

The “IIDEAS” architecture and integration methodology for integrating enterprises

Matthew West – Shell Services International, Julian Fowler – PDT Solutions

1. Introduction

This paper presents an architecture and a methodology for integrating industrial data for exchange, access, and sharing (IIDEAS). Together they support:

- integrating data which may be:
 - from different sources or different contexts,
 - described by different models, or
 - described in different modelling languages;
- sharing data among applications through system integration architectures;
- resolving conflict between models developed with different objectives;
- translating models between different modelling languages.

This work is being developed into an ISO Technical Specification as ISO18876.

2. Fundamental concepts and assumptions

The architecture and integration methodology presented here works on the following fundamental concepts and assumptions.

2.1 Integration models

2.1.1 Principles

The three-schema architecture for data models [1] shows that, for any data model, it is possible to construct views on the original model. Here this principle is extended to cover other types of model and modelling languages. In the integration of models, this process is reversed: a model is created for which the initial models are views. A model created in this way is an integration model with respect to the initial models in that it is capable of representing information with the scope of either or both of the original models. This is illustrated in Figure 1 below.

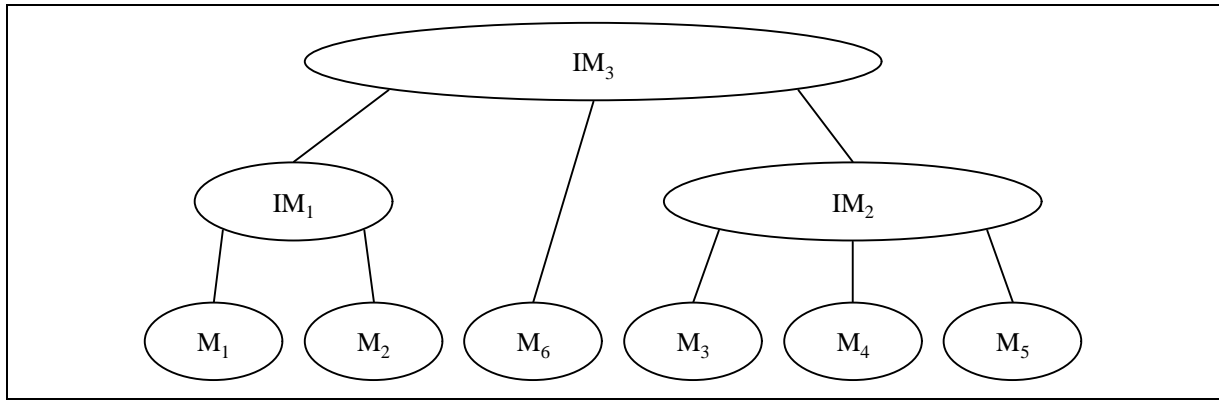


Figure 1 – Model Integration

An integration model can be created if a common understanding of the application models to be integrated can be established [2]. Difficulties in creating such a model point to a gap in human knowledge about the subject of the application models. Further, there may be more than one integration model to which an application model can be integrated, where the integration models support different ways of looking at the world. See Figure 2 below.

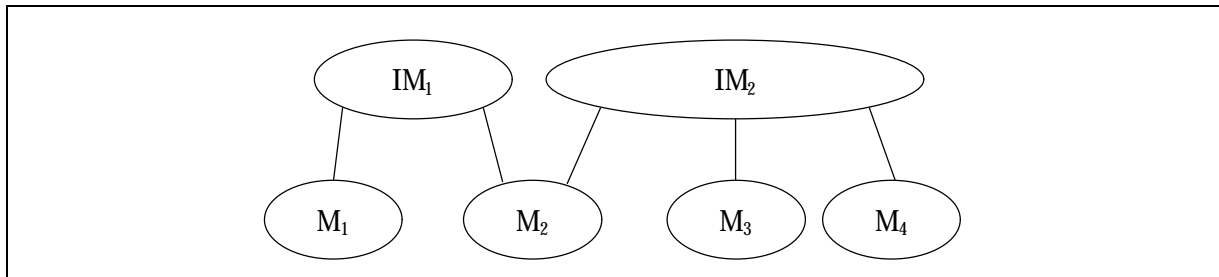


Figure 2 – Integration into more than one integration model

Models that have been created as integration models can themselves be integrated. This means that any arbitrary set of models can, in principle, be integrated at the cost of creating a new model.

One possible use for the architecture defined here is the development of an integration model that is stable in the face of the integration of additional models. Here stable means that the existing integration model does not need to be changed as more models are integrated, though extensions of the integration model may be necessary.

Integration models sometimes represent concepts that are more generic than the models they integrate. This is necessarily the case when the models being integrated have conflicting constraints affecting the information that is to be represented. These constraints should be preserved by the integration process, and held in some form other than in the structure of the integration model. The constraints can be held in the mapping specification or as data within the integration model.

2.1.2 Scope and context

Integration models can be created that consider two or more application models of interest – the scope and context of such an integration model can be no smaller than the combined scope and context of the application models being integrated. The relationship between such an integration model and the other application models that it integrates is shown in Figure 3 below.

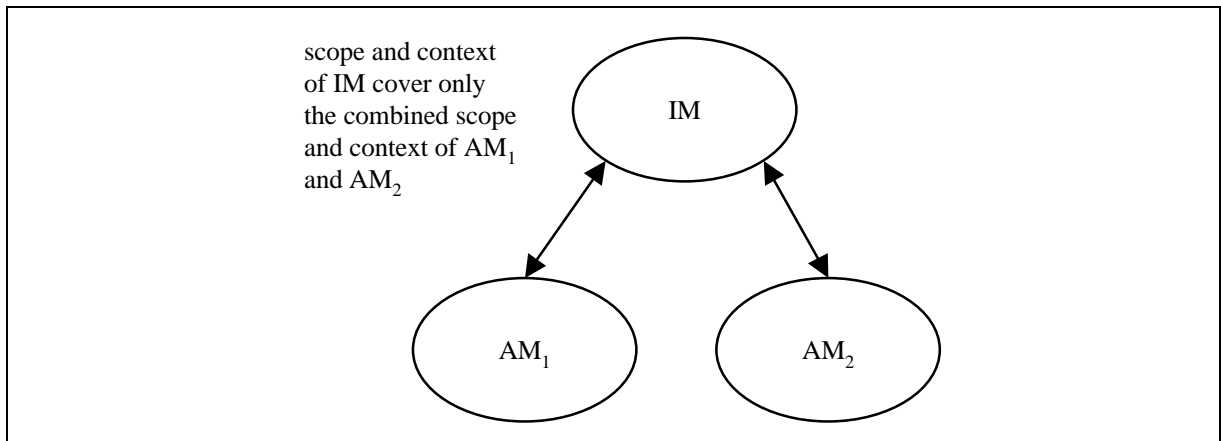


Figure 3 – A limited integration model

If requirements subsequently emerge to integrate additional application models with AM₁ and AM₂ (and hence with IM) it is unlikely that the context of these further models will fit within that of IM. This implies that IM cannot support the information represented by the further application models, and will itself have to be integrated with another integration model (created for this purpose or selected from candidate integration models). This is shown in Figure 4 below.

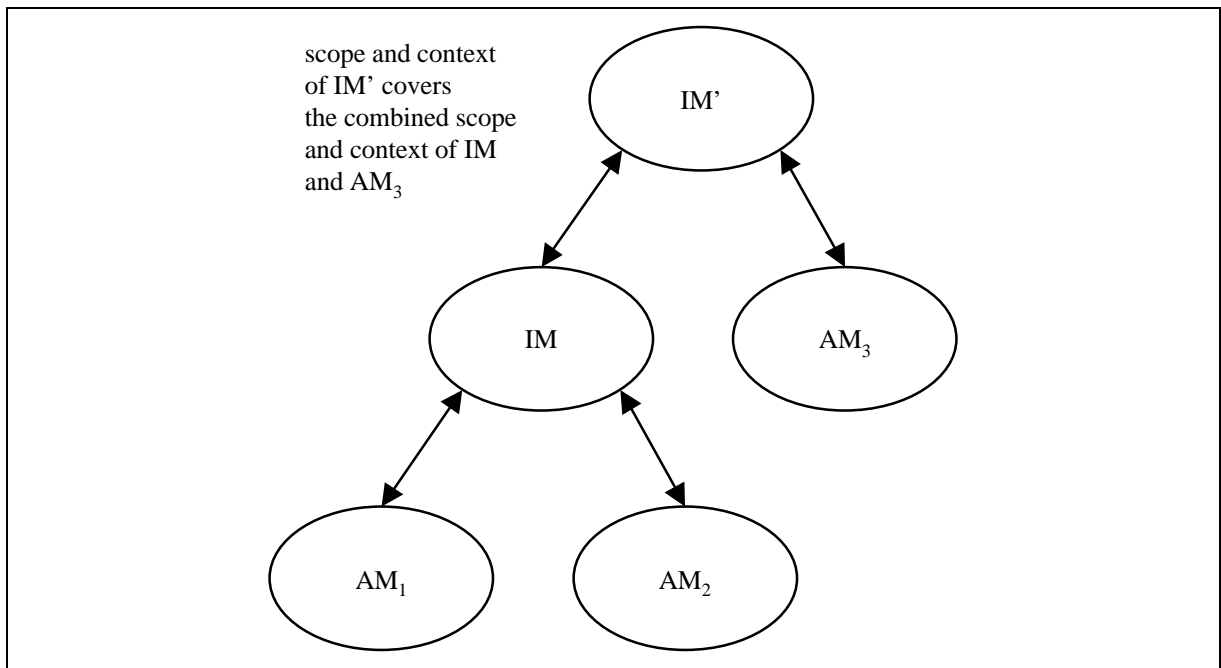


Figure 4 – Integrating an application model and a limited integration model

However, the initial integration model (IM) can be chosen to have a wide model context. This means that it can support the information needs of many different applications, even though its initial model scope is limited to that of the models that it integrates, as shown in Figure 5 below.

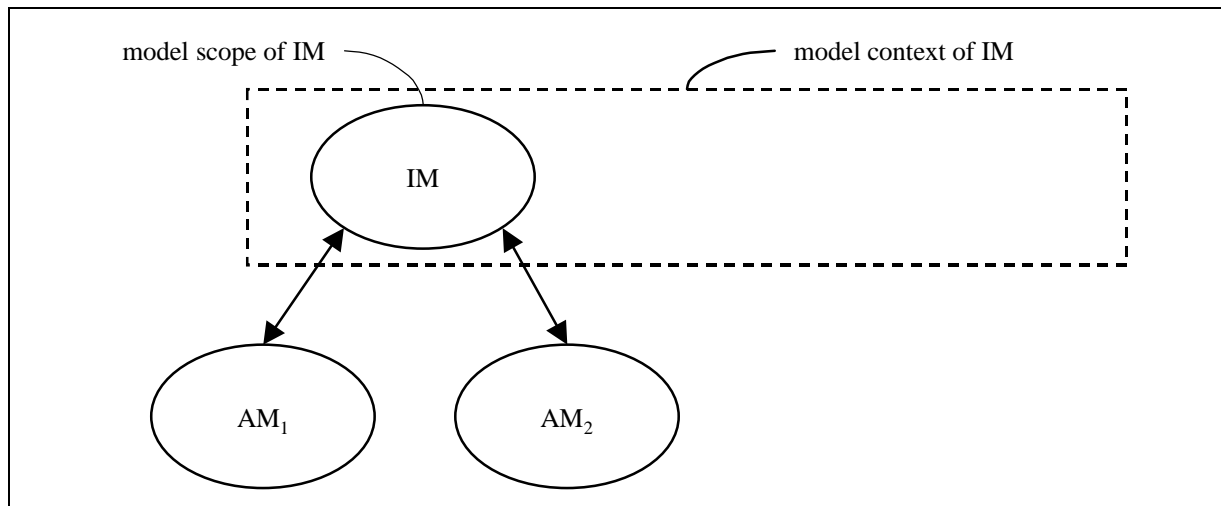


Figure 5 – Using an integration model with a wide model context

Integration of further application models can then be achieved through extension of the integration model – enlarging the model scope within the wide model context, as shown in Figure 6 below. In this case, the integration model can be extended (though the addition of constructs that represent specific and derived concepts) without changing its initial content or the mappings of the initial application models AM_1 and AM_2 .

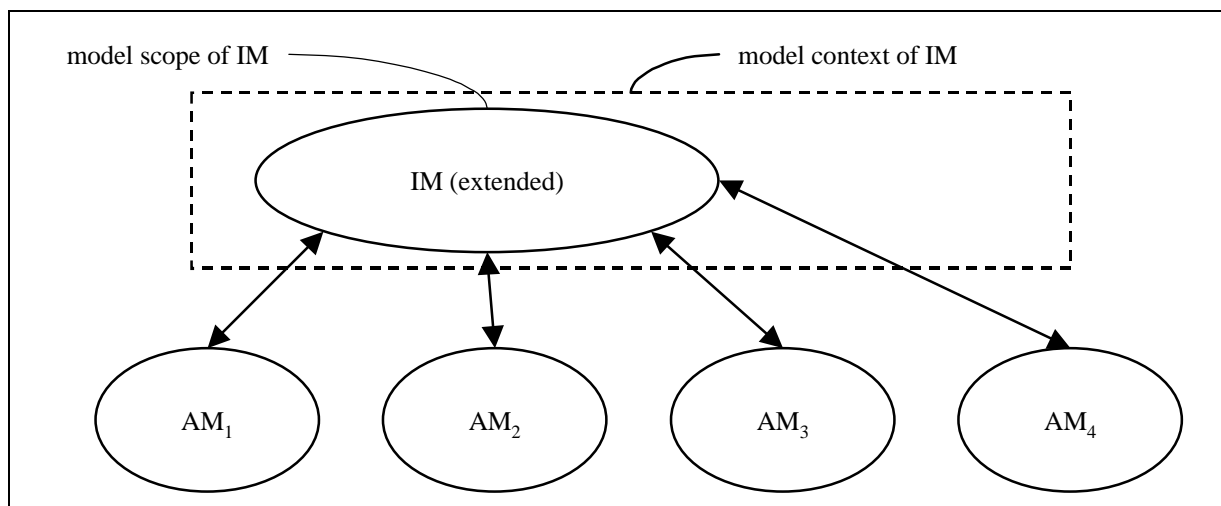


Figure 6 – Integrating additional application models

Integration models that have a wide model context are suitable for integrating many different application models and have the following characteristics.

- Each object is modelled in terms of its underlying nature, based on a specific paradigm or view of the world being modelled. This means that each object is modelled neither in terms of the role or roles that it plays in particular circumstances nor the information about the object that is of interest in particular cases.
- The names chosen for the model constructs also reflect the underlying nature of the subject of the model.
- Only those constraints that are applicable for the whole content and scope of the model are represented in the structure of the model.

- Classes are represented as nodes in a subclass/superclass (specialization/generalization) hierarchy which is rooted in a single class.
- The model represents and manages history and change.
- Artificial identifiers are used as a surrogate for the identity of the things represented by the model. These identifiers allow assertions of equivalence to be made across multiple files, databases, and other information systems. Other attributes and relationships are not used as parts of these identifiers.
- The modelling language chosen to represent the integration model is used consistently.

The following assumptions apply to the use of integration models in the integration process.

- The integration process does not change an integration model. Change here means alteration to the existing structure or content of the integration model, and is distinguished from the process of extension which adds new constructs to an integration model without altering those that form the integration model at the start of the integration process. If it is necessary to change rather than extend the integration model, a new integration model is created.

2.1.3 Integration model concepts

The concepts represented by an integration model can be classified as primitive concepts and as derived concepts. Primitive concepts are the building blocks for the definition of other concepts, and can be further classified as:

- foundation concepts,
- general concepts, or
- specific concepts

as represented in Figure 7 below.

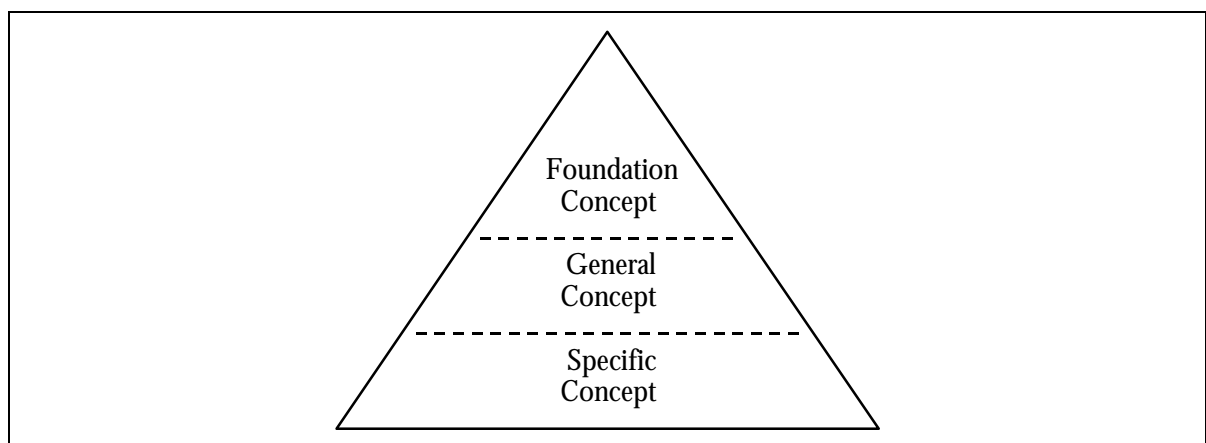


Figure 7 – Primitive Concepts

Specific concepts are dependent on general concepts that are dependent on foundation concepts, since all the lower concepts relies on the existence of one or more higher level concepts. For example, without the foundation concept of classification, there is relatively little that can be said about anything.

For example at the top level, an integration model might have foundation concepts like classification, connection and composition. General concepts might include those of physics, and finally discipline specific concepts that are limited in their range of application.

2.1.4 A full integration model

A full integration model, as illustrated in Figure 8, is more than just primitive concepts; it includes derived concepts – useful and valid combinations of primitive concepts. Only derived concepts that are of interest need be recorded.

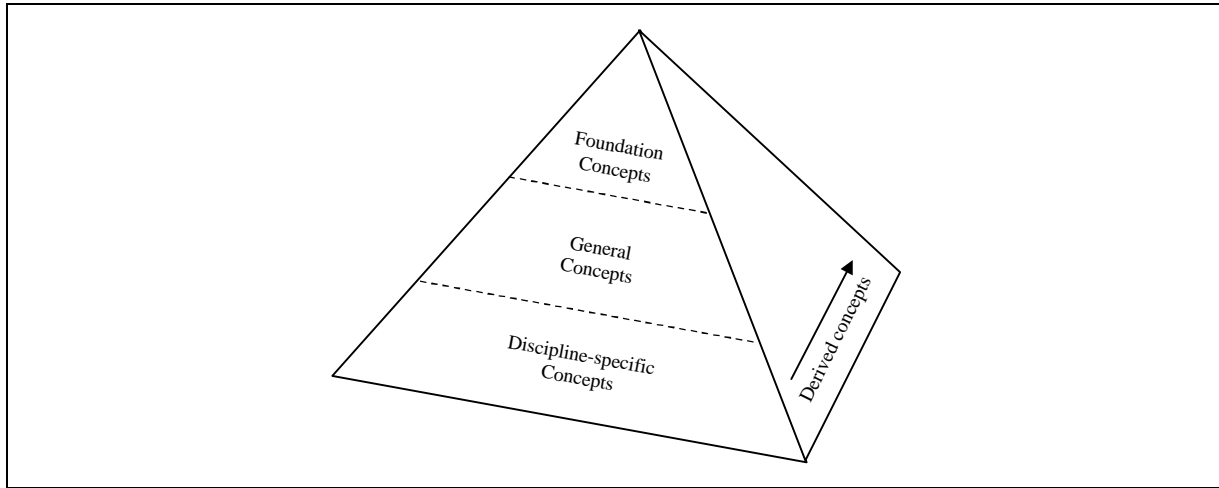


Figure 8 – A full integration model

This architecture does not require that primitive concepts are primitive forever. If a concept that is initially thought to be a primitive concept turns out not to be, then the concepts it is derived from can be identified/added, and the derivation added, so that it becomes a derived concept away from the front face of the pyramid. This allows flexibility to reflect an improved knowledge of the world, rather than reflecting knowledge of the world that is constrained by a modeller's knowledge at a point in time. Therefore, an integration model will need to be maintained and extended, and a mechanism for maintenance and extension will be necessary.

2.2 Mapping specifications

Mapping specifications specify the transformations that determine how the instances of one model can be represented as instances of another model. Mapping specifications are used in two ways, as follows.

The mapping specification can describe the mapping transformations between

- a subset of an integration model and
- a pre-existing application model which governs data that is separate from that governed by the integration model.

In this case the mapping specification describes the transformations that enable assertions of equivalence of instances of one model to be made with respect to instances of the other.

The mapping specification can describe the mapping transformations between

- a subset of an integration model and
- an application model that is used as an application view.

In this case the mapping specification describes how instances in the application view are created from instances in the integration model.

The following assumptions apply to the mapping specifications that are created during the integration process.

- New concepts or constraints are not introduced in the mapping specification, i.e. mapping specifications are limited to transformations of structure, terminology, and encoding.
- A complete mapping specification is bi-directional. However, the transformations of the first model to the second model may have to be specified separately from those of the second model to the first model, and the mapping in one direction need not be derivable from the mapping in the other direction.

3. Overview of the model integration process

The model integration process takes a number of application models and an integration model. It ensures that all the concepts of the application models are represented in the integration model, and develops a mapping specification between the integration model and each of the application models.

There are three possible cases for the integration process:

- the integration model and the application models both exist before the integration process starts;
- the application models to be integrated exist before the integration process starts, but not the integration model;
- the integration model exists before the integration process starts, but the application model needs to be developed from some statement of requirements.

The first of these covers all the elements of the other two, and is described here in outline for one application model.

Integrating an application model with an integration model is illustrated in Figure 9 below. The goal of this integration process is to allow the same information that is represented in the application models to be represented in the integration model without loss of meaning, and to allow transformations between these representations. The result of the integration process is a mapping specification between the application model and a part of the integration model. In order to define this mapping it may be necessary to extend the integration model so that it precisely represents the concepts found in the application model.

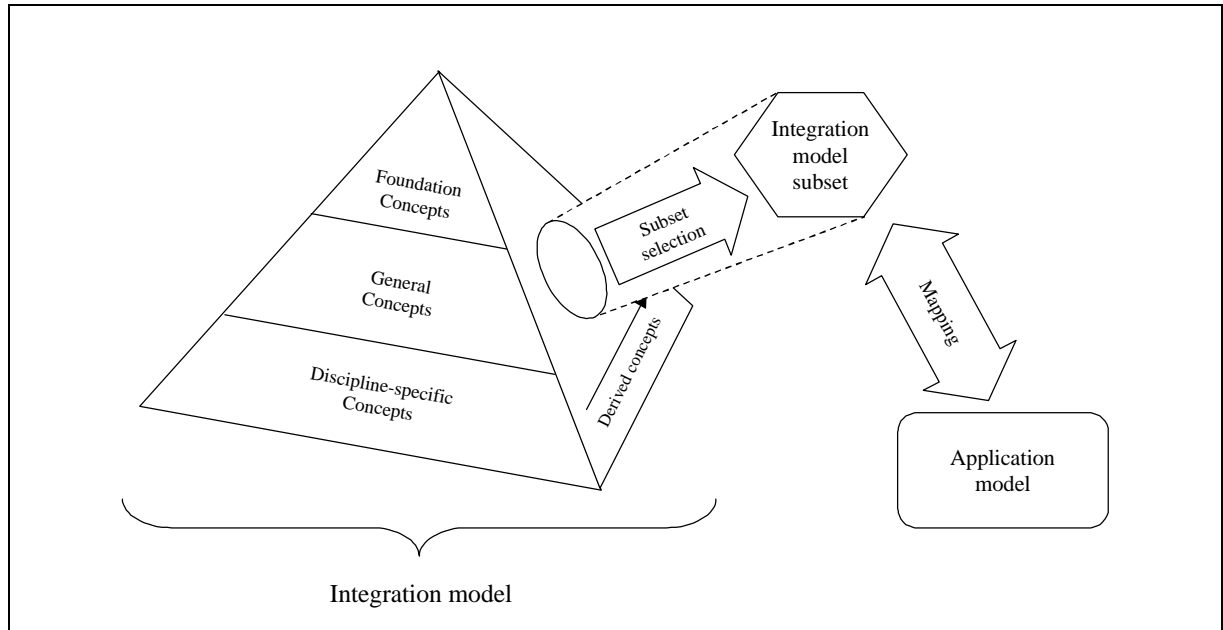


Figure 9 – Integrating application models with an integration model

Being an integration model derives from the role that it plays with respect to other application models. The fundamental characteristic of an integration model is that it integrates two or more application models.

The process of integrating an application model with an integration model is divided into a number of steps, as follows:

- analyse the application models and identify the equivalent concept of the integration model, including any constraints that apply, see Figure 10. Most application models have a context within which the model has to be understood, but which is not explicit in the model itself. Usually it will be inappropriate to add this information explicitly to the application model. In this case these requirements should be captured in the mapping specification as part of the integration process.

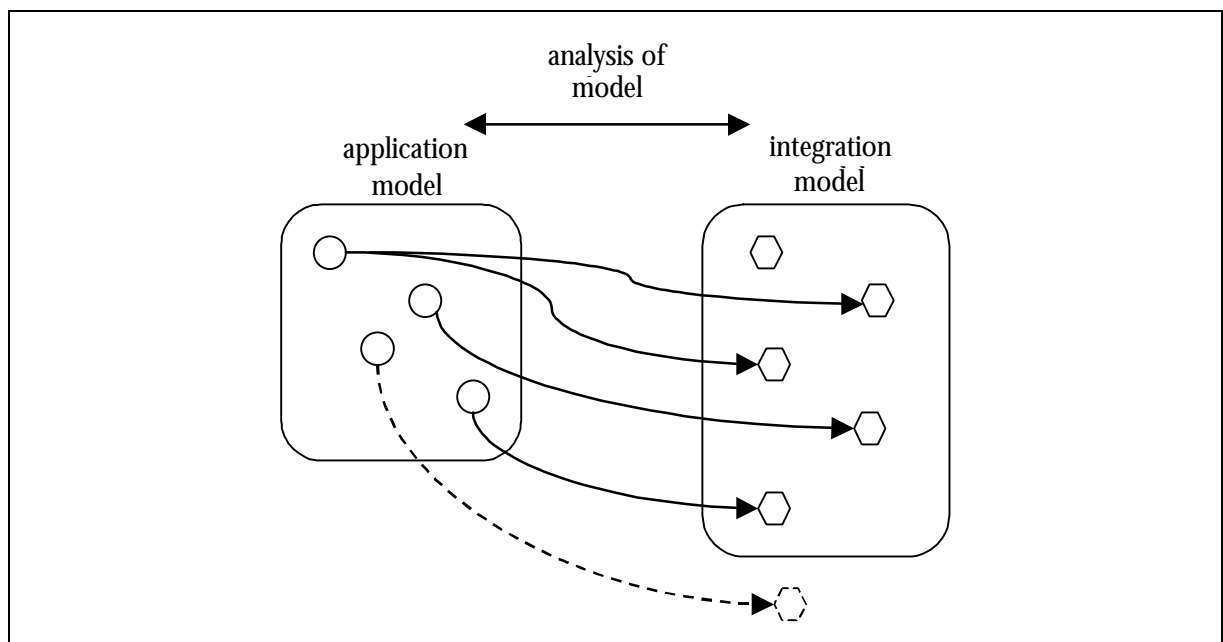


Figure 10 – Analyzing the application models

- if necessary, extend the integration model so that it includes all the concepts found in the application models, see Figure 11;

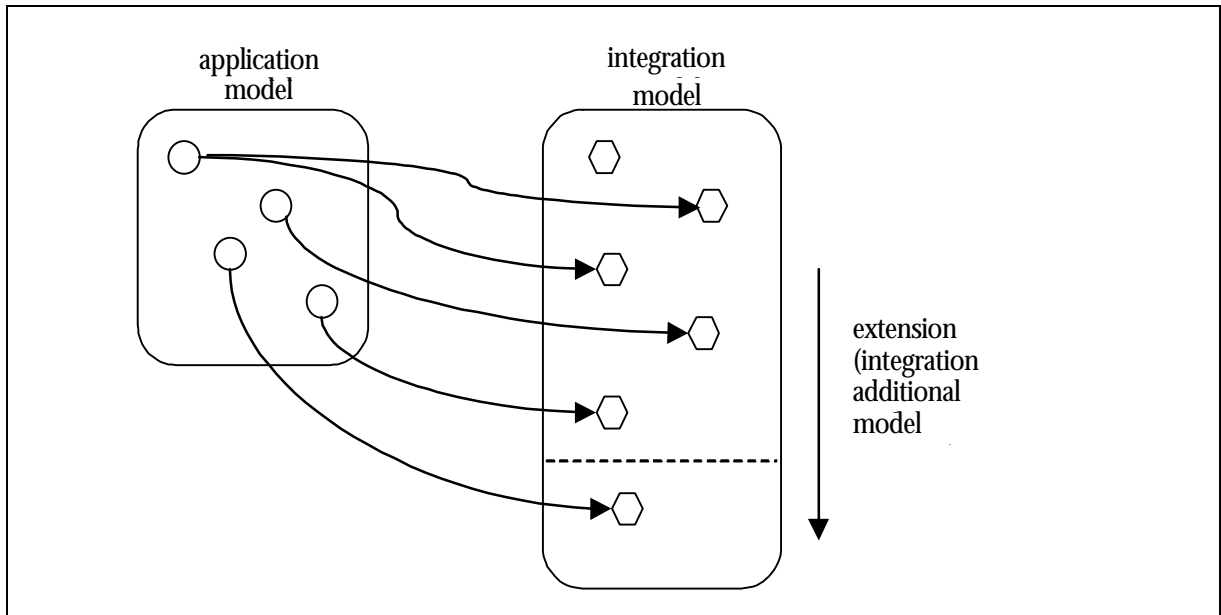


Figure 11 – Adding any missing concepts to the integration model

- identify the part of the integration model that represents the concepts in each application model, see Figure 12;

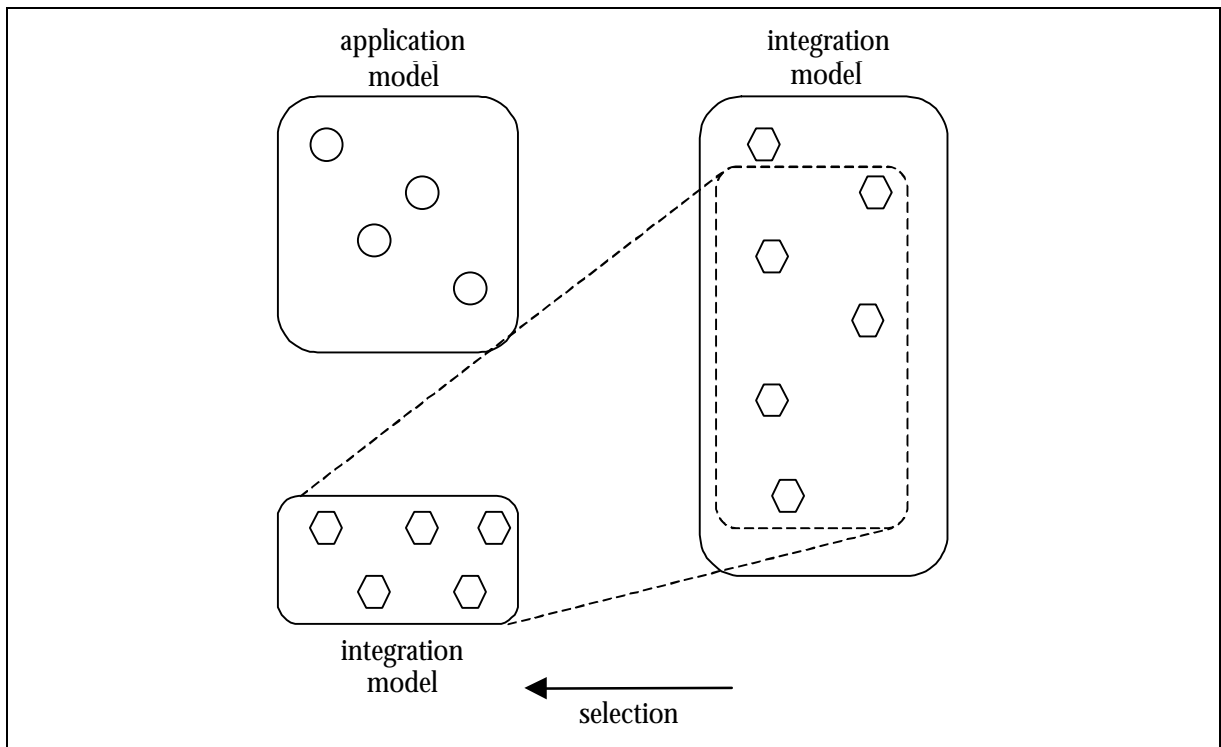


Figure 12 – Identifying the subset of the integration model

- create the mapping in each direction between each application model and the appropriate subset of the integration model, see Figure 13;

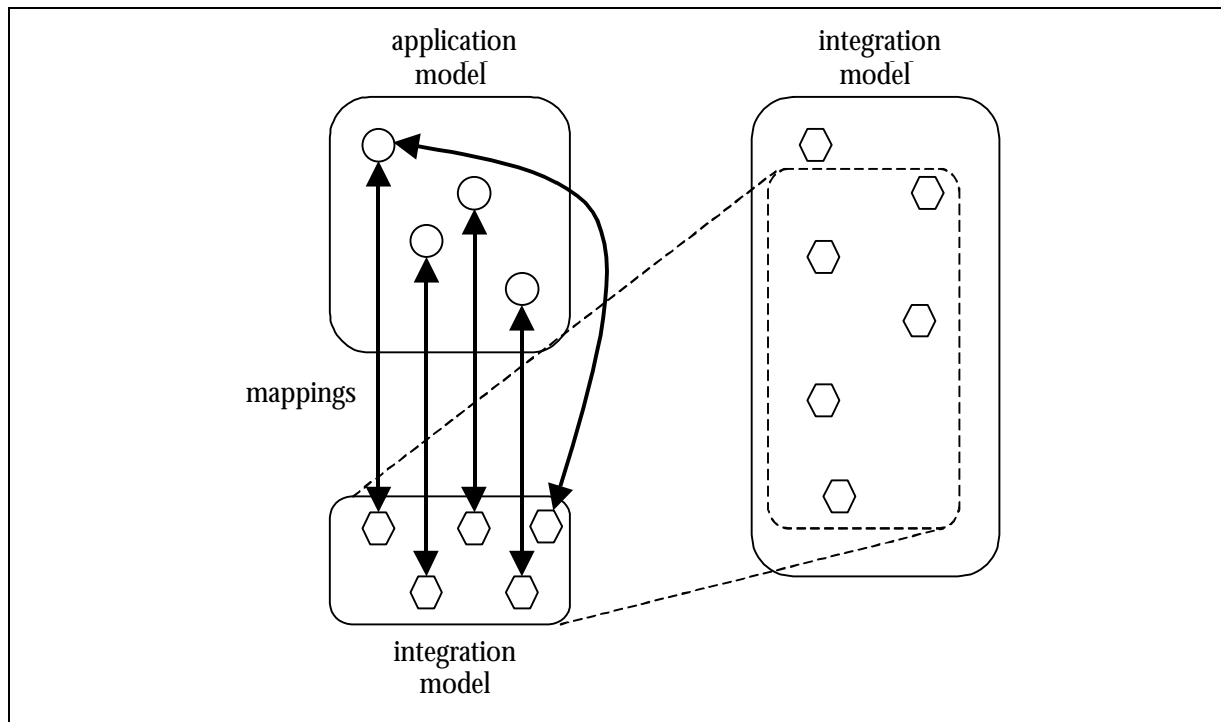


Figure 13 – Creating the mapping between the integration model subset and the application model

- specify any structural transformations, terminology transformations, or encoding transformations that apply within the mapping;
- specify any transformations that are necessary between model representations. For example. If an application model is specified in the XML Schema definition language and the integration model to which it is mapped is specified in EXPRESS (ISO 10303-11), a transformation between these languages will be necessary to map between different representations of the same concepts.
- repeat this process for all other application models to be integrated.

Most application models have a context within which the model has to be understood, but which is not explicitly represented in the application model itself. Mapping successfully in both directions requires that both the explicit model and its context be mapped into the integration model. For example, in a salary payment system may be an entity data type called **employee**. However, it is often implicit that each person represented by instances of this entity data type is an employee of the company that operates the system and legally eligible for employment under company and governmental policies.

4. Integration architecture components

4.1 Entity attribute relationship based components

Note that in what follows object oriented models are considered as entity attribute relationship models.

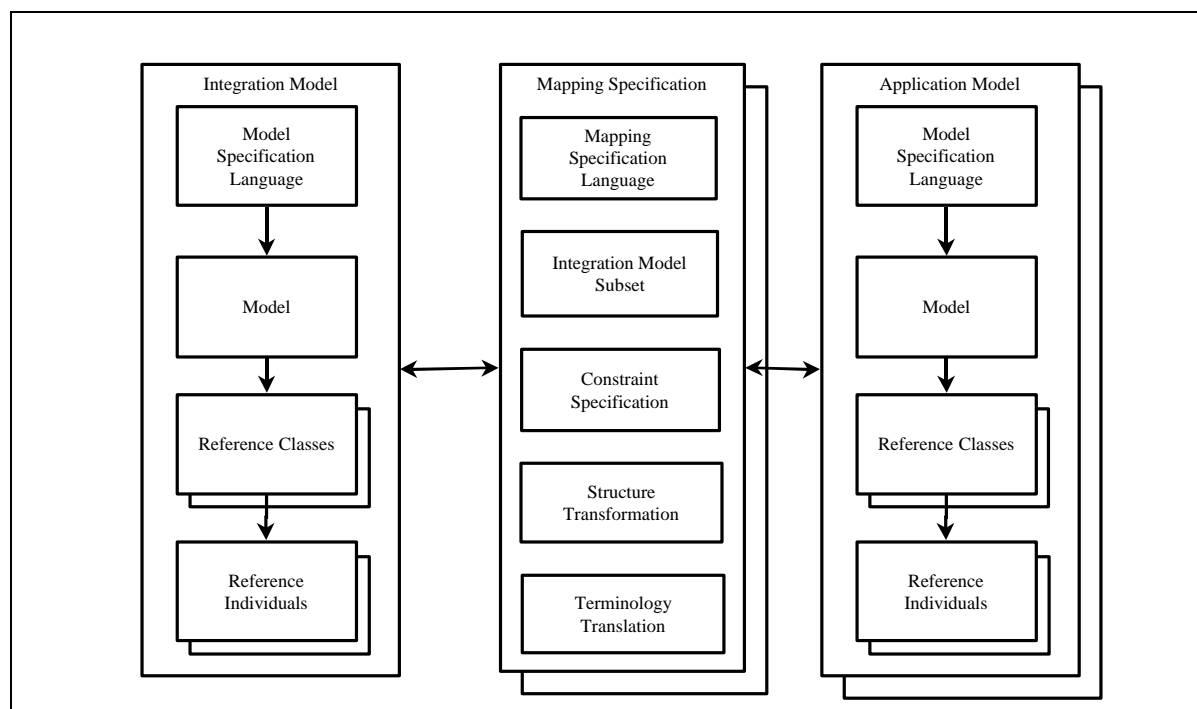


Figure 14 – Integration architecture components

Figure 14 gives an alternative view of Figure 9 and shows how the elements may be managed, as follows:

- integration models. This may be a single model with multiple levels of abstraction, using a suitable logic based language, such as KIF or EXIST, or a layered model, using an entity-relationship language such as EXPRESS with a data model and reference data libraries. Figure 14 illustrates the use of a model to define the structure of a reference data library that can hold reference classes and reference individuals. There may be a number of reference data libraries. These are to cover the discipline specific primitive concepts, and derived concepts. Procedures may be required for their development to ensure there is no duplication across the libraries.
- mapping specifications that specify a part of the integration model, constraints that specify the subset of the integration model that maps to the application model, structural transformations between the structure of the integration model and the structure of the application model, and finally terminology transformations between the integration model and an application model. Mapping specifications may also be required between model specification languages.
- application models.

5. Data mapping and consolidation

Mapping between models is not sufficient to achieve integration. This requires reconciliation of information represented according to the different models. This process is illustrated in Figure 15 below.

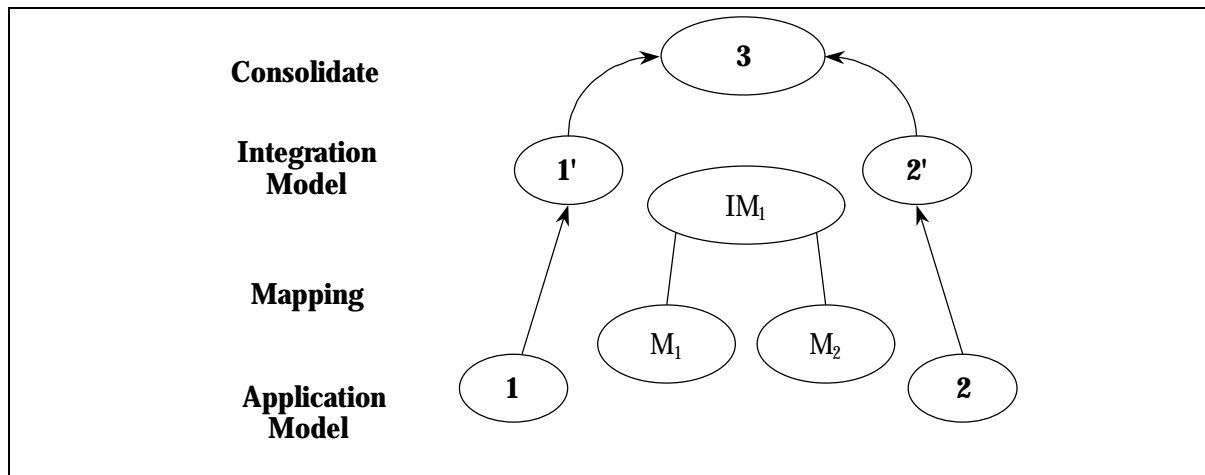


Figure 15 – Data consolidation

- Translate the data population 1, and 2, according to their source models into the data populations 1' and 2' according to the model IM_1 .
- Identify which data elements in the two data sets represent the same things, and consolidate them in data population 3.

It should be noted that this requires a common, reliable, persistent identification mechanism. This does not necessarily mean defining some new attribute for use by the various systems. It may be possible to use an existing attribute. There may be a collection of attributes and relationships that together can provide a unique identification. Any of these may be mapped to an independent identification scheme.

6. Exploitation

It is with some pleasure that we note that some commercial projects are starting to take an interest in IIDEAS. These include:

- Master Reference Data Project – Shell, and
- Multiview – a US DLIA funded project.

Multiview is aiming to integrate information across a range of application systems using a shared data environment, and for which ISO 18876 is seen as one of the contributing standards.

The Shell Master Reference Data Project is a pilot project that is part of Shell's e-Architecture initiative, designed to make a range of back office systems like SAP available for e-business.

7. Conclusions and next steps

This paper has presented the IIDEAS architecture and methodology, which is capable of supporting the wide spread integration of information. It is deliberately high level, and thus manages to be independent of issues like particular modelling languages, or particular integration models. On the other hand, it will also be necessary to develop more specific guidance on how to apply IIDEAS in particular circumstances such as particular modelling languages such as EXPRESS, and particular integration models such as ISO15926-2. Further, the architecture will give rise to methodologies for developing integration models, and languages that better meet the requirements of IIDEAS.

8. Bibliography

1. ISO/TR 9007:1988, *Information processing systems -- concepts and terminology for the conceptual schema and the information base*.
2. BARWISE, Jon; SELIGMAN, Jerry. *Information flow: the logic of distributed systems*. Cambridge: Cambridge University Press, 1997.
3. *EPISTLE Core Model V4* EPISTLE, 2001. Available from <http://www.stepcom.ncl.ac.uk>.
4. *Integration Definition for Information Modeling (IDEF1X)*. Federal Information Processing Standards Publication 184, December 1993
5. WEST, Matthew; FOWLER, Julian. *Developing High Quality Data Models*. Version 2.0, Issue 2.1. EPISTLE, 1996. Available from <http://www.stepcom.ncl.ac.uk>.
6. FOWLER, Julian, "Industry requirements for SC4 standards", 1998, ISO TC184/SC4/WG10 N173
7. GUARINO, Nicola, "Some Organizing Principles for a Unified Top Level Ontology", 1997, <http://www.ladseb.pd.cnr.it/infor/ontology/ontology.html>
8. GUARINO, Nicola, "Some Ontological Principles for Designing Upper Level Lexical Resources", 1998, <http://www.ladseb.pd.cnr.it/infor/ontology/ontology.html>
9. PARTRIDGE, Chris, "Business Objects, re-engineering for reuse", Butterworth Heinemann, 1996, ISBN 0-7506-2082-X