Describing Architectures Using RM-ODP

Thomas Kudrass
UBS AG, Postfach, CH-8098 Zurich, Switzerland
thomas.kudrass@ubs.com

This paper describes how the RM-ODP approach for system specification is applied to the description of the IT architectures of a large Swiss bank. The IT organization of UBS AG is defining several architectures based upon a business model which describes the main objectives of the bank considering the IT requirements. The semantics and the components of each architecture are explained using the basic five viewpoints of RM-ODP. Besides the viewpoint concept it is demonstrated how the relationships among all architectures and their components can be specified from an information viewpoint. Additionally, using the General Relationship Model (GRM) has proved its benefits in order to express the semantics of those architectures.

1. Introduction

According to the mission of the IT organization, information technology is considered as a key enabler for the banking services and products. IT supports the business units and the corporate center in their role as application owners through the entire life-cycle of applications development from new business requirements formulation up to its retirement. In order to provide these services to the different lines of business, IT operates an efficient, reliable IT infrastructure with high performance, utilizing competitive production facilities, the necessary processes, standards and the development resources with both banking and technological know-how.

Therefore, a number of IT-related architectures is defined in the IT organization. These architectures act as reference architectures which are obligatory for the development of new systems. The main purpose of the architectures is to ensure the re-use of existing IT components, be it business data model artifacts, development patterns or some technical components. All projects have to pass an architectural sign-off procedure proving that they are architecturally compliant. This may add another dimension to the fulfilment of the local business requirements by the respective projects. The sum of all architectures can be interpreted as a set of business and IT requirements which are globally valid and need not to be re-invented.

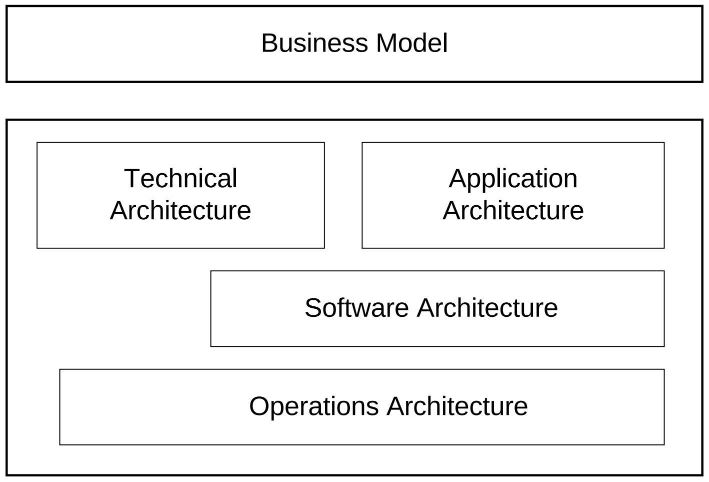

Figure 1: Architecture - Overview

Usually, the whole picture shows a layered architecture which suggests an increasing degree of abstraction dependent upon the position of the architecture in the picture. It is widely accepted that there are interfaces among the various architectures as an architecture may pose requirements to another one. What is obviously missing in the figure is the specification of relationships among these architectures. To figure out this is quite difficult, however, their semantics resembles realization relationships as they have been described in [KA98].

RM-ODP and GRM were chosen, among other things, to provide for abstraction and precision which are essential for understanding, and to reuse very general and generic patterns of reasoning already defined there. The approach of using the RM-ODP viewpoints should help to answer the following questions:

- Are there any criteria to decide what belongs to what architecture?
- What are the main parts of each architecture?
- What is the role of each architecture within the IT?
- How can the relationships among these architectures be described in a way that is generally applicable?

The rest of this paper is structured as follows. Section 2, shortly introduces concepts of RM-ODP which will be used later on. Section 3, briefly outlines the architectures and shows the possibility to exploit the idea of viewpoints at all levels of IT. Section 4, characterizes the different architectures by describing them in terms of all basic five viewpoints. Afterwards, the main relationships among the architectures are described using the GRM in section 5. The conclusion in section 6 summarizes our experience when using the RM-ODP approach.

2. What does RM-ODP provide?

2.1 Basic Definitions

According to RM-ODP, an architecture of a system is defined as "a set of rules to define the structure of the system and the interrelationships between its parts" [ISO95a].

Furthermore, RM-ODP defines a system as "something of interest as a whole or as comprised of parts" [ISO95a]. It can be a concrete or abstract thing. A component of a system may itself be a system, in which case it may be called a subsystem.

2.2 Viewpoints

The division of labor is typical for complex systems in traditional business and applies even more to today's complex information management system. Therefore, it is good practice to separate concerns which leads us to the introduction of different viewpoints from which a system (not necessarily an IT system) can be considered and described.

RM-ODP defines five basic viewpoints of a system and its environment. They are:

- Enterprise Viewpoint: Focuses on the purpose, scope and policies of a system.
- Information Viewpoint: Focuses on the semantics of information and information processing.
- Computational Viewpoint: Focuses on the distribution of a system through functional decomposition into objects which interact at interfaces.
- Engineering Viewpoint: Focuses on the mechanisms and functions required to support distributed interaction among objects in the system.
- Technology Viewpoint: Focuses on the choice of technology for a system.

Each of these viewpoints provides definitions of concepts and rules for the specification of a system from the corresponding viewpoint. Each viewpoint has to be described in a certain viewpoint language. All viewpoints have a common foundation and therefore all viewpoint languages use common concepts and constructs as well as viewpoint-specific ones. For example, the information viewpoint is typically described through data or object modeling, whereas the computational viewpoint may be specified through interaction diagrams. It is also allowed to use some language, which is typically associated with other viewpoints. Take as an example the specification of the enterprise viewpoint, which could be written using in part data modeling notation.

2.3 General Relationship Model

The General Relationship Model (GRM) [KR94, ISO95b] can be used to describe relationships among things like a system. The GRM encourages the definition of generic,

reusable relationship classes applicable to multiple applications. Some relationships are generic and apply everywhere (composition, subtyping, reference) but there also exist other (non-generic) relationships which are defined in the same manner (e.g. a "realization" relationship, [Kil99]).
We have to focus on the composition relationship (**C**) which can be encountered very often. According to RM-ODP, a composition of objects is defined as a "combination of two or more objects yielding a new object at a different abstraction level of abstraction. The characteristics of the new object are determined by the objects being combined and by the way they are combined" [ISO95a].
Besides the composition the reference relationship (**Ref**) is needed. The semantics of a reference relationship is that the referencing object determines properties of the referenced object, which can exist independently from the referencing object.

3. Overview

3.1 Architectures

The strategy of the bank (i.e., of their divisions) constitutes their **Business Model**. The business principles at a strategic level are reflected in the main elements of the business model. Determining the strategy requires to analyze the current situation. So, the environment of the bank has an impact on its business model. This comprises legal stipulations, which have a direct impact on the principles and rules as well as the position of the bank in the market and their business relationships with other partners.

It is widely acknowledged that the **Application Architecture** of IT is dependent on the business model. The semantics of this dependency becomes more evident by the use of RM-ODP. The application architecture describes a partitioning of business-oriented functionality into models according to the different RM-ODP viewpoints. These models act as reference models with elements that serve as basic building blocks, which appear as stable properties of the business specification for the overall IT. Furthermore, the application architecture provides a set of principles and rules representing high-level requirements, which can be arranged according to the RM-ODP viewpoints. Some of them have their origin in the principles stated in the business model while others are IT-related.

The **Software Architecture** as all other architectures is considered as a set of rules that are obligatory for the software development.
These rules can be expressed as guidelines and patterns for different scenarios of software development (e.g., how to integrate a third-party product, how to design an OLTP application system?) and may result in application infrastructure components which are generally available. The application infrastructure components must not add any business requirements as they have already been captured by the application architecture but provide a benefit by "componentizing" them and ensure a uniform behavior from the business perspective. For example, it is desirable to have a centralized input/output management or a

general historization of data. An integral part of the infrastructure software architecture provides is a catalog of patterns and their supporting software frameworks in order to guide and foster the development of components.
Roughly speaking, the main focus is on the HOW to develop - not on WHAT. Whereas the software architecture deals mainly with the computational and the technology viewpoint within the scope of IT, it can itself be divided into the five basic viewpoints of RM-ODP in the same way in order to characterize its parts.

The **Technical Architecture** prescribes a collection of products (basic platforms) and a set of rules and patterns that describe how to apply these products. The main difference with respect to software architecture is that these rules are product-specific (as far as rules have been defined). On the other hand, the technical architecture is more generic than software architecture since it is used for all applications.

The **Operations Architecture** describes the physical elements and processes needed in the IT operations department in order to run the software systems of the bank. It is possible to analyze each process of the operations architecture in terms of viewpoints.

3.2 Nested Viewpoints

Figure 2 shows how RM-ODP viewpoints are applied at all abstraction levels. From the bank business' point of view the entire IT appears as technology without considering its internal structure. But even other parts of the bank can be considered technology viewpoint. A similar patterns applies at the level of IT, which can be further subdivided into the basic five viewpoints. We can show how our architectures are mapped to these viewpoints within the scope of IT. Going one level down (with the example of application architecture) we can also specify viewpoints and relate them to the architecture's components. As we will see later on, the transition of one architecture to another one is often via the technology viewpoint (e.g., software architecture is considered technology viewpoint from the application architecture perspective).

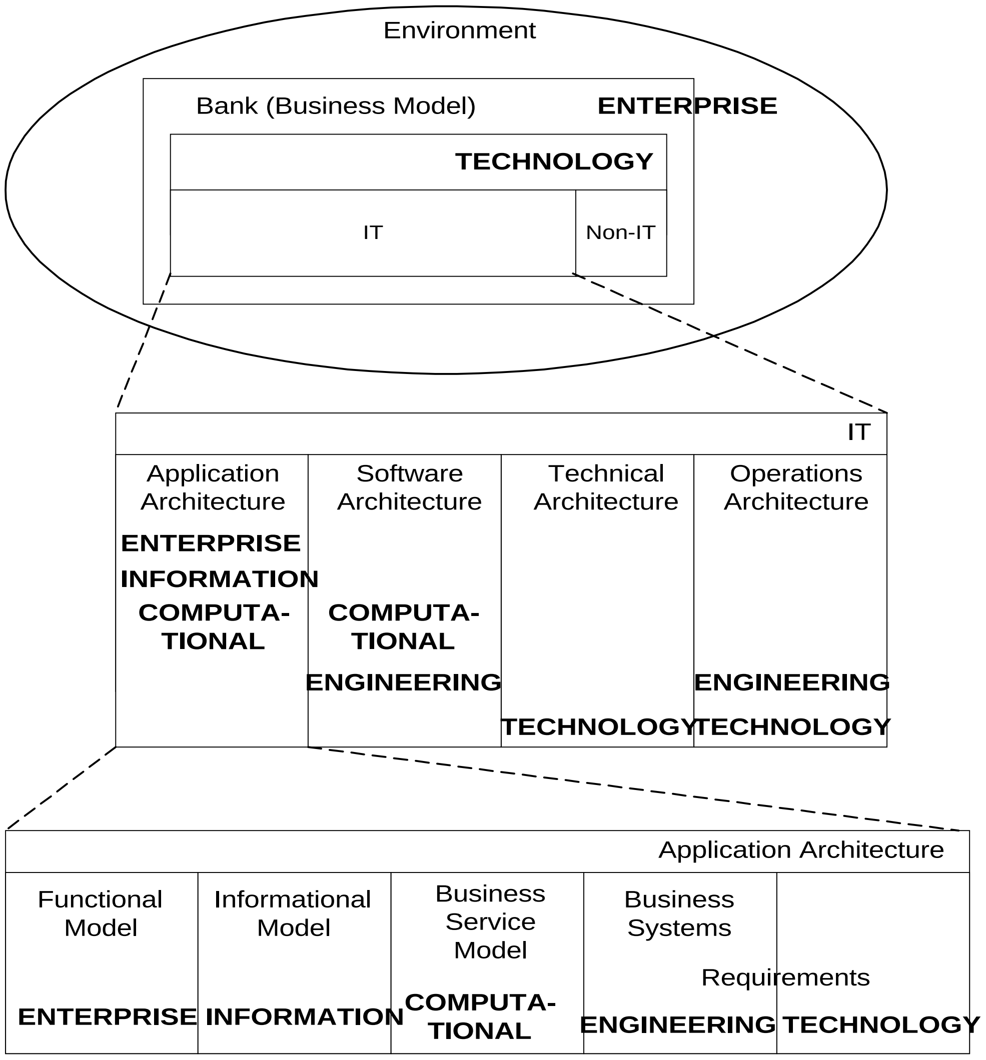

Figure 2: Nesting of Viewpoints

4. Definition of Architectures Using RM-ODP Viewpoints

4.1 Business Model

Enterprise Viewpoint

The enterprise viewpoint of the business model describes the bank's tasks and medium and long-term targets to be achieved that result in a set of business functions. The aim of the bank is to sell products to customers realizing a certain profit under consideration of the environment. This presupposes analysis of the requirements of the customers. It also includes policies (for example, bank's obligations and prohibitions). The organization model with organization principles and role descriptions is also considered part of the enterprise viewpoint without reference to the current organization structure.

Information Viewpoint

In order to realize the goals of the bank, the business model captures invariants that describe things and relationships such as products and services based on a classification of products into groups. Accordingly, pricing rules, rules for the definition of customer segments, customer acquisition and assignment of customers to product groups have to be established.

Computational Viewpoint

The computational viewpoint deals with "interesting" behavior which can be interpreted here as the main business processes at a general level (e.g., payment and mortgage loan processing). These business processes already consider general business rules (e.g., "do the availability check first before executing the client's order") which are considered preconditions in RM-ODP. In order to define business processes a role concept is required.

Engineering Viewpoint

The engineering viewpoint of the business model deals with aspects like distribution. Among them is the choice of distribution channels for certain product groups and targeted customer segments.

Technology Viewpoint

The technology viewpoint describes the properties of the "boxes" used for the implementation independently of whether these boxes are realized by people or IT systems. It is possible that some parts of the implementation are outsourced to external service providers. For example, this could be the development of new software or the operation of the IT infrastructure. The contribution of IT represents a considerable amount of the technology viewpoint which has to be analyzed in much more detail in the other architectures.

4.2 Application Architecture

Enterprise Viewpoint

The enterprise viewpoint of the application architecture comprises the business functions which characterize the banking tasks (independent from specific organizations or processes). The hierarchical decomposition of business functions into lower-level business functions forms the functional model of the application architecture. The semantics of the relationships between the business functions within the hierarchy has already been investigated in some more detail [BK99].

Information Viewpoint

The informational model established in the application architecture represents the conceptual model of the business information, which is currently described by an Entity Relationship Diagram at different refinement levels.

Computational Viewpoint

The computational viewpoint is covered by the specification of business services, which can be seen as implementation of elementary business functions. The behavior of the service is specified by pre- and postconditions that are provided in addition to a service signature. This has to be in accordance with the main business processes of the business model.

Engineering Viewpoint

The engineering viewpoint of the application architecture is addressed by a partitioning concept which introduces business systems. Business systems are bundles of business functions according to organizational or functional criteria. For example, there could be a business system, which clusters all business functions, which are mainly related with the functionality needed by the bank towards the retail customer (Business System „Front Retail"). Business systems provide and require business services, and their interaction is the subject of the computational viewpoint.

Another engineering dimension, which puts more emphasis on IT, is the assignment of the business services to three logical layers (enterprise, intermediate, presentation). While the enterprise layer deals with information, the intermediate layer hosts business logic, which is mainly specified in the computational viewpoint. Presentation service is not subject of the application architecture because its focus is on banking functionality without considering presentation aspects.

Technology Viewpoint

From the application architecture perspective a business system is a composition of software systems which implement either business functions or data maintenance. From the application point of view they appear as black boxes.

4.3 Software Architecture

Enterprise Viewpoint

The enterprise viewpoint is determined by design requirements of software engineering and non-functional technical requirements. Among the design requirements are for example:

- flexibility (better modifiability)
- composability (design components in a way that software systems can be built by assembling them)
- adaptabiliy (design in a way independently from the underlying technology)

The technical requirements address availability, manageability, performance and fault tolerance and apply to the infrastructure components provided by software architecture.

The requirements as sketched above result in some basic principles for structuring software systems such as re-use of services and components, single point of reference ("one fact at one place"), separation of the interface of a service and its implementation.

Information Viewpoint

The elements introduced by the software architecture and their relationships can be described from an information viewpoint (e.g., Software System, Component System, Business Rule), which could lead to a meta model of the software architecture.

In the same way, the structure of the components of the application infrastructure and their relationships can be described from an information viewpoint (e.g., exception handling, audit, resource usage, business day controlling, historization).

Computational Viewpoint

Software design styles can represent patterns of interaction. For example, rules have to be established for the messaging principles (synchronous vs. asynchronous). The same applies to the interaction of a system with the user (batch processing vs. interactive dialog system). Further aspects are principles of the interaction between systems and with the environment outside the bank.

The behavior of the application infrastructure components can be described from a computational viewpoint. Dependent on the chosen design style for communication (messaging vs. event-driven) the interaction between the infrastructure components can be described as a flow of messages or a publish & subscribe structure when using an underlying event service.

Engineering Viewpoint

From an engineering viewpoint, software architecture defines design styles for the construction of software systems (which are not necessarily object-oriented). A typical design style characterizes the distribution of functionality into three logical tiers.

Technology Viewpoint

The components and services provided by the software architecture are at a logical level. They are independent from special hardware or software platforms but pose some requirements to them.

For example, it can be stated that a relational DBMS with certain capabilities is needed in order to implement a persistence service. Here, the "relational DBMS" is considered technology viewpoint from the software architecture perspective.

4.4 Technical Architecture

Enterprise Viewpoint

There are several non-functional requirements to the infrastructure that is provided by the technical architectures. Among them are:

- processing capacity
- response time

- availability (e.g., 24 hours a day)
- recovery from failure (time to recover, extent of data loss)

These requirements can be seen as decision criteria for the selection of certain products that are declared platform standards for the development of business applications and common services (e.g., communication services). The scope of the technical architecture comprises: hardware, operating system, database systems, communication, transaction management and middleware software. Additionally, policies (obligations, prohibitions) as well as purpose and scope with respect to the respective infrastructure are part of the enterprise viewpoint.

Information Viewpoint

The elements of the technical architecture can be described from an information viewpoint which includes precise specifications of invariants represented graphically (in figures) and in any other appropriate manner. For example, it might be difficult to provide a graphical abbreviation for all application-specific invariants.

Computational Viewpoint

The computational viewpoint is focused on platform-specific rules for the use of a certain platform. An example is a decision matrix for the choice of a certain communication product. Possible decision criteria that affect the selection are data quality or data transfer rate. Note that rules of the technical architecture are product-specific. For example, the prohibition of updates on views in relational databases is considered part of software architecture because it addresses the general view-update problem, which is not product-specific. Yet, rules how to apply certain features of a product are considered part of technical architecture (e.g. "don't use the field mode of the ATI communication product").

Engineering Viewpoint

The engineering viewpoint includes distribution, checkpoints, and related issues.

Technology Viewpoint

Although the products and their features are known within technical architecture (because it is its business) it is abstracted from the fact that all products have to be available as installed software systems. The details of versioning and packaging are not the subject of the technical architecture but are considered as technology viewpoint.

4.5 Operations Architecture

Enterprise Viewpoint

The enterprise viewpoint addresses the main targets and systems management functions, which are required from IT operations. From a development perspective the software systems are just logical systems characterized by the functionality they provide. Only when being operated they become physical systems as installations of logical systems. From the business perspective IT operations is responsible to fulfil the non-functional requirements for each system using their infrastructure. All requirements result in IT service contracts with the customers that stipulate the single tasks and the quality of the service.

When decomposing the systems management functions into a lower detail level it is possible to describe each of them from an enterprise point of view as well.

Information Viewpoint

The operations architecture has an information model representing meta data about managed objects and their relationships. Managed objects comprise hardware, software, data, network infrastructure and services. It happens that some elements of the information viewpoint here details certain concepts used in the technology viewpoint of the technical architecture. As an example for this, a standard software platform providing a common service (DBMS providing persistence), which is technology viewpoint in the technical architecture, turns to be an element of the information viewpoint in the operations architecture as far it becomes a physical (i.e., installed) component of the IT infrastructure of the bank.

Computational Viewpoint

The computational viewpoint describes in a process model the interaction of the main processes of the systems management such as configuration management, software control and distribution, monitoring etc. Each process can be further refined by decomposing it into smaller functions, activities and data flows.

Engineering Viewpoint

The engineering viewpoint is reflected in all processes because a distribution dimension has always to be considered.

Technology Viewpoint

The technology viewpoint describes requirements to tools for the automation of each systems management process.

Take as an example a platform for a configuration management database or a system monitoring tool.

5. Architectures from an Information Viewpoint

One issue to be resolved is that of making the semantics of the relationships among architectures explicit. The discussion in the previous section has shown that the anatomy of architectures has to be considered. Otherwise we would draw too simple conclusions about the interplay of our architectures (cf. figure 1).

As figure 3 suggests, the components of the business model reference components of the application architecture (AA), i.e., their properties have a direct impact on the AA components.

By posing requirements to IT application architecture itself references the software architecture which can be decomposed into three main parts. The *Design Styles* component of the software architecture defines in a generic way the foundation for the other software architecture parts which results in a reference relationship with them. The application infrastructure provided by software architecture can also be interpreted as a set of components which requires a technical platform provided by IT products of the technical architecture. The *Design Styles* do also impact the policy how to use an IT product. That means, generic guidelines have to be translated into concrete rules that apply to the respective product.

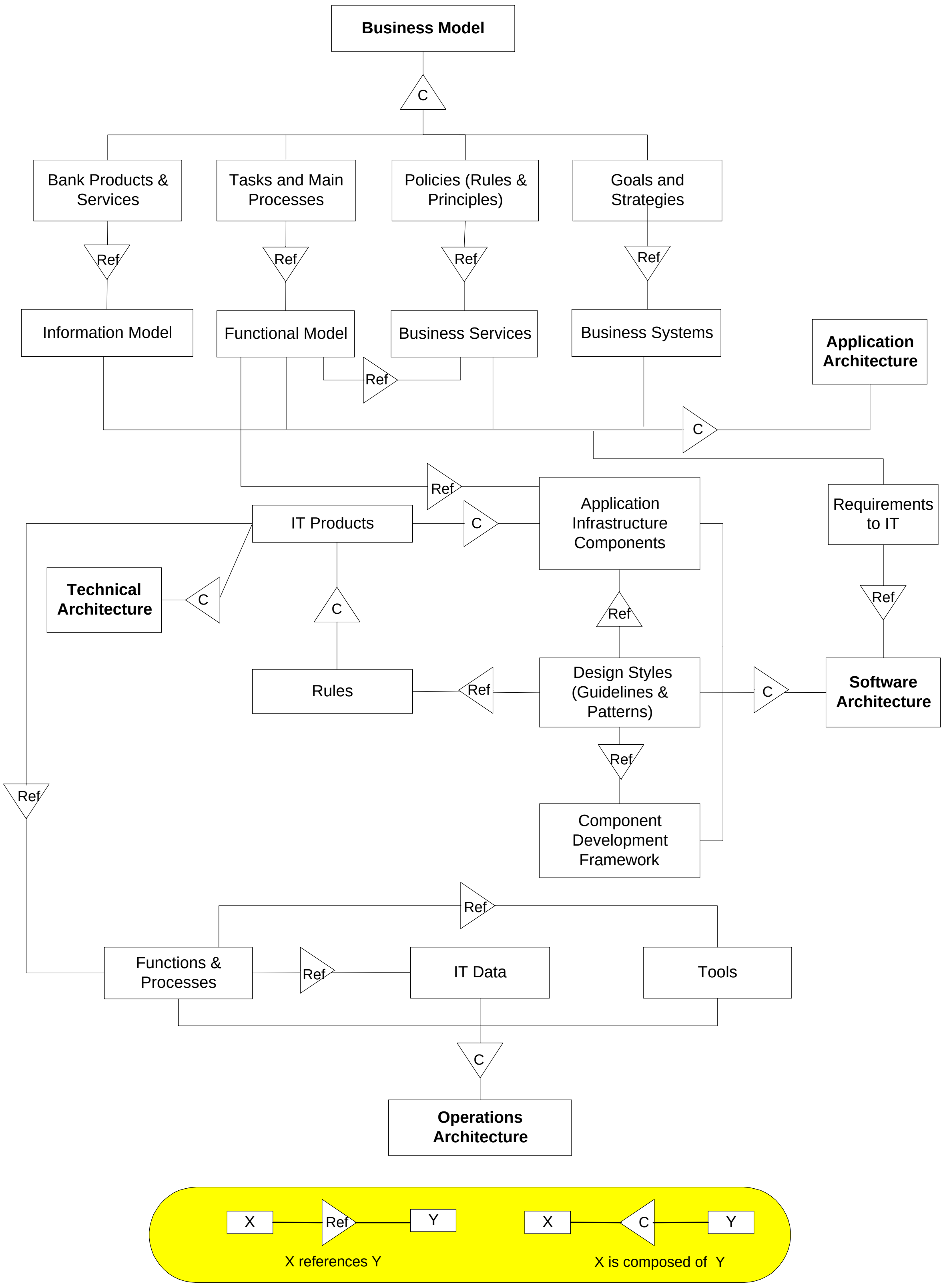

Figure 3: An Information Viewpoint on Architectures

Once the IT products have been identified as part of the technical architecture, they become the subject of IT operations' processes and, therefore, reference them.
The other components of the operations architecture, data model and tools, are derived from the functions the IT operations organization has to fulfil.

Note, that figure 3 is focused on functional business requirements. Besides, there are non-functional requirements that can be derived from the business model and reference particularly the properties of the technical architecture. Further, the technical architecture determines both restrictions and opportunities for the software architecture and, transitively, for the business (not shown in figure 3). The same approach is also true about the relationship between the software architecture and the business-related architectures.

6. Conclusions

- ***Think in viewpoints to understand the semantics***

It was possible to outline each of those architectures (in a more informal way) in terms of the RM-ODP viewpoints, which helped to understand the semantics of each architecture better.
There have been two cases:
Mostly, single parts of architecture could easily be assigned to a certain viewpoint because their semantics was obvious. Take as an example the information model of the application architecture, which could simply be mapped to the information viewpoint of RM-ODP.
In some other cases a deliverable of architecture embodies different viewpoints, e.g., the description of the technical architecture is a combination of information and computational viewpoint in most cases (the structure of a system together with interactions between them).
Also, relationships are essential to understand components; by using relationships we can understand, for example, the similarities between differently named but similar things (including components); and the differences between similarly named but different things.

- ***Viewpoints can be nested***

As it has been shown, viewpoints can be defined within a certain scope, e.g., in a given architecture. A viewpoint may be expanded into the specification of a new system at a different abstraction layer with the complete range of all viewpoints but restricted to the scope of this system. Take as an example the technology viewpoint that addresses mainly the implementation that becomes the business of the underlying system.
So we end up in a nesting of viewpoints which can be traced from the business targets announced by the CEO to a certain version of a software package.
On the other hand an architectural component can be resolved into viewpoints in order to refine its description.

- ***Understanding semantics helps to find borderlines among architectures***

By following the approach of separating concerns according to different viewpoints it results in a separation of parts which belong logically to different architectures. However, it is still practice to describe them at the same place, e.g., rules for logical data modeling (software architecture) together with a list of prescribed DBMS (technical architecture).

- ***Understanding relationships by understanding components***

We have shown that it is very useful to look at the components of architectures in order to understand their relationships because they can be much more manifold than one would expect.

The presented paper can be interpreted as an overview paper on IT architectures which outlines the opportunities of RM-ODP. It reflects the author's position based on contributions from architectural teams of different areas. It sketches the first steps to specify architectures but much more work is required.
In order to gather more experience with RM-ODP the application architecture has been chosen as a candidate to investigate the use of RM-ODP in much more detail. The contents of the application architecture provides elements belonging to all viewpoints together with a meta model which is being aligned with the RM-ODP standard (for some more interesting details see [BK99]).

Acknowledgements

I would like to thank Haim Kilov for encouraging me to work on this topic and providing me constructive advice and suggestions. Thanks to UBS AG to support the work and to my colleagues from the bank, Moheb Zaki for helpful discussions and reviewing this paper, and Othmar Bernet for his contributions.

References

[BK99] O.Bernet, H.Kilov, *From box-and-line drawings to precise specifications: using RM-ODP and GRM to specify semantics*, Proceedings of the 8th OOPSLA Workshop on Behavioral Semantics, Denver, 1999.

[ISO95a] ISO/IEC JTC1/SC21, *Open Distributed Processing - Reference Model - Part 2: Foundations*, IS 10746-2/ITU-T Recommendation X.902, 1995.

[ISO95b] ISO/IEC JTC1/SC21, *Information Technology. Open Systems Interconnection - Management Information Services - Structure of Management Information - Part 7: General Relationship Model*, 1995 (ISO/IEC 10165-7.2).

[ISO95c] ISO/IEC JTC1/SC21, *Open Distributed Processing - Reference Model - Part 3: Architecture*, IS 10746-3/ITU-T Recommendation X.903, 1995.

[KA98] H.Kilov, A.Ash, *An information management project: What to do when you business specification is ready*, Proceedings of the 2nd ECOOP Workshop on Precise Behavioral Semantics, Brussels, 1998.

[Kil99] H. Kilov, *Business Specifications - The Key to Successful Software Engineering*, Prentice Hall, 1999.

[KR94] H.Kilov, J. Ross, *Information Modeling: an Object-Oriented Approach*, Prentice Hall, 1994.