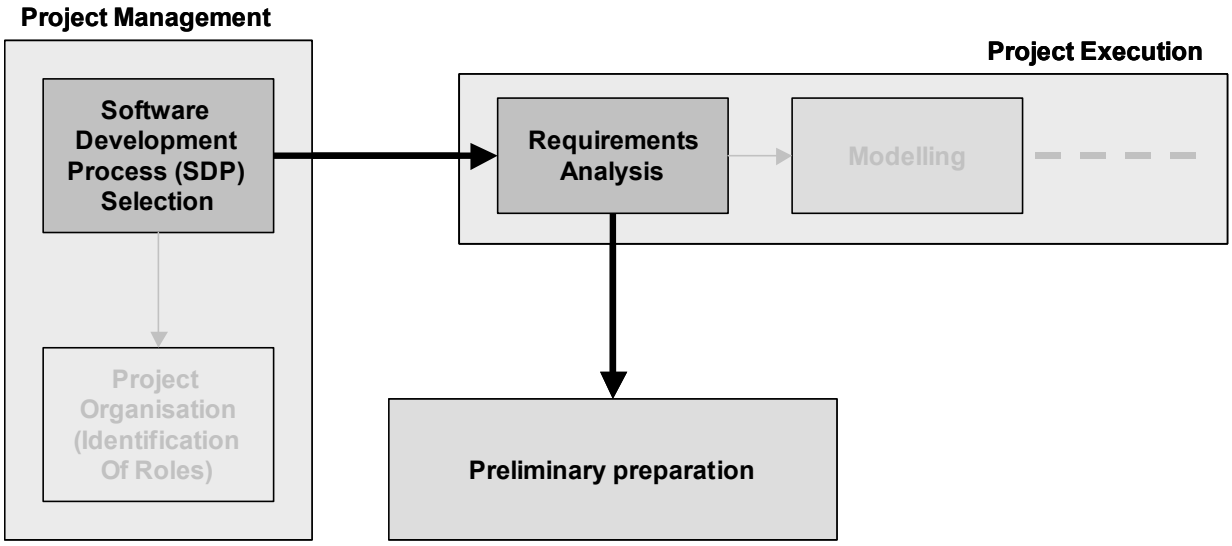

Figure 6 Dependency of the preliminary preparation on the requirements analysis

When model driven techniques are used for requirement analysis, certain preliminary preparation activities precede the requirement analysis. For example, this can be the case if a UML profile or a meta-model is available for the User Requirement Notation (URN) [9]. Identifying such a profile or meta-model is a preliminary preparation activity.

4.2 Detailed preparation phase

Figure 7 shows the activities of the detailed preparation phase.

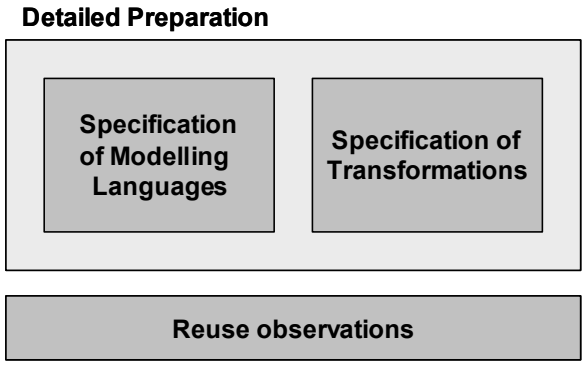

Figure 7 Detailed preparation phase and its activities

In the detailed preparation phase we identify two activities:

- Specification of Modelling Languages: Following the identification of specific needs for modelling languages, this activity identifies the concrete general purpose and/or domain specific modelling languages that shall be used in the execution phase. Also source and target meta-models used by the transformations are created during this activity. Process roles for performing this activity include the domain experts.
- Specification of Transformations: Model transformations need *rules* and *annotations* to control the transformation process. The rules, defined on the meta-model level, can control the transformation of any annotated source model to a target model. Note that the meta-models of the source and target models are not necessarily identical. Rules can be formalized in a certain language or meta-model. The rules may also simply be code in a programming language. Annotations are information related to a model and optionally based on their own meta-model. This activity is concerned with the specification of the needed transformation rules and annotations. For a discussion on using transformations and annotations see [6], section 6.

During this phase, the notion of an MDA component may be relevant. In the project MODA-TEL, this term has been defined to denote *an exchangeable and deployable package of meta-models or UML profiles, consisting of source meta-models/profiles, target meta-models/profiles and annotation meta-models/profiles plus an optional transformation model based on some transformation meta-model/profile, needed to describe a transformation process from source models to target models and optionally to target code.* See [6], section 5.3 for more information on MDA components.

As of today there is no standard or commonly accepted definition for an MDA-Component, but assuming the definition above, the use of existing available MDA components implies the use of a certain modelling language and certain transformations embedded in the MDA component.

In this phase, the methodology foresees the capturing of specifications for reuse. Namely transformation specifications as well as specification of modelling languages are subject for re-use in future projects.

 © MODA-TEL Consortium 2003

4.3 Infrastructure setup phase

Figure 8 shows the activities of the infrastructure setup phase.

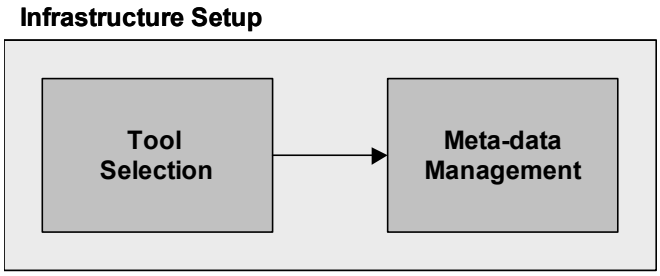

Figure 8. Infrastructure setup phase and its activities.

In the infrastructure setup phase we identify two activities:

- Tool selection: A number of activities have to be handled by tools, such as (i) the definition of models and meta-models, (ii) the transformation and code generation based on the model information, (iii) the definition of constraints and rules to verify the compliance of the models. Additionally, the MDA tools must enable the definition of behaviour and support its simulation, as well as provide traceability capabilities to associate code fragments to original model elements. Which MDA-tool is right for a certain undertaking depends mainly on the level of Model Driven Engineering capabilities it must provide. For the selection of appropriate tools, all requirements from the software engineering perspective are identified and mapped to capabilities of existing tools available on the market. Important but not necessarily functional capabilities with respect to tools are extensibility, integration with XML-based techniques and integration with specialised tools that support modelling behaviour of specific domains as well as interoperation with other tools on the market.

  The objective of this activity is the selection of one or more tools that will support the development process and should be based on an evaluation of the aspects described above. However, business decisions may influence the selection of tools (e.g. tools already available in a corporation or engineers already familiar with a certain tool). The tool selection may have an impact on each of the preparation activities, as well as on the meta-data management. For a discussion on identifying and applying MDA tools, see [7] section 5.

- Meta-data management: Meta-data provides in most cases information about the structure of data, e.g. what are the data types that are available, how many fields are defined in each data type, what data aggregations are valid, etc. Different technology families usually define their own way of managing meta-data as well as how meta-data repositories are generated and manipulated. Meta-data support different activities e.g. can be used during transformations, to store information about available resources, to support migration, or to support applications during runtime. For a specific project, the precise need for meta-data as well as the way to manage meta-data is defined during this activity. For a discussion on meta-data management, see [7], section 8.

© MODA-TEL Consortium 2003

5 Execution phase

Figure 9 shows the activities of the infrastructure setup phase.

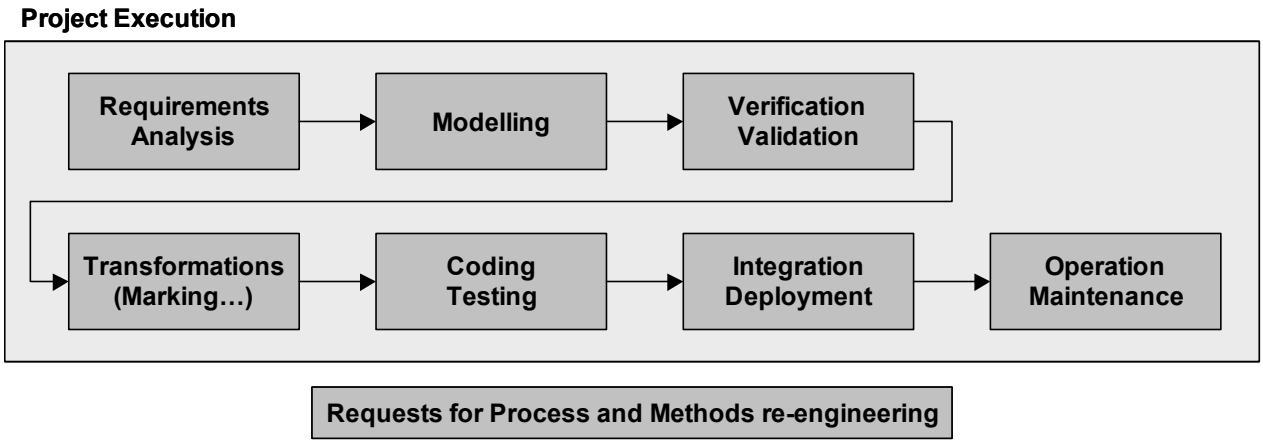

Figure 9 Execution phase and its activities

The MODA-TEL methodology is partitioning the project execution phase into seven activities. The project execution phase is the main implementation phase of the project in which the developers are engaged to apply the know-how. The activities in this phase depend on the selected SDP, but nevertheless, some activities can be identified in every SDP. The activities identified here serve showing which activities are relevant to the MDA, and how they relate to "standard" SDP-activities)

- Requirements analysis: As stated earlier, the requirements analysis activity itself is part of the execution phase and the work products and sub-activities are defined by the selected SDP. Generally the requirements specification aims at the establishment of a well-defined terminology and structuring captured information. Such structured information is used later to refine the requirements into design-oriented models. Optimally resulting models should point back into the requirements for traceability and test specification formalisms. Model driven techniques can be used for requirement analysis, e.g. if a UML profile or a meta-model is available for the User Requirement Notation (URN) [9]. It could be also possible to have some model-to-model transformation that creates a starting PSM from the requirements model.
- Modelling: This is the activity, which by means of one or more appropriate modelling languages, formally specifies, constructs, documents and possibly visualizes the artefacts of distributed systems. This activity is concerned with developing software engineering specifications that are expressed as a system's object/component model. The products of this activity are specifications of the structure of these artefacts, such as names, attributes and relationships with other artefacts. The behaviour model describes the behaviour of the artefacts in terms of states, allowed transitions and the events that can cause state changes. These models are created inside tools that support the representation of the artefacts and their behaviour.
- Verification/Validation: This activity is concerned (i) with the process of determining whether or not the products of the modelling activity fulfil the requirements established by the requirements analysis activity, and (ii) with the process of evaluating whether the products of the modelling activity are free from failures and complies with the requirements established by the requirements analysis activity. To some extend, existing technologies allows these activities to be performed (semi-) automatically by tool support. However it is important to note that establishing a verification/validation strategy for the produced models is an important sub-activity.
- Transformations: This activity is concerned with the refinement of the models produced by the modelling activity by means of rules and annotations that control the transformation process. The artefacts defined by the modelling activity are refined by defining data structures and procedures, defining message protocols for the interactions, mapping the artefacts into classes and mapping these into constructs of a programming language. For a discussion on model transformation see [6], section 6.

 © MODA-TEL Consortium 2003

- Coding/Testing: This activity is concerned with the development of code that complements any automated generation of code or code fragments. Usually coding is still required by developers. The same applies also for the execution of test cases. To some extend automatic testing is possible, but usually manual testing according to various levels of testing activities applies.
- Integration/Deployment: As is common in large organisations, new services and applications have to co-exist with established systems and work in already existing infrastructures. This activity is concerned with the integration of the newly developed parts into the existing systems landscape. The MDA approach allows for modelling of the new over-all functionality at the platform independent level, this being the level of integration. The model representation of the existing systems can be developed by reverse engineering, but usually it is sufficient to model the interface functionality and the required common meta-data. For a discussion on application integration see [7], section 4.3.

  The deployment sub-activity is concerned with parts of the overall management of a distributed application's lifecycle being materialised by a set of interconnected component instances running on a given node or collection of nodes (the platform). The deployment sub-activity corresponds to the transfer of implementations to appropriate nodes, so that corresponding parts of the application can be instantiated and started on these nodes. Associated with deployment is the initial configuration being the first observable state of a distributed application at run time.
- Operation/Maintenance: This activity is concerned with parts of the overall management of a distributed application's lifecycle including dynamic configuration, dynamic service upgrade, service migration to different nodes etc. The precise nature of the activity is prescribed by business requirements.

In general, the activities in the execution phase can be repeated more than once, i.e. if failures, defects or other problems are discovered in one of the activities the process should resolve the issue at the modelling activity.

All activities in the execution phase are encouraged to propose refinements and improvement of the applied processes and methods. The propositions should then influence the preparation phases; either the preliminary preparation, the detailed preparation, or both, depending on the intricacy of the proposition.

© MODA-TEL Consortium 2003

References

[1] http://www.modatel.org

[2] Object Management Group. *Software Process Engineering Meta-model V1.0 (SPEM)*, (formal/02-11-14) November 2002

[3] Object Management Group. *MDA-Guide, V1.0.1*, (omg/03-06-01) June 2003

[4] Object Management Group. UML 2.0 Superstructure, (ptc/03-08-02) August 2003

[5] MODA-TEL project deliverable, *Assessment of the Model Driven Technologies – Foundations and Key Technologies*, http://www.modatel.org/public/deliverables/D2.1.htm

[6] MODA-TEL project deliverable D3.1, *Model Driven Architecture Definition and Methodology*, available via http://www.modatel.org/public/deliverables/D3.1.htm

[7] MODA-TEL project deliverable D3.2, *Guidelines for the Application of MDA and the Technologies covered by it*, available via http://www.modatel.org/public/deliverables/D3.2.htm

[8] *The Capability Maturity Model*, Carnegie Mellon University, Software Engineering Institute, Addison Wesley Publishing Company, 1995

[9] ITU-T Recommendation Z.150: *User Requirements Notation (URN) – Language requirements and framework*, Geneva, February 2003

 © MODA-TEL Consortium 2003