Multimodel Application Specification, Integration, and Evolution − Experiences from the Reinsurance Industry

Hans Wegener

Group Information & Process Architecture
Swiss Re
Mythenquai 50/60
CH-8022 Zürich
Hans_Wegener@swissre.com

Abstract: Swiss Re has established an architectural framework, tools, and concepts that enable business people to specify their view of the business in a consistent and formal manner. Based on repository technology, this business model is refined for each application. The architectural framework ensures that a specific list of integration and evolution problems are either solved or their complexity reduced. Our experience show that this generally yields very satisfactory results, but that the problems intrinsic to a large, globalized company (existence of legacy systems, different jurisdictions) are still felt strongly.

1 Introduction

A reinsurance company, or reinsurer, insures insurance companies, also called (primary) insurers. Much like insurers assume risk for a fixed price from their clients, either persons or organizations, reinsurers assume risk from primary insurers. Depending on their needs, primary insurers mainly buy reinsurance to either assume a larger portfolio of similar small risks, and/or to pass on parts of a specific very large risk. For further information on the subject of reinsurance, we refer the reader to [Car00].

While both insurers and reinsurers assume risk, they are different in that a primary insurer is a retailer with a relatively large customer base and relatively few highly standardized products, while a reinsurer is a wholesaler with a relatively small customer base and relatively large number of products specifically designed for one client. As a result, the volume of data and transactions is not nearly as much of a challenge for a reinsurer as it is for a primary insurer, or a retail bank, for that matter. On the other hand, given product diversity, the size (and therefore clout) of many clients, and the global reach of the business, standardization is much more difficult on the data side as well.

At Swiss Re, complexity is felt very strongly at the corporate center, especially in financial accounting and management reporting. The recent emphasis on corporate governance and regulatory compliance demands a consistent perspective on the business be implemented by applications. The nature of reinsurance business, the existence of long-standing legacy systems, and the absence of simplification alternatives made it practically impossible to establish a joint, unifying architecture upon which both existing and future applications may be built. Hence, lacking a viable architecture upon which to

construct them, an architecture with which to integrate systems had to be sought instead. Therefore, our goal was threefold:

1. Establish and manage a consistent model of the business and its financial status to be specified by and communicated to reinsurance and finance professionals.

2. Implement applications based on a stepwise refinement of the above business model and support their integration by mapping the (implementation) models onto each other.

3. Decouple applications in a way that their interfaces prove stable under evolution of the above business model.

We have written earlier about our architectural framework [Mar03] and detailed our use of business terminology for modeling applications [WA03]. In this article we report about our latest experiences with application specification, integration, and evolution at different levels of abstraction.

The remainder of this document is structured as follows: Section 2 refines and extends our architectural framework with respect to the mentioned goals and elaborates our practical approach; Section 3 discusses some experiences made over the last year; and 4 summarizes the findings.

2 Specification, Integration, and Evolution Framework

2.1 Architectural Perspectives and their Governance

In this article we use the terminology of the Model Driven Architecture (MDA), as defined by the Object Management Group (OMG), to describe and delineate architectural concepts [MM03]. Generally, we distinguish between three models:

- a computation-independent model (CIM),
- a platform-independent model (PIM), and
- a platform-specific model (PSM).

The CIM is used to represent concepts important to business. These include, for example, the general ledger, risk model figure aggregation rules, carrier structure, or customer segmentation. The PIM is used to represent a refined application perspective on the business. This comprises, for example, business processes, change management, or access control. The PSM is used to represent the technology substrate applications are directly implemented upon. This includes, for example, user interfaces, relational databases, or messaging middleware.

For each of the above (CIM, PIM, and PSM), our company provides standardized artifacts to be used in design and development. Three different corporate teams are responsible for governing architectural practice regarding each model (information and

process architecture, application architecture, and technology architecture). Mandatory project milestones are defined to check project artifacts produced for architectural compliance.

2.2 "To Be" and "As Is" Models

In theory one could discard the CIM and just live with PIM and PSM. However, the distinction between CIM and PIM is important to us. To business, the concept of change is a remote one. Anecdotal evidence suggested to us early on that one has to separate between the way business people see current goings-on and their perspective on history. Especially in non-life reinsurance and financial accounting, where complexity is high and – literally – decades of business matter, this distinction is of crucial importance. Furthermore, as any bigger company in the financial services industry, we are plagued by legacy applications, which is a substantial investment not easily done away with.

To achieve a clean separation of concerns, the CIM deals as a carrier of the "to be" world, while the PIM is the place where "as is" abstractions and, most importantly, change are dealt with. The CIM is mapped to the PIM by a transformation mechanism to be detailed in Section 2.3. But there is not one single PIM, there are – literally – dozens. In addition to history and legacy systems, the global nature of our business adds jurisdiction to the equation. From the perspective of the corporate headquarters, this is not a relevant abstraction, either. Therefore, PIMs abound in that there is plenty of business abstractions in individual PIMs that in theory belong to CIM. Notably, this is less an effect of bad architectural management than of the intrinsic complexity of the business. As a result,

- the CIM abstractions are moderately complex and not numerous, but
- the number of PIMs is substantial and thus considerably increases complexity.

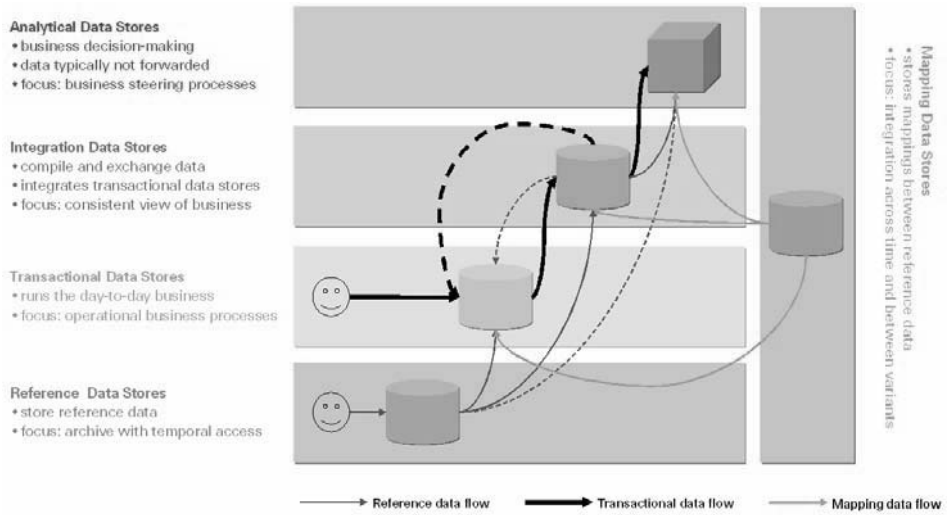

Figure 1: The integration architecture distinguishes five different types of applications/data stores.

Transactional, integration, and analytical data stores use reference data stores to pick their appropriate PIM. Based on information about application change adoption characteristics, the mapping data stores provide PIM-to-PIM mappings. Both reference and mapping data stores span the entire architecture. Dotted lines indicate optional data flow, the smileys indicate user interaction (i.e., input).

For example, the general ledger used for producing our annual report consists of about 900 entities (i.e., accounts) each split by about one or two handful from around 90 attributes (i.e., dimensions). However, these numbers are drastically inflated by the fact that variants of this report are used in about 20 other contexts, which are mostly produced quarterly, each with a different structure and constraints defined thereupon. Considering that this is only the financial accounting world, overall complexity becomes more apparent.

Architectural artifacts are thus specified at different levels of abstraction, with different models in mind. In order to manage them concurrently, our integration architecture had to handle three separate goals:

1. Define, structure, and declare constraints between business abstractions in a model. We designed a tool enabling business users to model them on the basis of business terminology. Thus they establish a CIM dubbed the *reference* model. The tool repository is fully historized, providing version-based as well as temporal access to its content.

2. Allow for variations of the business model in the way it is implemented. For different applications, the structure can be manipulated, detail taken away and adorned with technical representations of business abstractions. The resulting PIM is a *context-specific* model of the reference model. This is, again, provided by the mentioned modeling tool.

3. Provide mappings between different models based on their history or other metadata to enable application integration and model evolution. PIMs can be mapped to other PIMs only with reference to the CIM. We designed a second tool for mapping between different PIMs that makes use of the CIM for purposes of disambiguiation, which frequently occurs. Provision of mappings can still be manual at times, as some business decisions cause ambiguities that would not have been caused based on a purely formal approach.

The division of labor between historization, mapping, and integration is chosen for different reasons. Legacy applications have little choice but to operate on the premises they were designed upon. Therefore, the choice of change adoption has to reside with the application integrating others. Separating between historization and mapping enables us to avoid CIM bloat, which would be a usability hazard to business people.

2.3 CIM and PIM Abstractions

Our language for creating business models is the Swiss Re Data Language (SDL). As the name suggests, it is primarily designed to support modeling of content rather than behaviour. The SDL Tool is our repository and modeling environment that supports

users in designing and understanding the structure of the business based on the SDL, which has a history of four years. The most recent major release went productive in June 2003.

In the SDL, reinsurance and/or finance concepts such as type of claim, currency, account, industry segment, or investment category, are described by business terms. Each business term exhibits a descriptive name (evoking its meaning), and a formal definition (denoting its meaning). Name and definition are used to specify and/or understand the business semantics behind concepts. Depending on their context of use, business terms describe attribute types, attribute values, or entity types. They belong to the CIM.

Business terms form the terminal symbols in SDL. They are then structured, in our case hierarchically to form reference trees. A reference tree captures the segmentation of the business (into specific and generic concepts) as it should be from the corporate center perspective. Reference trees are represented by an attribute type and group together attribute values. They belong to the CIM.

Business terms can also be used to represent entity types. Because it emanated from the financial accounting world of our company, at this moment SDL only supports hierarchical relationships between entities (as in the general ledger). It is considered to extend this model to more flexible use of relationships as in entity-relationship modeling. The resulting structure, represented by an attribute type, is referred to as a custom tree. (The term custom tree stems from the SDL project and is actually based on a historic misunderstanding. It is a misnomer that happened to survive and become commonly used.) The values of the attribute type representing the custom tree act as entity types, which in turn compose a number of attribute types together. The resulting model element forms, again, part of the CIM.

Our carrier for forward-engineering PIMs out of CIMs is the context. A context can be anything from an application (which it normally is) to an industry standard to a mere illustration. It serves two separate purposes. First, it is used to associate business terms with a context-specific representation, the codes. Second, it is the platform for modeling context-specific variants of a reference tree, the filtered tree. As such, context are the place where metadata about structural mappings between CIM and PIMs resides. As the name suggests, a filtered tree contains a subset of the values of the reference tree it is based upon. Furthermore, the taxonomic relationships between the values are congruent with that of the reference tree in that a parent of a value in the reference tree must also be its parent in the filtered tree, as long as it also occurs there.

PIMs are administratively separate from the CIM, but their is a wide number of interdependencies. In order to minimize the potential harm that can be caused the SDL Tool checks overall repository consistency (CIM and PIMs). Hard constraints guard against violations of repository integrity, while soft constraints merely provide hints at potential inconsistencies outside the realm of the SDL Tool proper. For a more detailed discussion of the various constraints that apply in SDL we refer the reader to [WA03].

3 Experiences

SDL is a global Swiss Re standard that contributes to the production of some of the most important information artifacts of our company, notably the annual report. We were most successful in the financial accounting and non-life reinsurance areas, less so in life-and-health reinsurance, asset management, or financial investment. However, this may be more an effect of our own organizational focus on the former two groups than that of lack of usefulness.

3.1 Business Domain Modeling

We established a methodological framework which ensures that during analysis the artifcats relating to the CIM and PIMs are captured. Application interfaces are specified in terms of the SDL-IDs of the business terms used. Along with the chosen change model (temporal or version-based) it becomes drastically less complex to provide mappings from source or to target PIMs.

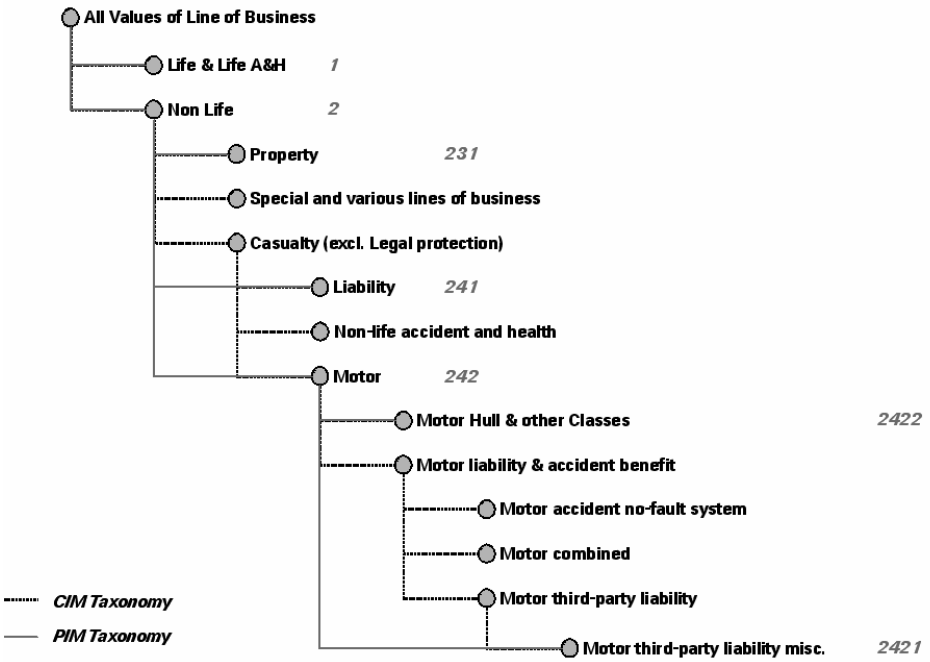

Figure 2: The line of business (extract) as seen from the corporate center perspective ("to be") and from the financial accounting point of view ("as is"). The numbers are the codes as they are used in the latter's context.

The SDL Tool currently has about a handful of content administrators that use it on a day-to-day basis, and a couple of hundred occasional (business) users that search and navigate the repository content. The tool's system interfaces are used by applications in various corners of the corporation to integrate with other applications. The CIM

comprises of an estimated 4'500 business terms, around 100 reference trees, and one custom tree. Here are a few modeling experiences:

Model Normalization From the perspective of the CIM, normalization is just as much a problem as it is in technical domains. The line of business currently consists of 741 business terms. Much of this detail (the reference tree spans 9 levels) can be attributed to specifics of facultative reinsurance, where proper premium pricing requires as much detail as possible. But still, in order to reduce complexity, a workgroup has proposed simplifications, which would create three more reference trees, but reduce the size of the mentioned one by about 30%. Not surprisingly, this being one of the main reference trees in the reinsurance part of the CIM, implementation cost is close to prohibitive. This demonstrates the value of the PIMs. As long as complete normalization has not been achieved, variability has to be supported. On the other hand, commonality in the upper levels of the line of business is still pretty good: the first 4 levels of the financial accounting and non-life reinsurance PIMs share 94% of the business terms; at level 5 the value already drops to 45%.

As another example, proportional reinsurance business typically exhibits an obligatory (treaty) type of agreement, while non-proportional business usually implies a facultative (individual) type of agreement. As an effect, people often do not distinguish the two, leading to business terms like "proportional treaty" or "non-proportional facultative". These simplifications occur frequently and are often caused by task-specific requirements due to the division of labor (in our and other companies) along products. Underwriters dealing with proportional/obligatory business typically do not have anything to do with non-proportional/facultative business and vice versa. Explicitly separating between type of business and type of agreement would force business users to make an artificial step caused by the underlying business model.

Jurisdiction is a tricky subject. In a globalized company, jurisdiction should not really play a role. However, it turns out that legislative standardization is still very patchy in financial services – even in the European Union. For example, many lines of insurance in Germany (e.g., fire) have to be reported by content of portfolio (e.g., business or private clients). We are unsure as to whether to reify jurisdiction as an abstraction at PIM or CIM level to reduce the number of PIMs around. This topic is under discussion. However, for the time being the line of business contains numerous business terms that could be simplified by extracting the jurisdictional aspects into an attribute of its own.

While in terms of access control, CIM and PIM administration is entirely separate, practice shows that the two get easily mangled by the administrators. Thinking is very application-oriented, model reuse (via the CIM) a thought still a little alien to business. However, there is slow movement in the (in our opinion) right direction. For about two years, IT has operated a team of experts to centralize administration in the hands of a few. This is slowly bearing fruit, much because the change process behind it is driven by that team as well, which is highly appreciated by business.

Model Simplicity vs. Model Expressiveness Our use of soft constraints was caused by the need to accomodate conflicting needs in the design of the CIM (and, as an effect, the PIMs). Many of the constraints are not generic to the entire domain. We had three choices:

1. express this in the form of additional CIM elements, thus adding to model complexity,

2. express this in only a few PIM elements, thus abandoning parts of our CIM-centric model,

3. not express this in any model, thus allowing for inconsistencies in some PIMs or the CIM.

We opted for choice three, because the other options were expressly rejected by business users as "too complex." Thus we ended up with a CIM and PIMs that were not expressive enough to model all the concepts existing in the real world. SDL Tool administrators are merely warned of (not prohibited from causing) inconsistencies. However, we did not yet run into any serious problems. Little (if any at all) confusion was caused by the missing model elements. It turned out that in the specific context of use there was so much redundant (i.e., tacit) knowledge available that the lack of expressed constraints did not really make much of a difference for model consistency. Expressing specific needs in the CIM can have a detrimental effect on other parties, while not adding critical capabilities for the other. This makes the CIM much less usable for many while not even really helping the remaining few. One can exploit organizational correlations between IT and business by leaving out model elements (i.e., non-generic business concepts), thus increasing the suitability for the task of model(s) for business users. For example, the homonym rate in model repository was thought to be a strong indicator of glossary intelligibility. However, in financial accounting we found that our request to eliminate homonyms was rejected because it was not percieved as useful.

A consequence of this kind of modeling are consistency issues. If the modeling environment is no longer capable of checking consistency, the model is incomplete with regards to the problem at hand. We have used different techniques to deal with the resulting design space (cf. [Weg01]), for example by having interfaces deal with additional repository object states in change management (see below).

3.2 Change Management

The CIM is defined as the model relevant to reflecting changes in the understanding of the business. Therefore, all CIM elements are versioned to the end of managing their lifecycle. Different changes can occur to the structure of a reference tree. Attribute values can be added, removed, moved, promoted, demoted, shifted, merged, or split. Business terms themselves can be changed as well, notably their definition or name. Business term versions are valid starting at some point in time, possibly becoming invalid at another. At any given time, at most one version (and with it the associated structures like reference or custom trees) is valid. Names are not identifying properties of business terms (homonyms do occur), hence we use a unique identification number – the SDL-ID – to unambiguously identify them.

The SDL Tool does not support branching. Instead, temporal selection is the mechanism for choosing repository object versions belonging to a specific configuration. As there is

at most one such valid version, this is a straightforward task. The worst that can happen is that no valid version exists. This raises the issue of handling CIM elements that are no longer considered valid. To deal with this, we make a distinction between the lifecycle of business terms and the structures they are part of. Any change to the structure of a reference tree leads to a new version; the reference tree itself may only be valid or invalid. If it is invalid, it is no longer considered part of the CIM. On the other hand, a value may posess a third state. If it is invalid, but part of a valid reference tree, it is considered deprecated (see Tab. 1).

Value	**Reference Tree**	**Status**
Valid	Valid	Valid
Invalid	Valid	Deprecated
Valid	Invalid	Invalid
Invalid	Invalid	Invalid

Table 1: The state of a business term is determined by its own validity, the validity of the tree it is or has been part of, and whether it still is currently part of it. If it is not part of the currently valid reference tree, the value is generally invalid (with respect to that tree).

The distinction between business term version history and the term's current status gives way to an (in our opinion) elegant architectural decision we took with respect to how applications integrate and react to evolution of the CIM. Generally, there are three main players in our integration architecture:

1. a number of applications (OLTP, ODS, DWH, or OLAP), designed for their own PIM,
2. one model repository, containing the CIM and all PIMs, and
3. one mapping repository, with PIM-to-PIM mappings across a PIM's history and between different PIMs.

Applications use the model repository to store the PIM underlying its own design. The model repository is used to design the CIM, version n, and forward-engineer the PIMs, version n for context version m, from it. If a CIM version $n+1$ occurs, the PIMs version $n+1$ for context version m can be generated based on available metadata. If a local change occurs, a single PIM version n for context version $m+1$ can be created, leaving all other PIMs unaffected. This way it is possible to keep applications in sync with the CIM.

Due to the structural integrity constraints, a PIM-to-CIM mapping is much more simple than mapping between different (often consecutive) versions of a PIM, or even worse, between different PIMs that are structurally only loosely correlated. As for the mapping of different versions of a PIM, ambiguities can occur, for example values can exhibit more than one parent value in the filtered trees they belong to. (It can also happen in reference trees.) Here, manual intervention is required. The metadata provided for

mapping a PIM between different versions is stored in the mapping repository and used by applications. The model repository is uninterested in it. The same holds for mapping between PIMs, where ambiguities can occur as well, for example values existing in one PIM's filtered tree but not the other's. (Note that both PIMs are understood to refer to the same CIM version, for which reason no history-related ambiguities occur there.)

As far as the lifecycle of attribute values is concerned, mappings are provided for values that are truly invalid and have a known successor. Applications are required to map them, especially at inbound interfaces. If no mapping is available, the value must be rejected. As for deprecated values, the question when to make a deprecated value truly invalid is a business decision. Hence, mappings may be provided or not, and applications may use them or not. More specifically, some applications (notably OLTP systems with strong auditability requirements) may decide to never change the original values associated with some stored fact (see Fig. 1). However, the assumption is that for most systems, the deprecation period is not indefinite, especially in management reporting.

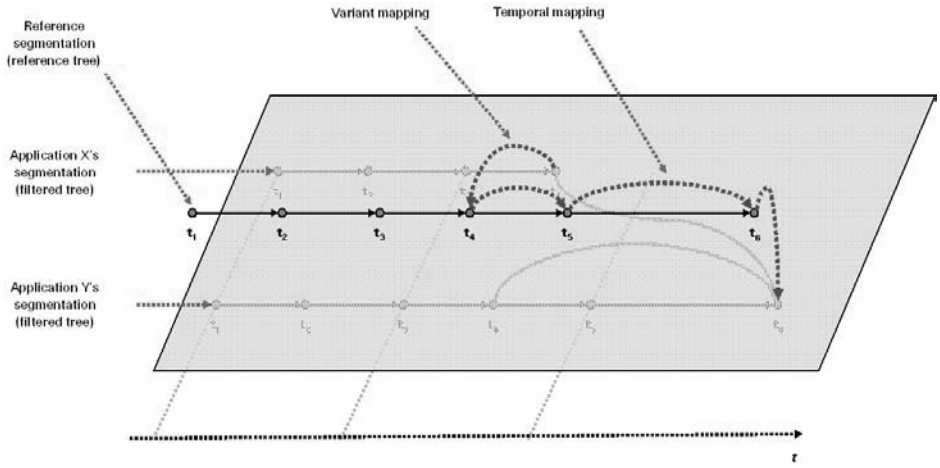

Figure 3: Variant (PIM) integration via the reference (CIM) and across time as seen at Swiss Re. The filtered trees can be easily mapped to the reference tree and back using context-specific metadata. Temporal mappings are based on successor information. If two applications X and Y want to integrate, they first map from the source application's PIM to the CIM, then across time and back to the target application's PIM.

The welcome side-effect of such an architecture is that it allows for the automatic handling of PIM-to-CIM mappings (and back to a PIM again). Given that we are driving towards this goal, this is important to us. At the same time, there is a wide range of tools at the discretion of applications to integrate with other applications one the one side, and to extend or shorten their own adoption of the latest CIM version on the other side. If, however, they decide to always stick with the current CIM version, the complexity they are exposed to is tangibly reduced.

Along with the different version selection mechanisms of the SDL Tool (temporal or version-based), we have begun to make application interfaces a first-class abstractions in the PIMs and use mechanisms similar to profiles in the Unified Modeling Language to further automate mapping generation. Applications will specify their change adoption

model (e.g., real-time, delayed, none), involved PIMs, and the mapping engine will be capable of (semi-automatically) providing PIM-to-PIM mappings, even across different referred-to CIM versions.

For example, the Non-Life Reinsurance Data Warehouse validates incoming data based on the SDL. It uses the currently valid CIM, i.e. its PIM is derived using temporal selection. Downstream applications use the same PIM (which sort of establishes a non-life-specific CIM of itself). The PIMs of upstream applications are partially based on specific PIM versions. However, mappings over time can, as long as both PIMs are based on the CIM, be derived from successor information.

The lack of branching in the versioning mechanism of the SDL Tool has turned out not to be a problem. Time is a much more important concept to business people, which is why we found the temporal perspective on versions to be the dominant mechanism used. Late-breaking changes or corrections are an exception, leading to the effect that consecutive repository object versions are usually also ordered in terms of their validity. Amendments are thus usually made to the most recent version, virtually eliminating cases of changes to other, much earlier versions.

4 Summary

We have presented our approach to the specification, integration, and evolution of application (interfaces) with respect to more than just one model. Our mechanism foots on the idea of one "to be" model, the CIM being used as the forward-engineering basis for many "as is" models, the PIMs. Both of these are based on business terminology, making proper specification more straightforward. Integration and evolution is simplified by means of a threefold division of responsibilities between applications, model repository, and mapping repository. The lifecycle status of business terms is used as a basis to decide on whether and how application interfaces must react to changes in CIM or PIMs.

Our experiences have generally been positive, though it has become obvious to us that legacy applications play a major role in a corporation as large as ours. This again was used as an argument for the existence of many PIMs concurrent to the CIM. Nevertheless, governing CIM design does take place, which means that a substantial degree of standardization is feasible.

Returning to the three goals defined at the outset of this article (establish and manage a consistent model of the business, stepwise refinement of the model, support for mapping between models, decoupling of applications such that interfaces prove stable under business model evolution), we can say that we have made a big step. The complexity intrinsic to mappings has not yet been fully mastered, but we are on a good way to achieving that goal as well in the long run.

Acknowledgements

The two introductory paragraphs on reinsurance business are an (almost) verbatim copy from [Mar03]; used with permission.

References

[Car00] R.L. Carter. *Reinsurance: the Definitive Industry Textbook.* Euromoney Institutional Investor PLC, 2000.

[Mar03] Robert Marti. Information Integration in a Global Enterprise – Some Experiences from a Financial Services Company. In Gerhard Weikum, Harald Schöning, and Erhard Rahm, editors, *BTW 2003, Datenbanksysteme für Business, Technologie und Web,* volume 26 of Lecture Notes in Informatics (LNI), pages 558–567. Springer, February 2003.

[MM03] Joaquin Miller and Jishnu Mukerji. *MDA Guide.* Object Management Group, June 2003.

[WA03] Hans Wegener and Gunnar Auth. Spezifikation von Fachkomponenten mit der Swiss Re Data Language. In Klaus Turowski, editor, *4. Workshop Modellierung und Spezifikation von Fachkomponenten,* Bamberg, October 2003.

[Weg01] Hans Wegener. Generative Programming and Incompleteness. In Krzysztof Czarnecki, editor, *OOPSLA Workshop on Generative Programming,* October 2001.