INFORMATIK 2011 - Informatik schafft Communities
41. Jahrestagung der Gesellschaft für Informatik , 4.-7.10.2011, Berlin
www.informatik2011.de

Representation of Process Models

Dr. Reinhold Thurner, Dr. Georg Metz
Metasafe GmbH
CH8700 Küsnacht, Rebweg 21Switzerland
info/at/metasafe.eu

Abstract: Process models for IT, IT-Operations (ITIL), IT-Development (CMMI), IT-Governance (COBIT) are important means to establish assessable quality measurement and control for the IT-industry. The systems are evolving to a common model and become increasingly complex. A flexible architecture and a powerful infrastructure are required to support the development, publishing, training and practical application of these models. The first part of this paper takes a broad view on existing process models and their development. The second part explains a generalized architecture for the representation of process models. The third part describes the Metasafe Repository as a data platform (DBMS) to manage the models and the instance data of IT-processes.

1 IT-Process-Models

Professional engineering and the mastering of planning and manufacturing processes is the basis of the modern manufacturing industry. The IT industry had to learn the same lessons.

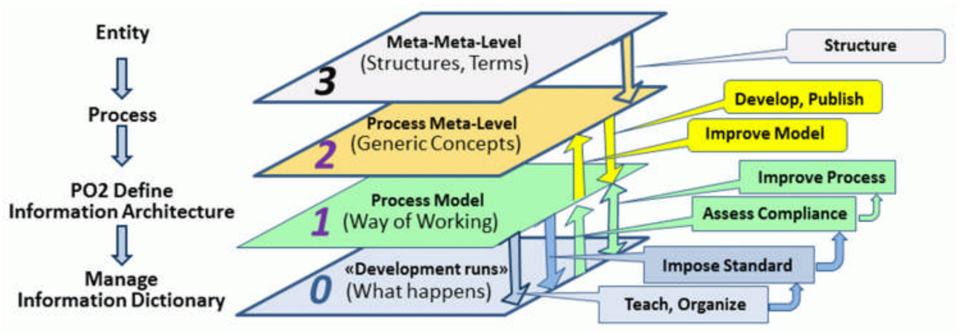

Entity
Process
PO2 Define Information Architecture
Manage Information Dictionary
3 Meta-Meta-Level (Structures, Terms)
2 Process Meta-Level (Generic Concepts)
1 Process Model (Way of Working)
0 «Development runs» (What happens)
Structure
Develop, Publish
Improve Model
Improve Process
Assess Compliance
Impose Standard
Teach, Organize

Fig. 1 – the Layers of IT-Processes, Adapted from [ROL1]

When IT-systems process complexity increased (level 0 – "what happens") and the control of those processes began to decrease, system engineers started to think *about* the processes (level 1 – way of working) in a more generalized way (level 2 - the concepts).

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

In 1968 the famous NATO meeting of Garmisch-Partenkirchen in Germany created the term “software engineering”. It took quite some time until the process-engineering aspect was formally introduced.

The milestones of the development of process models:

- In the early 1980’s the CCTA published in UK the “IT Infrastructure Library” ITIL to establish (and enforce) standard practices in *IT-Operations.*
- 1989 Watts Humphrey wrote his book about the “Capability Maturity Model” CMM and introduced the idea of a “verifiable capability maturity assessment”. CMM became the de facto standard model for *Software Development*.
- 1996 ISACA – the Information Audit and Control Association – published the first Version of “COBIT” for *IT-Governance*. The mission is “to research, develop, publicize and promote an authoritative, up-to-date, international set of generally accepted information technology control objectives for day-to-day use by business managers, IT professionals and assurance professionals.” [ISA00]. COBIT is concerned with the questions:
    - Are we doing the things the right way?
    - Are we getting them done well?
- 2006 ISACA published the first version of the “ValIT-Framework to extend the COBIT-Framework by the question:
    - Are we doing the right things?
    - Are we getting the benefits?

Dedicated models have been developed for special application areas. Some are even established as a prerequisite to be selected as a supplier: The automotive Spice [ASP01] model e.g. is used to assess suppliers of the auto-industry and to check if they reach the SPICE-capability level three which is a prerequisite to be listed as an accepted supplier.

Fragmentation of models: The initial models for IT-operation, IT-development and IT-governance used different terminologies and structures; different models for IT-development were introduced in the US (CMM), Germany (V-Modell), France (Meurise) and Switzerland (Hermes). This lack of standardization was a problem for tool-developers and even more so for international companies.

Model integration: The process models are evolving to higher levels of integration and standardization. COBIT integrates ValIT and RiskIT into COBIT5. ITIL/V2 changed to a service-oriented-view with ITIL/V3. The Software Engineering Institute has moved from the early version of CMM to a collection of models under the label of CMMI. “Enterprise Spice” published “An Integrated Model for Enterprise-wide Assessment and Improvement” [ISO02] in September 2010. The Zachmann Model [Zach] is a blueprint for a global integrated model.

Tools: Models are complex artifacts. Tools are required to store and manage the data, to create appropriate representations (graphics, text), reports and to provide access for the users of the model.

Data infrastructure: The information contained in the models is stored in structured text files (PDF, Word), Databases / Repositories, XML/XMI-Files, graphical representations. One form of representation can be transformed into another one.

Model economics: The management, training, certification and assessment of these models, has created a multi-million dollar business for the assessment "industry" with considerable costs and – if the models are properly applied - benefits for the users. Despite the substantial functional overlap between the models for IT-Governance, IT-Development and IT-Operations each of these models has its own structure, its own terminology and its own certification and assessment rules. Users would prefer a common terminology and an integrated model to *get the benefits* from the use of the models in a less expensive and easier manner.

2004 The international standards organization ISO published the norm ISO/IEC-15504-1 to create a standard vocabulary for process models and types of models [ISO01] – which belongs to the level 2 in the layer model.

The key definitions of ISO 15504-1:

- *"Process Reference Model:* a model comprising definitions of processes in a life cycle described in terms of process purpose and outcomes, together with an architecture describing the relationships between the processes.
- *Process Assessment Model:* a model suitable for the purpose of assessing process capability, based on one or more Process Reference Models.
- *Process Capability Determination:* a systematic assessment and analysis of selected processes within an organization against a target capability, carried out with the aim of identifying the strengths, weaknesses and risks associated with deploying the processes to meet a particular specified requirement.
- *Process Improvement:* actions taken to change an organization's processes so that they more effectively and/or efficiently meet the organization's business goals."

Conclusion: Process Models are still evolving from "general best practices" to "rules of compliance" and further to "models for continuous improvement". The scope is extended to cover the entire IT-lifecycle and includes portfolio and investment-management (ValIT).

2. Layer Architecture of Process Models

In general the term "Process Model" or "Process Framework" refers to a "process reference model" or a "process assessment model". These models are abstracted descriptions *about* "what happens in reality", i.e. *about* the "instances of processes". The descriptions tend to be large and complex. To develop, explain or check such a model one needs a further abstraction - a "metamodel" – i.e. a description of a description. A layer architecture describes the "meta-layers" of models and their purpose.

2.1 The Metamodel Architecture of OMG

The question how to describe a description is not new: "OMG was in need of a metamodeling architecture to define (the modeling language) UML. The MOF (the Meta Object Facility) is designed as a four-layered architecture. MOF metamodels are usually modeled as UML class diagrams. A supporting standard of MOF is XMI, which defines an XML-based exchange format for models on the M3-, M2-, or M1-Layer."[OM01][WP01] This model is also the basis for EMF – the Eclipse modeling framework. EMF-models belong to the layer M2 and are used to document and generate the java-code which belongs to the M1-layer. It is up to these java-programs to deal with the information of the reality – i.e. the M0-layer.

This four-layer architecture can also be applied to the representation of process models. It extends the (initial) three layer model of Fig.1 [ROL1] by an additional layer on top to provide a description for the metamodel.

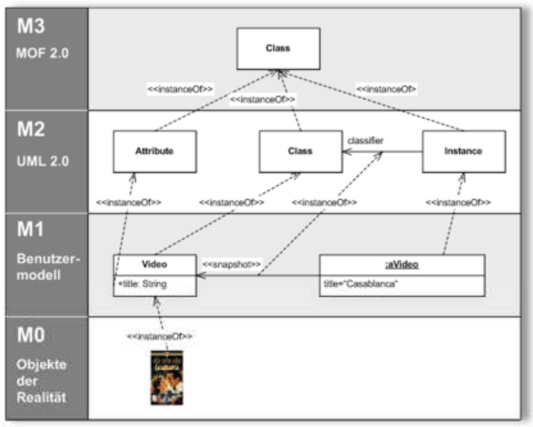


Fig 1: The 4 Layer Metamodeling Facility [WP02]

- The M0-Layer corresponds to the application of the model in a subject area.
- The M1-Layer corresponds to the detailed description of the process model e.g. in COBIT the 34 processes with their attributes and relationships.
- The M2-Layer defines the metamodel of the M1-Layer: It describes terms used in the M1-Model (Entities: process, resource, outcome; Attributes: ID, name, description, date; Relationships: hasOutput, isInput, contains, creates, isResponsible). These terms are described by metaattributes (ID, reference, description, representation etc).
- The M3-Layer – the metameta-layer - defines the vocabulary and grammar for the description of metamodels.

2.2 The M3-Model – based on the Entity-Relationship-Model

Purpose:

- Provide a uniform (data) structure which is able to reflect all the elements of the lower levels (M0 to M2).
- Control tools to manage M2-models

Process models (M1 to M2) and process instances (M0) are structures of highly connected fine grained data. The ERA-model reflects this structure in a generalized way. [Chen]. The elements of a process model (e.g. process, outcome, category, basepractice) are modeled as *Entities* which are connected by *Relationships* (e.g. relationship notes, cardinality, data flow, process flow, responsibility). The properties of entities and relationships are described by *Attributes* (e.g. name, purpose, ID).

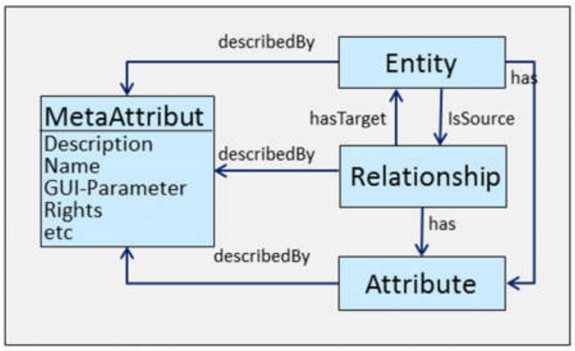


Fig 2: The Entity-Relationship-Model (simplified)

The elements of the M3-level (Entities, Relationships and Attributes) are described by MetaAttributes (e.g. Description, Syntax-Rules, Format, Representation, Widget etc.) to provide a self-description of the M3-Model.

2.3 The M2-Layer – the MetaModel

The metamodel defines the structure of a concrete process model (based on the rules defined by the M3-layer). The metamodel describes all the relevant terms of the domain, their properties and relationships.

***Purpose*:**

- Complete description of the metamodel (see MetaAttributes)
- Provide a structured description of the content of the (M1) model.
- Support the development of a (M1) model / translation into other spoken languages
- Teach a (M1) model to users / assessors: The audience can understand the structure and purpose without digging into all the details.
- Compare models with other models (e.g. COBIT with ITIL with ISO with CMMI...)
- Clone a (M1) model to adapt it to a specific environment without losing the structure and the reference to the original, create submodels for specific purposes

Tools
A model-editor to develop / document the model and to create individual views of the model in different representations (Database, XML/XMI, structured text, graphics). Import and export-facilities to exchange models. Tools for consolidation of models.

Examples:
- Entities: Process, Outcome, Workproduct, Person, User,
- Relationships: hasOutput, isInput, isResponsible, follows,
- Attributes: ID, Name, Purpose, Description, Cardinality, Relationshiptype etc.
- Structures: "Process hasOutput BasicWorkProduct",
  "Process hasInput BasicWorkProduct", "Process use Resource",
- MetaAttributes for e.g. WorkProduct: Description, Create/Read/Update/Delete Rules, rules for creation of the ID (key) of an instance,

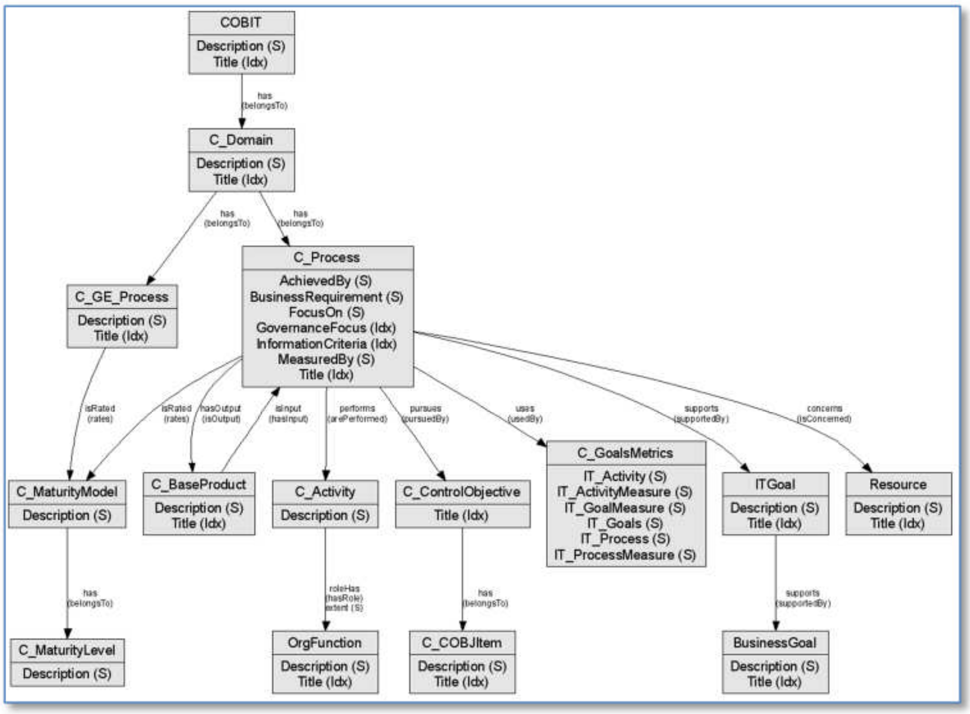


Fig 3: Simplified Metamodel for the COBIT4.1 Reference Model

2.4 The M1-Layer – the model (or Framework)

Purpose:
- Describe how "it should be done", i.e. how individual processes should work
- Teach users / assessors, provide certificates for organizations or individuals
- Describe measurement principles, prescribe nonfunctional requirements

- Support the assessment "Are we doing the things the right way?" "Are we getting them done well?" [ISA02]
- Support the assessment "Are we doing the right things?", "Are we getting the benefits?" [ISA02]

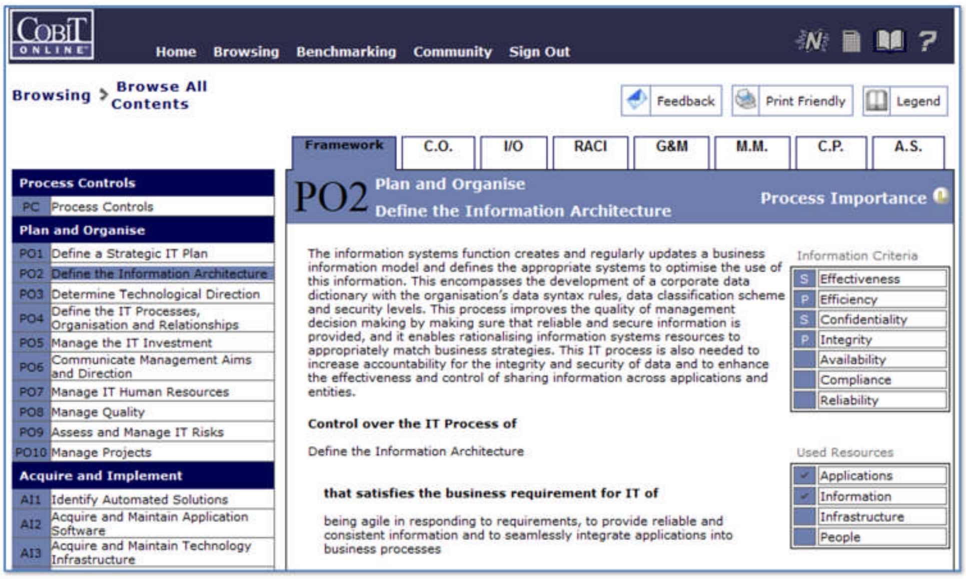
Tools

- Editors to develop / check the model, to create cross references and dependencies.
- Tools to publish the model or extracts of the model in various representations (Database, XML, structured text, graphics).

Example

A model consists of a medium number (few to a few hundred) of instances of each entity-type. The models are normally published in form of structured text, XML or HTML for publication on the web.

COBIT is described in a 200-page PDF-file [ISA03] and in an Access-Database which can be accessed via COBIT-Online [ISA04]. "Frankfort School of Finance" [GO08] has also implemented an access for COBIT.



Process Controls

PC	Process Controls

Plan and Organise

PO1	Define a Strategic IT Plan
PO2	Define the Information Architecture
PO3	Determine Technological Direction
PO4	Define the IT Processes, Organisation and Relationships
PO5	Manage the IT Investment
PO6	Communicate Management Aims and Direction
PO7	Manage IT Human Resources
PO8	Manage Quality
PO9	Assess and Manage IT Risks
PO10	Manage Projects

Acquire and Implement

AI1	Identify Automated Solutions
AI2	Acquire and Maintain Application Software
AI3	Acquire and Maintain Technology Infrastructure

PO2 Plan and Organise
Define the Information Architecture
Process Importance

The information systems function creates and regularly updates a business information model and defines the appropriate systems to optimise the use of this information. This encompasses the development of a corporate data dictionary with the organisation's data syntax rules, data classification scheme and security levels. This process improves the quality of management decision making by making sure that reliable and secure information is provided, and it enables rationalising information systems resources to appropriately match business strategies. This IT process is also needed to increase accountability for the integrity and security of data and to enhance the effectiveness and control of sharing information across applications and entities.

Control over the IT Process of

Define the Information Architecture

that satisfies the business requirement for IT of

being agile in responding to requirements, to provide reliable and consistent information and to seamlessly integrate applications into business processes

Information Criteria

S	Effectiveness
P	Efficiency
S	Confidentiality
P	Integrity
	Availability
	Compliance
	Reliability

Used Resources

✓	Applications
✓	Information
	Infrastructure
	People

Fig 4: COBIT-Online [ISA04]

The COBIT4.1 framework consists of more than 10.000 individual elements (entities, attributes and relationships). ISACA has published in addition the "Assurance Guide", the "Assessment Guide", the "ValIT"-Framework and "RiskIT". ISACA is integrating these models into COBIT5. ISACA has announced that the resulting model will be based on a "Knowledge base" and stored in a repository.

2.5 The M0-Layer – the Real World Instances

The M0-level represents the processes of the real world and deals with the (IT-) data. Systems which support these processes have to cope with large volumes of data, with complexity, with multi-user access and above all with frequent changes. Large systems especially implement several federated repositories instead of one single repository holding all the data. However these repositories should share a common data model and should be based on the same data infrastructure (Repository Manager), which supports the management of complex information in a secure multi-user mode.

***Examples*:**

- The COBIT-Process (M1) "PO2 Define Information Architecture" (see Fig 4) describes as a control objective: "Enterprise Data Dictionary and Data Syntax Rules - Maintain an enterprise data dictionary that incorporates the organisation's data syntax rules. This dictionary should enable the sharing of data elements amongst applications and systems, promote a common understanding of data amongst IT and business users, and prevent incompatible data elements from being created."
- The COBIT-Process (M1) "DS9 Manage the Configuration" Ensuring the integrity of hardware and software configurations requires the establishment and maintenance of an accurate and complete configuration repository. This process includes collecting initial configuration information, establishing baselines, verifying and auditing configuration information, and updating the configuration repository as needed. Effective configuration management facilitates greater system availability, minimises production issues and resolves issues more quickly."
- A CMDB (configuration management data base) is a core element for IT-Operations in ITIL/V2 and even more so of ITIL/V3.
- CMMI defines the role of CM and for IT-Service-Management: "The purpose of Configuration Management (CM) is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits." [CMMI]

3 The Implementation of Models with the Metasafe Repository

The Metasafe© Repository was constructed as a general purpose repository to store fine-grained, highly interconnected entities in a proper Database Management System (DBMS): This includes multi-user-support, transactions, controlled access and the ACID-conformity. The repository is based on a multi-level ERA-Model.

The repository was designed to support very large models with thousands of elements on the model-level (entity-types, relationship-types and attribute-types). The repository can handle millions of instances on the M0 and M1-level and gigabytes of data. The repository uses special caching techniques to provide the necessary performance: Even in a rather modest environment (PC, Windows/7, standard desktop) about 10.000 entities per second can be written and about 40.000 per second can be read. It provides isolation and fine-grained access-rights in a multi-user environment.

The use of the repository to manage process models (M1 to M3) is just one of the many possible applications. Typical other applications are Application and Investment Portfolio Management, Balanced Scorecard Reporting and Analysis, Information Dictionary, Masterdata Management, Data Governance, Risk Management, Enterprise Architectures.

The model of the repository is organized in a 4-layer structure where each structure has distinctive functions. The numbers in Fig 5 indicate the size of each layer in number of elements: The M0 may contain gigabytes of instance data. The M1 layer of a single repository instance is used to manage the models of the M0-layer using dedicated catalogs. A (M2) model which controls thousands of data-elements on the (M1) level model could consist of thousands of elements, which in turn are described by a (M3) metamodel which consists of a small number (hundreds) of elements. The layers provide a means for a gradual reduction of the span of control.

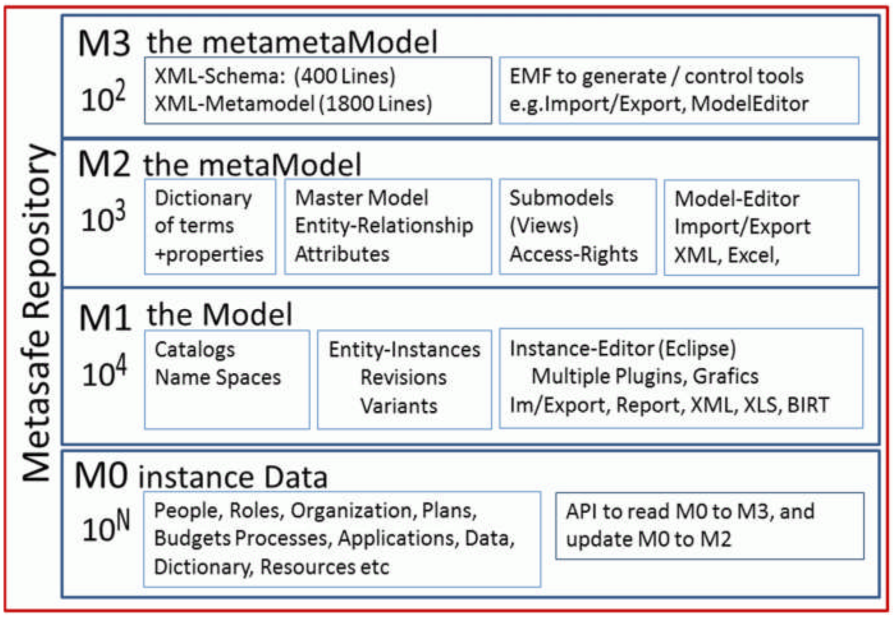

Metasafe Repository
M3 the metametaModel
10²
XML-Schema: (400 Lines)
XML-Metamodel (1800 Lines)
EMF to generate / control tools
e.g.Import/Export, ModelEditor
M2 the metaModel
10³
Dictionary of terms +properties
Master Model Entity-Relationship Attributes
Submodels (Views) Access-Rights
Model-Editor Import/Export XML, Excel,
M1 the Model
10⁴
Catalogs Name Spaces
Entity-Instances Revisions Variants
Instance-Editor (Eclipse) Multiple Plugins, Grafics Im/Export, Report, XML, XLS, BIRT
M0 instance Data
10ᴺ
People, Roles, Organization, Plans, Budgets Processes, Applications, Data, Dictionary, Resources etc
API to read M0 to M3, and update M0 to M2

Fig 5: The logical Architecture of Metasafe

4.1 The MetametaModel M3

The metamodel is used to bootstrap the system core. It describes the basic components of the M2-layer, their attributes and their relationships in a XML-document. The structure of this XML-document is described by a XML-Schema to enforce the consistency when the metamodel is updated. These XML-documents are used for code generation (static) and to control the M2-level tools (dynamic). They represent the innermost core of the system.

4.2 The MetaModel M2

The M2 Layer contains the dictionary, the master model and the submodels (views). The MetaModeller tool (an Eclipse RCP plugin) but also any other program using the Metasafe-API can be used to manage the models. The MetaModeller accesses the repository via the standard API like any other application.

The dictionary describes all the elements (entities, attributes and relationships) with their metaAttributes, as e.g. type, description, format, domain, check-procedure.

The master model describes the entire structure of the M1-level. The master model is created from elements of the dictionary and only from these elements: Some values of the metaAttributes defined in the dictionary can be overridden in the mastermodel depending on the context.

Submodels are constructed as views of the mastermodel and must conform to the mastermodel at all times. Some values of the metaAttributes can be overridden at submodel-level: e.g. the label of an attribute could be changed in the submodel to another language.

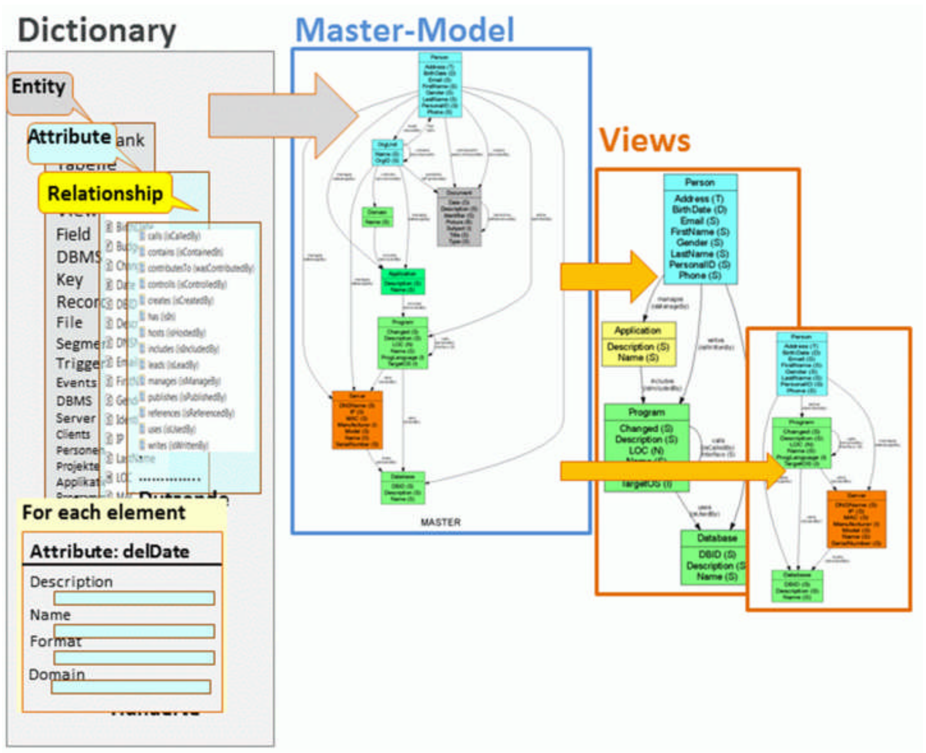


Fig 6: The Structure of the M2-Layer

Strict rules are enforced by the system core to keep the models and the data consistent; these rules are based on the definitions of the metamodel. The core checks the consistency whenever a change request is issued: e.g. the core accepts a request to add an attribute type to an entity in a subschema only if it this type is defined in the dictionary and in the master schema of this entity. This elaborate structure supports the development of very large models and consistentancy for individual purposes.

The subschema-concept and the access-rights tied to the subschemas is an important feature to substructure even very large models into individual and protected areas of interest.

4.3 The Model M1

The M1-Layer is structured into separate storage units (namespaces) called “Catalogs”. Such a catalog can be used to store the (M1) process models to keep them separate from the M0-instance data – and use them to control the instance data. For each entity multiple “variants” (or editions) and “revisions” can be instantiated. Variants are explicitly named, versions of an entity and can be, for example, used to reflect the state in the lifecycle of an entity; revisions are numbered versions of an entity. Revisions and variants can be combined.

Relationships are named bidirectional connections between fully qualified versions of entities. Entities and relationships have attributes as defined in the model.

4.6 The Instance Level M0

The instance data are stored in dedicated catalogs back to back with the models (M1). This level is structured like M1, i.e. in revisions and variants. Various tools are provided to import/export, browse, and display the content of M1 and M0.

5. The Functional Architecture of the Metasafe Repository

The Metasafe repository is designed to store large numbers of entity-types and relationship-types (m:n) where each entity-type (=table) has only a relatively small number (10 to 100.000) of instances and the entities are highly connected by relationships. The repository is comparable to a classical DMBS, however with an Entity-Relationship-Attribute-model. It consists of three layers – the generic tools, the Core (or Repository Manager) and the Persistence layer.

Tools or any other model based application sit on top of this structure. They can be developed by extending the standard Metasafe-tools (Eclipse Plugins) or can be built as standalone applications using the Metasafe Java API.

5.1 The Persistence-Layer

Metasafe stores the physical data in a highly compressed and (optional) encrypted format in a couple of tables of an RDBMS – e.g. Oracle, SQLServer, DB2 or Derby. The structure of the persistence DB is not affected by changes of the models. Changes can be done on the fly and are transaction protected.

The persistence-layer is accessed via a service interface which can be adapted to virtually any RDBMS.

5.2 Metasafe Core:

The metasafe core manages the data structures (Models M0 to M3) and communicates with the persistence layer. The metaLib API for Java is an elaborate framework to access the repository services.

Access is protected by fine-grained access-rights which are granted to user (or applications) of the repository.

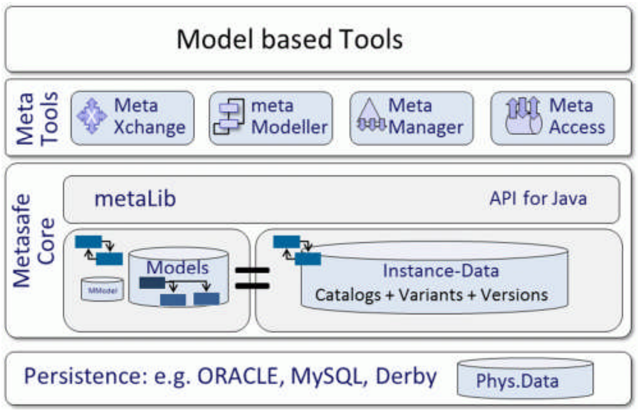


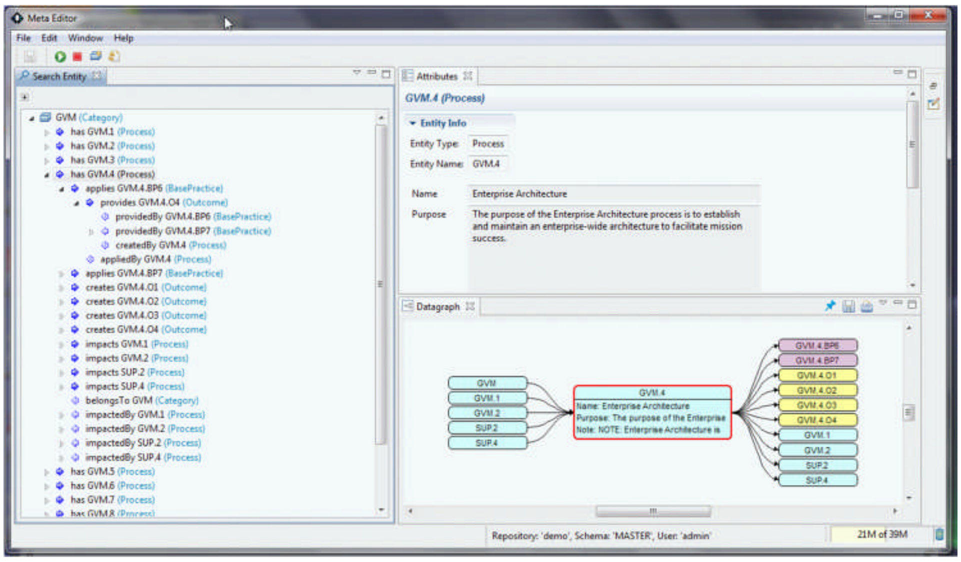
Fig 7: The Functional Architecture of Metasafe

5.3 The Metasafe-Toolbox

A growing collection of generic tools support the usage and management of the repository. The tools access the repository via the standard API. The majority of the tools are plugins for the Eclipse environment.

The MetaModeller is an editor to develop, maintain, document, import / export M2-Models. Views (submodels) can be cloned from the global master model. Models can be changed on the fly without the necessity to unload/reload the content of the repository. Fine grained access rights can be granted to users and user groups.

The MetaEditor manages models on the M1 and M2-layer. It consists of a set of plugins which can be extended to provide special functionalities. The MetaEditor interprets the models and the metaAttributes to generate the user interface on the fly.



```
GVM (Category)
    has GVM.1 (Process)
    has GVM.2 (Process)
    has GVM.3 (Process)
    has GVM.4 (Process)
        applies GVM.4.BP6 (BasePractice)
            provides GVM.4.O4 (Outcome)
                providedBy GVM.4.BP6 (BasePractice)
                providedBy GVM.4.BP7 (BasePractice)
                createdBy GVM.4 (Process)
            appliedBy GVM.4 (Process)
        applies GVM.4.BP7 (BasePractice)
        creates GVM.4.O1 (Outcome)
        creates GVM.4.O2 (Outcome)
        creates GVM.4.O3 (Outcome)
        creates GVM.4.O4 (Outcome)
        impacts GVM.1 (Process)
        impacts GVM.2 (Process)
        impacts SUP.2 (Process)
        impacts SUP.4 (Process)
        belongsTo GVM (Category)
        impactedBy GVM.1 (Process)
        impactedBy GVM.2 (Process)
        impactedBy SUP.2 (Process)
        impactedBy SUP.4 (Process)
    has GVM.5 (Process)
    has GVM.6 (Process)
    has GVM.7 (Process)
    has GVM.8 (Process)
```

GVM.4 (Process)

Entity Info

Entity Type	Process
Entity Name	GVM.4
Name	Enterprise Architecture
Purpose	The purpose of the Enterprise Architecture process is to establish and maintain an enterprise-wide architecture to facilitate mission success.

GVM.4
Name: Enterprise Architecture
Purpose: The purpose of the Enterprise
Note: NOTE: Enterprise Architecture is

Repository: 'demo', Schema: 'MASTER', User: 'admin'

21M of 39M

Fig. 8: Examples of Views of the MetaEditor

The BIRT-Interface connects the repository via an erSQL (entity-relationship Structured Query Language) to the BIRT (Business Intelligence and Reporting Tool) [EC02]. Information from the repository can be formatted, filtered, processed and exported in various formats (doc, pdf, html, xls). Information can also be stored in a hypercube and output from the hypercube can be generated. This facility enables the users to create reports in an easy and user friendly manner.

The Excel-Interface provides as facility to import/export data in normalized form from/to Excel.

MetaExchange: Models and Instance-Data can be imported/exported via XML.

6 Conclusion

A consistent architecture and a corresponding data platform is a powerful basis to construct, maintain, publish and teach model based systems and to develop specific tools. Systems of a medium to large size can especially be developed faster, at lower cost and in a much more flexible way than with a standard RDMBS and the traditional design /generate / code / compile cycle.

References

[ASP01 http://www.automotivespice.com/
[Chen] Peter Chen: "The Entity-Relationship Model: Towards a Unified View of Data." ACM Transactions on Database Systems 1:1. 1976. 9–36
[CHR01] Chrissis, M. B., Konrad, M., Shrum, S.: CMMI: Guidelines for Process Integration and Product Improvement, 2nd Edition, Addison-Wesley (2007)
[CH01] Software Processes, Work Flow and Work Cell Design, Gerhard Chroust, Kepler University Linz, December 1, 2002
[CLE01] Clenio F. Salviano, Adriana M. C. M. Figueiredo, Unified Basic Concepts for Process Capability Models Proceedings of The Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE'08) pp. 173-178, San Francisco, CA, USA, July 1-3, 2008
[CMMI] CMMI® for Services, Version 1.3, November 2010, CMU/SEI-2010-TR-034
[EC01] Eclipse Modeling Framework, at www.eclipse.org,
[EC02] The Eclipse BIRT reporting system www.eclipse.org/birt/
[GO08] Matthias Goeken, IT-Governance-Practice-Network, Management Research Centre Frankfurt School of Finance & Management
[HU01] Humphrey, W.S., Managing the Software Proces, Addison-Wesley Reading Mass. 1989.
[ISA00] http://www.isaca.org/Knowledge-Center/cobit/Documents/COBIT4.pdf, Page 9
[ISA01] http://www.isaca.org/Knowledge-Center/cobit/Pages/Downloads.aspx
[ISA02] http://www.isaca.org/Knowledge-Center/Research/Documents/VALIT-framework.pdf
[ISA03] http://www.isaca.org/Knowledge-Center/cobit/Documents/CobiT_4.1.pdf
[ISA04] COBIT-Online - http://www.cobitonline4.info/Pages/Public/Home.aspx
[ISO01] ISO/IEC, ISO/IEC 15504 -Information Technology Process Assessment
Part 1: Concepts and vocabulary
Part 2: Performing an Assessment
Part 5: An exemplar Process Assessment Model (2006)
[ISO02] Enterprise SPICE® - An Integrated Model for Enterprise-wide Assessment and Improvement Technical Report – Issue 1, Sept 2010
[OM01] www.omg.org
[ROL1] Claudine Rolland. Modeling the Requirements Engineering Process, 3rd European-Japanese Seminar on Information Modelling and Knowledge Bases, Budapest, June 1993
[WP01] http://en.wikipedia.org/wiki/Eclipse_Modeling_Framework
[WP02] http://en.wikipedia.org/wiki/Metamodeling
[WP03] http://en.wikipedia.org/wiki/Meta-Object_Facility
[Zach] The Zachmann Framework, Picture from the Website www.ZachmannInternational.com