Dr. Reinhold Thurner, Metasafe GmbH (Schweiz) www.metasafe-repository.com
Dr. Wolfgang Martin, Wolfgang Martin Team, www.wolfgang-martin-team.net

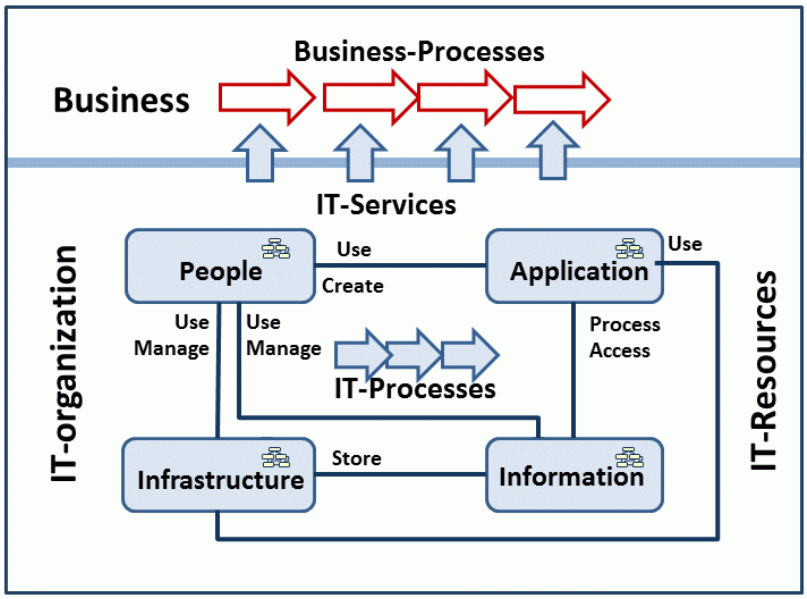
Agile and Industrialized IT Management

Business agility and industrialization

IT has made and makes still an important contribution to the industrialization of the businesses. Business organizations face new challenges in the global, highly competitive and hyper dynamic markets of today. They are under constant pressure to reduce cost *and* to adapt to change or to drive change themselves. Nowadays "Change the business" is as important as "Run the business".

Impact for IT

This has a profound impact also on the provision of IT-services: The IT organization must reduce cost and must follow all changes of business strategy and tactics immediately. IT-organizations which cannot cope with this requirement do not provide value to the business. Sure – one can outsource the provision of IT-services – but the responsibility for IT-service-management cannot be outsourced.



Challenge for the CIO

CIOs striving for sustainable success of an IT department and for optimizing the value contribution of IT to the business have to transform a traditional IT department into an agile and industrialized organization. Industrialization means to achieve operational excellence in all IT services, processes, and products. Agility means the capability for flexibility so that all IT services, processes, and products can be changed on the fly and with the speed that business requires and needs.

Business and IT-Systems

Modern organizations use thousands of individual manual processes or machine processes which are completely intertwined with by IT-services to manage their business and their data. The IT-organization provides these IT-services and manages the necessary IT-resources – the people, the applications, the information and the IT-infrastructure (according to COBIT).

IT-resource Management

These resources depend on each other and undergo constant changes. Data about the resources and their dependencies are necessary not only to control the day-to-day operations but at least important to direct the future.

It depends on the effectiveness of IT- management how flexible, reliable and cost effective the IT-services can be delivered to the business. Therefore, the challenge is to implement an industrialized and agile IT management for constantly reducing cost and improving the agility of IT services.

Challenges for IT Management

IT4IT

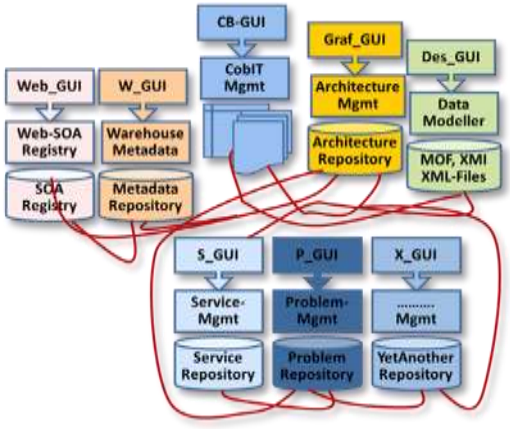
IT-Systems are complex (and expensive) artifacts themselves: Like any modern business organization also an IT-organization needs IT-services –an IT4IT to manage the IT-landscape. Many such "IT-applications" have been developed to support the management of the IT-resources, the acquisition, development and maintenance of business-applications and IT-infrastructures and to run the day-to-day operations.

These IT-applications deal with data *about* the IT-resources (often called Metadata) in contrast to Business Applications which deal with business data. E.g. the “Sales Data Base” holds data about the sales of the business domain – a data dictionary contains data *about* the “Sales Data Base”

First generation of IT-applications

The first generation of these IT-applications is structured in "classical silo-type" manner: One IT-application (tool) for each problem and each application manages its own data. In other words: IT uses a fragmented environment for IT Management.

A fragmented architecture has some severe drawbacks:



Redundant Data The same data are used by several IT-applications, data are scattered across many data stores. To avoid duplicate data entry and redundancy some applications exchange data - in most cases in a point-2-point manner – euphemistically called “copy management”. The same “information” with a sometimes a different meaning is stored in many places – in guidelines, compliance standards, glossaries, development documentation, emails, excel-sheets etc. Consequently, fragmentation costs more money and needs more resources due to broken, insufficiently automated and standardized IT processes. Fragmentation reduces quality, because fragmented IT processes are rather error prone. Finally, fragmentation endangers speed of change due to the complexity to foresee the impact and risk of changes in one system to other systems. Finally, for the same reasons, the amount of testing is dramatically increased.

Inconsistent Data Even worse, every IT system in use has its own vocabulary that depends on the specific scope of each IT system. Every time you add, delete or update a new item, you must not only make sure that the changes will occur in all data bases, but additionally, you must create new translation tables and/or change the new item to all existing translation tables. This makes changes slow and costly. IT becomes a road block for the business, the business does not understand the true reasons, but gets the impression of an IT function that is not business oriented.

Inaccurate Data Redundancy and inconsistency of data drives inaccuracy of data, because different IT services and processes or different IT systems define and view data differently. This impacts data quality. For some applications, quality of certain data has the highest priority, while for other application this piece of data may be a nice to have. So it is difficult and costly to ensure that data has the same quality and is complete and current everywhere in the enterprise. Sometimes, data is even manually re-entered from IT system to IT system. This makes data quality doubtful resulting in mistakes, failures and additional cost.

Narrow View These IT-applications cover a small area but are still very complex and difficult to understand. They provide an isolated view of one single subject, but lack an integrated view. Consequently data are pulled together in Excel-sheets for global management and reporting. Crucial management information is drawn from hundreds of Excel-sheets which are scattered throughout the organization. Information about development artifacts is kept in file-systems (e.g. CVS) or drawings.

Standards There is no lack of standards and models which describe some key processes of an IT-landscape: e.g. COBIT and ValIT define the core IT management processes, RiskIT the rules for risk management, CMMI structures the development process, OMG proposes UML - a language for design and implementation of software, ITIL/V3 defines the processes of IT-operations

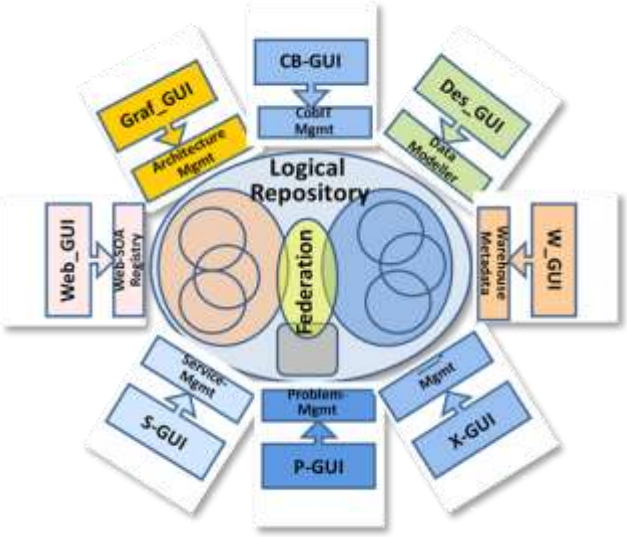
and service delivery. The focus of these standards is predominantly on processes and much less on the data associated with these processes.

Take Away IT applications structured in silo-type manner prevent industrialized and agile IT services, processes, and products. This fragmentation is the main blocker for CIOs when striving for sustainable success of an IT department and for optimizing the value contribution of IT to the business.

Towards a Solution – IT-Consolidation

Consolidation of resources and reduction of complexity is a key ingredient to cost reduction and agility. This recipe concerns not only the IT-Infrastructure (as e.g. Servers, System Software, Networks) the business-applications and the business data but also the IT-data, the metadata.

Consolidated metadata remove the problems of complexity, data fragmentation and data redundancy. Consolidated metadata stored in a central repository are based on a common data model. Such a repository provides individual views for each "IT-application" on the same consolidated data and most importantly ensures the consistency of the "meaning" of the data with a well-documented and enforced data model.



What are the key elements of such a "logical repository"?

- Consolidation of the individual data models with an integration model which is the master of all submodels (views) to keep the system consistent
- Support for submodels to provide a well-controlled individual view for each application.
- Integrated storage of the data-models *and* the corresponding metadata.
- Powerful access mechanism (API) for IT-applications to access the models and the metadata.
- Tool set to support the modeling, browse, query, edit, report, import and export of models and metadata.

Model Consolidation - A Model of Models ("Meta Model")

The first prerequisite towards an industrialized and agile IT Management are reliable information about all IT-resources – metadata based on a common data model. This is the basis to control day to day operation and to decide on future developments.

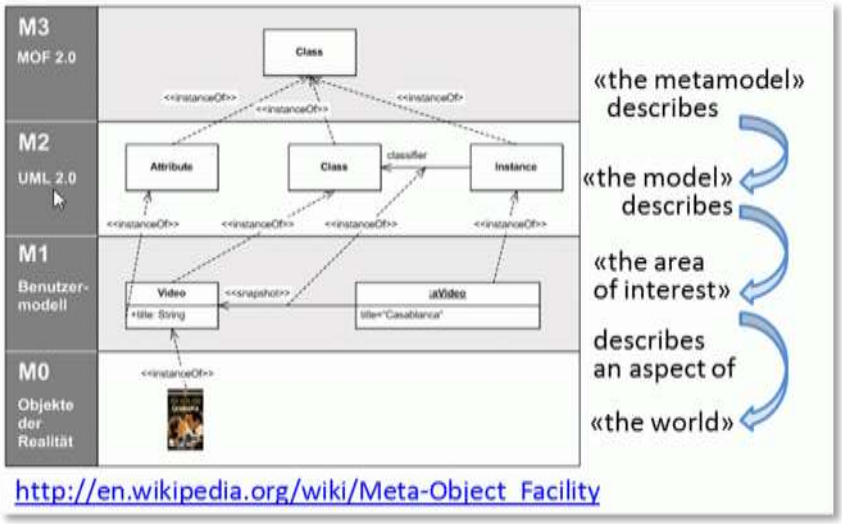
This information comprises data about all IT services, process, and products and a reliable and easily accessible description of these data. In the end this question is related to the old question of how to represent knowledge and knowledge about knowledge or how to describe descriptions.

The OMG with UML and MOF

In information engineering, there are several answers to the question how to describe descriptions. It was the OMG (Object Management Group) which amalgamated three diverse modeling concepts (Booch, Rumbaugh, Jacobson) into one large "Unified Modeling Language" UML. The language and the accompanying documents are so big that (Wikipedia) "OMG was in need of a metamodeling architecture to define the UML" and created MOF – the Meta Object Facility.

To make a long story short, the meta modeling architecture is quite simple and straight forward. To avoid a meta-meta-inflation it was decided to define a four layer model with a distinct role for each layer.

The MOF-architecture was designed specifically to describe the UML-world. This architecture can be interpreted in a more general manner – as a blueprint for the structure of models and metaModels.

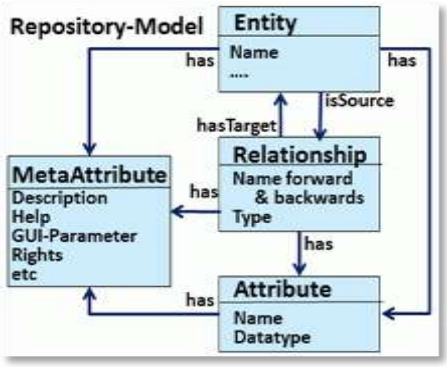


Blueprint for a generic Metadata Platform

Purpose A generic metadata platform acts as a "database for metadata and models" similar to a DBMS for business applications. It provides access services in multi-user-mode to the models and the data stored in the repository. A metadata platform (MDBMS or Repository) provides a flexible data infrastructure for IT-applications which manage the metadata of an organization. A MDBMS (Meta Data Base Management System) is a set of software programs that control the organization, storage, management and retrieval of metadata and their models in a metadata database.

Model and Scope A MDBMS exposes the view of an entity-relationship-attribute-model. A MDBMS stores models *and* metadata. MDBMS supports the versioning of the entity-instances.

Access A MDBMS provides services to application programs (API) to access the models *and* the metadata controlled by access rights. Several applications can access individual overlapping views of the metadata in parallel and are isolated from each other by a session/transaction mechanism.



Tools A MDBMS provides generic tools for the administrator and for the users to manage, query, import, export the models and the metadata.

Sizing A MDBMS is able to manage large, structured models *and* high volumes of Metadata: Medium to large organizations deal with terabytes of business data or data about IT-resources. The volume of metadata to control these data with a control span of 10^3 reaches millions of elements. The data volume of metadata typically ranges from several Megabytes to Gigabytes, but not Terabytes. The number of entity-instances and relationships can reach several millions. The structure contains multiple models which must be kept logically consistent. The models may reach a size of hundreds of types.

Federation Multiple instances of data bases can be defined and accessed in parallel by an application in one session. A "yellow-page"-database can be created to store the consolidated models of all instances and serve as a model of models.

MDBMS vs Other vs Relational Database: A MDBMS is structured to reflect an entity-relationship-model and stores structured self-described models (metaAttributes) and metadata. It integrates model manipulation and data manipulation. A RDMBS in contrast uses a relational model; model description is very restricted and model manipulation is separate from data manipulation. A RDBMS is designed for business-data and can manage very

large databases – far beyond the scope of an MDBMS. It is designed for data of the M0-level and contains the bare minimum of M1-information.

vs. Hybernate generates static data access code for a defined data model. Data definition and data manipulation are completely separate tasks. Each change in the data model requires not only a new code generation run but also an unload / restructure / load of the data base. A metadata platform integrates the data definition and data manipulation and interprets the data model dynamically. Changes to the data model can be done on the fly.

vs MOF/UML: The main difference between a MDBMS and the MOF/UML-structure is the physical implementation: It stores models and data as dynamically accessible (via an API) storage structures in contrast to the language based structures of MOF/UML. The UML-tools operate exclusively on the M2-level with internal M3-information.

Data dictionaries, CMDBs operate on the M1-level with a rather fixed model on the M2-level.

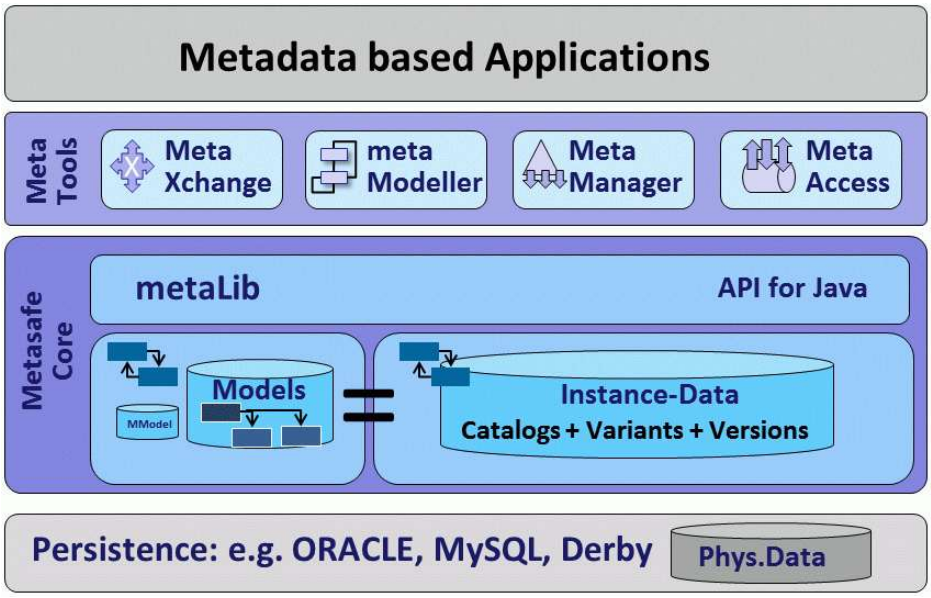
Take Away A logical repository covers the layers M1 to M3. It is based on a common data model and can serve as generic metadata platform. Hence, the implementation of a metadata platform is the first, but essential step towards industrialized and agile IT services, processes, and products.

The Metasafe Metadata Platform

The Metasafe Metadata Platform provides an easy and efficient data infrastructure for any metadata based application. It is constructed as a service package between the application layer and the persistence layer.

Hence, the Metasafe Metadata Platform reduces the complexity when consolidating various data models and enables IT organizations to build common data models not only for all existing data models of IT applications in an enterprise, but also to easily expand the common data model when data models of new, additional IT applications have to be integrated. It empowers the first and most important step towards an industrialized and agile IT.

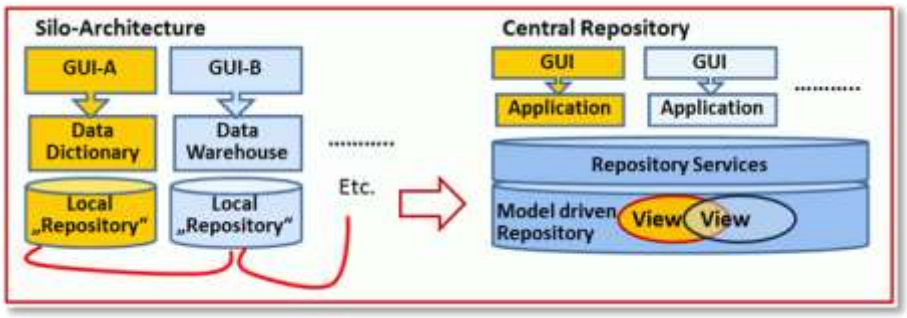
The Architecture of Metasafe



Applications

IT-applications like a CMDB, a data dictionary, a tool for risk management, a portfolio management system, an application architecture system etc. have one common characteristic: They deal with complex, fine grained data and need a flexible data platform. Metadata driven applications access shared metadata and use the models to control, present or navigate the metadata. Metasafe was designed bottom up to support this kind of IT-applications.

Substantial gains can be achieved by moving from the old silo structure of applications to a layered model-driven structure. These IT-applications are more user-friendly and faster and easier to develop.



Each "application" accesses only the data which are described by an application specific view. This view concept reduces the complexity for individual applications and provides a protected and controlled access also to shared data. The resulting applications are much smaller, easier to develop, easier to maintain und more flexible.

Persistence

Metasafe stores the models *and* the metadata in a secure and controlled manner: A real database is used - instead of lots of individual XML-Files or Excel-Sheets or private catalogs. Metasafe packages and compresses the data and store them in a relational data base.

The DBMS serves as a physical data store – the data in the database are protected against uncontrolled access with DB-tools.

metaLib

Metasafe provides an API (application programming interface) with a rich set of functionality to access the models *and* the metadata in a meaningful and efficient way. The functions range from basic access functions to complex services. Developers use the API to create generic IT-applications faster and cheaper.

Metasafe Core

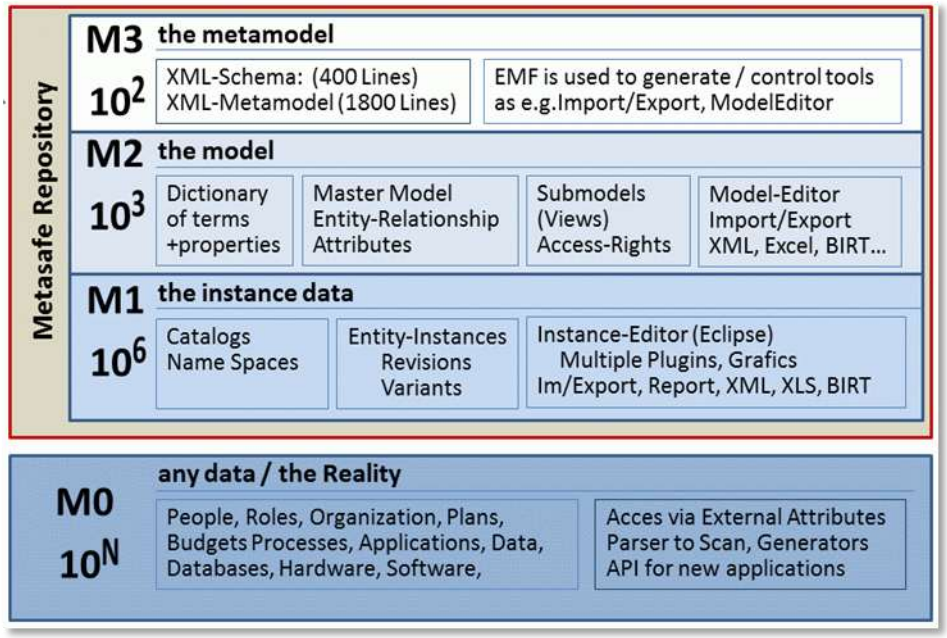
The metasafe core manages the layer M3, M2 and M1 of the 4 layer model. Each layer has a precisely defined function and associated services and data structures. In addition there is a direct interface to the M0-layer via the so called “External Attributes”. The metasafe core maintains the inner structure and isolates the sessions of several users from each other. The core invokes the persistence interface to store or retrieve data from the persistence database.

Example: Information Dictionary

An information dictionary (-> COBIT PO2 process) to support data governance could an example for an application. The relevant area of interest on the M0 layer comprises all data bases and files and interfaces etc.

The M1-layer contains all the business objects with their properties and relationships to data which encode these properties. Consequently this includes the descriptions of the databases and their tables, the record-descriptions and interfaces, the persons who are responsible (RACI), references to all programs which access or process these data etc. The volume of these highly interconnected metadata can amount to millions of individual entities and relationships. The M2 layer could be a model (or submodel) with some hundred elements (entity-types, attribute-types and relationship-types). See e.g. OMG Knowledge Discovery Meta-Model (KDM).

Metasafe – full integration of three Metalayers



10^N span of control

These numbers indicate the size of each layer in elements: The M0 layer contains terabyte of potentially relevant data. The M1 layer of a single repository instance is used to manage data about a specific area of interest of the M0-layer. The To control millions of data on the (M1) instance level a (M2) model could consist of thousands of elements, which in turn are described by a (M3) metamodel which is made up of a small number (100) of elements. The layers are abstractions and provide a means for a gradual reduction of the span of control. From the perspective of a model-driven architecture the levels M1 to M3 are tightly coupled in contrast to level M0 which is loosely coupled.

M3 metamodel

The metamodel is used to bootstrap the system core. It describes the basic components of the M2-layer, their attributes and their relationships with an XML-document. The XML-Schema is used to describe the structure of the XML-document to make sure that changes to this document do not introduce any inconsistencies. These documents are used for code generation (static) and to control the M2-level tools (dynamic). This innermost core of the

system is not accessible for the repository user and serves exclusively as an infrastructure for the development of the repository itself.

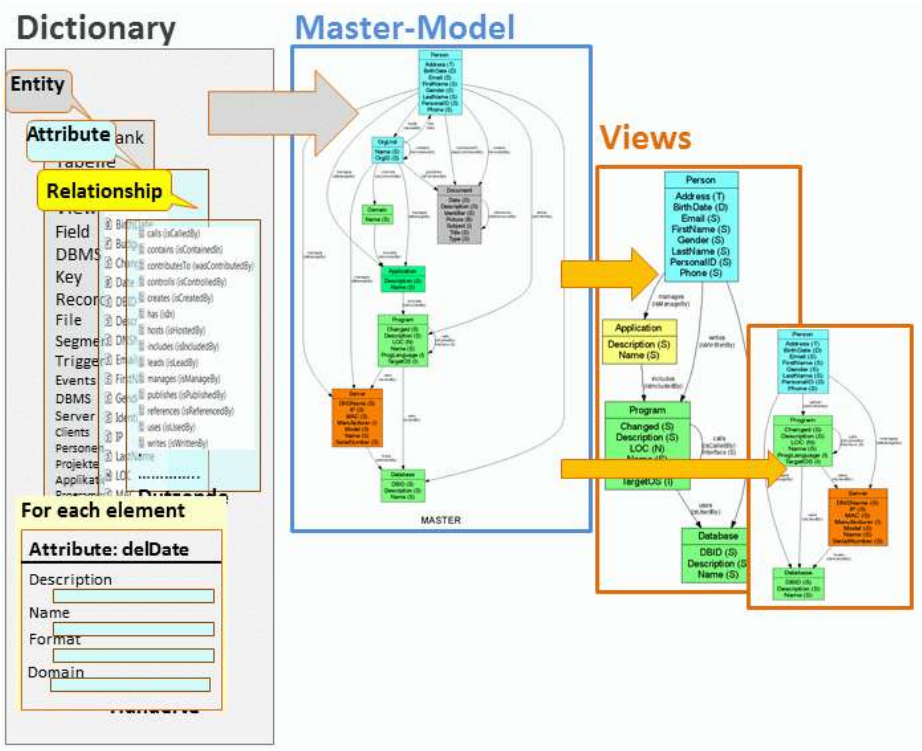
M2 model

The model layer contains the dictionary, the master model and submodels. The contents can be created with the metaModeller Tool or by any other program using the API. The metaModeller in this sense is just 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 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.

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



Strict rules are enforced by the system to keep the models and the data consistent; these rules are controlled by the core based on the definitions of the metamodel. The core checks the consistency whenever a change request is issued: e.g. a request to add an attribute type to an entity in a subschema is accepted by the core only if it is defined in the dictionary and in the master schema of this entity.

It is this elaborate structure which allows to develop really large models and consistent for individual purposes. Classical data modeling tools have only a

myopic view of submodels, but do not support an integrating mastermodel and dictionary. It is virtually impossible to keep these models consistent.

M1 Instances

The instance level contains the instances of the metadata. It is structured into separate storage units called “Catalogs”. For each entity multiple “variants” and “revisions” can be instantiated. Variants are explicitly named versions of an entity and can e.g. be 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.

Summary

Agile and industrialized IT Management creates sustainable value to the business. It is a best practice and gives the ultimate answer to the classical question: Does IT matter?

The main blocker to agile and industrialized IT Management is a fragmented landscape of various IT systems. This silo-type structure causes the troubles IT is suffering from: redundant data, inconsistent data, inaccurate data, narrow view, lack of standards. Fragmented data about IT systems means fragmented IT processes, inconsistent services and products that are not delivered in time. Business suffers from lack of speed, lack of flexibility, and lack of quality.

The solution to fragmentation is a common data model for all IT applications. It can be designed and implemented by a metadata platform. This idea follows the OMG MOF-architecture. The result is a logical repository that empowers an iterative migration of all IT applications’ data model into a consolidated common data model. Now, industrialized and agile IT services, processes, and products can be delivered. Now, IT matters.

About Metasafe

Contact: info@metasafe-repository.com

For more information see www.metasafe-repository.com