

Charteris White Paper

Four Levels of Data Integration

Draft 01

Robert Worden (robert.worden@charteris.com)

11 March 2005

CONTENTS

CONTENTS	2
1. INTRODUCTION	3
2. LEVEL 1	4
3. LEVEL 2	5
4. LEVEL 3	6
5. LEVEL 4	7

1. INTRODUCTION

Application Integration is the biggest cost driver of corporate IT. While it has been popular to emphasise the business process integration aspects of EAI, it remains true that data integration is a huge part of the problem, responsible for much of the cost of EAI. You cannot begin to do process integration without some data integration.

Data integration is an N-squared problem. If you have N different systems or sources of data to integrate, you may need to build as many as $N(N-1)$ different data exchange interfaces between them – near enough to N^2 . For large companies, where N may run into the hundreds, and N^2 may be more than 100,000, this looks an impossible problem.

In practice, the figures are not quite that huge. In our experience, a typical system may interface to between 5 and 30 other systems – so the total number of interfaces is between $5N$ and $30N$. Even this makes a prohibitive number of data interfaces to build and maintain. Many IT managers quietly admit that they just cannot maintain the necessary number of data interfaces, because the cost would be prohibitive. Then business users are forced to live with un-integrated, inconsistent data and fragmented processes, at great cost to the business.

The bad news is that N just got bigger. New commercial imperatives, the rise of e-commerce, XML and web services require companies of all sizes to integrate data and processes with their business partners' data and processes. If you make an unsolved problem bigger, it generally remains unsolved.

Users and software vendors have devoted huge efforts to tackling the N^2 data integration problem. The solutions available today can be grouped into four main levels of increasing sophistication and power:

1. Hand coding of data interfaces
2. Source-to-target mapping and translation tools
3. Integration hubs and brokers
4. Full model-based integration

This article discusses the costs and benefits you can expect at each level.

2. LEVEL 1

Most data integration problems are still tackled on a one-off basis, as part of individual development projects. One project typically builds or installs a new system, together with the interfaces it needs - to somewhere between 5 and 30 existing systems (the initial number is at the low end, and then grows through a system's lifetime). This means the interfaces are built at **Level 1**, by hand coding of the necessary data conversions.

The drawbacks of Level 1 are well known. Even to build one data interface, you need to understand in depth the data structures on both sides of the interface. Each side might be a relational database with hundreds or even thousands of tables, or an XML standard with thousands of element types. This 'double deep knowledge' of both systems is rarely found in one head, and is very expensive to acquire. Gaps in knowledge lead to design errors, and hand-coding adds further errors – which will not all be found in testing. Scale this up to the hundreds of interfaces found in a typical large company, and you have the 'data spaghetti' seen in most companies today - together with the cost overruns and the failed projects.

3. LEVEL 2

This headache gave rise to the present generation of **Level 2** data mapping tools. There are many of these tools on the market, such as BizTalk Mapper from Microsoft, or Transformation Server from Data Mirror. Typically, these tools use a graphical display of 'source' and 'target' data structures on two halves of the screen – showing two trees of nodes for the relational tables, columns, or XML elements in the source and target. You drag-and-drop to create **mappings** from source nodes to target nodes, inserting data conversions on the way. The tool then generates translation software, or translates automatically, from the mappings.

Level 2 is an advance over the hand coding of level 1, but it does not solve all problems. It does not solve the N^2 problem; if you have N different systems to interface, you may have to build as many as N^2 (or in practice, $5N$ to $30N$) sets of mappings. Nor does it solve the double deep knowledge problem. To build or maintain even one set of mappings, you require deep knowledge of the systems on both sides, and this double knowledge is rarely available.

Another limitation of Level 2 solutions has emerged with the advent of XML. Node-to-node mapping works well for relational databases, where information is explicit in the values of fields in records. However, XML allows deep nesting structures, with information carried by the nesting. Translating to and from XML may require complex structural transformations, preserving the information in the structure. Node-to-node mapping does not give good control of structural transformation; to get it right, you will sometimes be forced back to complex hand coding – losing the original cost benefit of the mapping approach.

So while the level 2 tools can give cost savings over level 1, they leave some problems unsolved; they are not the cost breakthrough that will solve data integration.

4. LEVEL 3

A few vendors offer the **Level 3** solution of Integration Hubs, or Integration Brokers. Typical of these are IBM's CrossWorlds, and Oracle's 9i Applications Interconnect. These hubs address more than data integration; they have important other functions, such as process integration and orchestration. Here we focus on their data integration role.

With an integration hub, data are not translated directly from system A to system B; they are translated via a central data representation on the hub. In principle, this should reduce the number of required translations from N^2 to $2N$ (from each system into and out of the central representation), solving the N^2 problem. However, in practice things are not so simple.

In hub-based products, you map source and target data models onto a common central data model. However, in well-known hub products, the central data model does not have the full expressive power of a UML class model, to represent many classes of object, their properties, and all the associations between objects. Typically, the central data model supports only **hierarchical fragments** of a full class model. For instance, one such fragment might have a top 'customer' object and hierarchical links to subordinate 'customer contact' objects.

If the data you wish to translate is entirely within one of these fragments, then translation in to a central fragment, followed by translation out of the fragment, does the job – solving the N^2 problem. However, data are not always hierarchical (this is why relational databases replaced hierarchical databases, twenty years ago). When the data span two or more hierarchical fragments, things get more complicated. To translate the links across fragments, you will need to hand-code. Hub-based tools typically have facilities to develop hand code and integrate it with tool-generated code. However, if you use a lot of hand coding in the hub, it is not clear how much cost you will ultimately save.

It is widely acknowledged that installing an integration hub is an expensive move. Before you do so, you will need to analyse your application needs carefully - to be sure that your problem is a good fit to what the tool can do, so that the cost reductions in data integration really justify the investment.

5. LEVEL 4

The **Level 4** solution to data integration is to map all data sources onto a central data model which has the full expressive power of a UML object model, then to do all data translations automatically from these mappings. If this could be done, the costs of integrating N data sources would be just the cost of developing N sets of mappings – solving the N^2 cost problem. Are there tools to do it, and is it practical?

There is an emerging category of tools under the name of *semantic integration* which claim to do just this. Companies in this space include Modulant Inc and Unicorn Inc in the US, and Charteris plc and Network Inference in the UK. Some of their products use the language of ‘ontologies’ rather than UML class models, associated with the W3C semantic web initiative; but there is strong overlap between UML and ontology concepts. The products differ in their detailed capability; but they all go beyond the ‘hierarchical fragment’ capability of level 3; they can handle non-hierarchical associations and inheritance; and they have the potential to make real inroads on the N^2 data integration cost problem.

Even if these tools work well, you may have another concern about the Level 4 approach. Is it possible to build a UML object model (or ontology) which is rich enough to encompass all data sources, without getting mired in complexity? Surely that would be a corporate data model – we tried corporate data models ten years ago, and they failed.

There are many reasons why many corporate data models failed. Typically they were developed before the widespread acceptance of object-orientation – so they did not use inheritance to reduce complexity. A corporate data model was typically based on the data models of a few core systems – so it built in physical design decisions and complexity from those systems. Finally, the tools for mapping databases onto a logical model were not well advanced. There are now techniques to build large UML business object models, and to map physical data sources onto them, giving models of feasible complexity.

In any case, you do not need a full corporate business object model to use the level 4 approach. You can build an object model spanning just the CRM domain, or just the finance and reporting domain; you may have plenty of systems to integrate just within those domains. You can start small and to expand the solution as its feasibility is proven. To map a system onto the business object model requires detailed knowledge of just that one system, and the business model - so the Level 4 solutions do not require deep knowledge of two systems at once, to make good mappings and translations.

Data integration is a huge problem, and there is a bewildering variety of tools to tackle it. Many of these tools also tackle other important aspects of application integration – such as security, recovery, transaction management and process orchestration – and offer pre-packaged adapters to common data formats. So the choice of a data integration tool is never going to be a simple one. However, understanding the four levels of data integration capability - understanding what they can do for you and what they cannot do - is an important step on the way to making that choice.