AN INTEGRATION METHODOLOGY BASED ON THE ENTERPRISE ARCHITECTURE

Marta Guerra mncg@netcabo.pt

Miguel Pardal mflpar@yahoo.co.uk

Miguel Mira da Silva mms@dei.ist.utl.pt

Instituto Superior Técnico
Departamento de Engenharia Informática
Av. Rovisco Pais
1049-001 Lisboa
Portugal

Abstract
Integration needs are difficult to predict and solve appropriately, and this leads to multiple partial solutions that lack a broader vision of the problem. The Enterprise Architecture defines a global view of the business data and processes, and of the applications that manage and support them.
In this paper, we propose an integration methodology based on the Enterprise Architecture's business semantics that reuses and extends data analysis techniques to specify essential integration details. The methodology proved useful in practical test cases, achieving immediate integration goals that are sustainable throughout the Enterprise's information systems life cycle.

Keywords: Integration methodology, Enterprise Architecture, CRUD Matrix, Entity-relationship diagram, Data-flow diagram

1. INTRODUCTION

Today's Organizations use *information systems* to support their businesses. These systems were created spontaneously to respond to specific needs within each organizational unit and usually lack a global Organizational plan. This evolution produced a situation where multiple existing applications manage replicas of data. This leads to high costs due to the use of incoherent information and the efforts to maintain coherence [Inmon93].

Present *integration technology* allows several systems conceived as "islands" to become mutually accessible and share data. Integration methodologies currently specify rules at the application level and do not address the information's integrity problems. For instance, if a replication of Client data occurs while planning to integrate three applications, which one should be ultimately responsible for the Client's data? Current integration methodologies are unable to answer this question, because they ignore crucial items in the information systems development – data entities and business processes [Laudon02].

The *Enterprise Architecture's* (EA) mission is to identify a set of applications, which are aligned with the data entities and business processes of the Organization. The EA specifies the mission of each application and its responsibility and rights over the information entities and business processes [Spewak93].

The *Create-Read-Update-Delete (CRUD) Matrix* is the graphical representation of the EA, containing the relationship between data entities, business processes and the applications. Figure 1 shows an example CRUD Matrix.

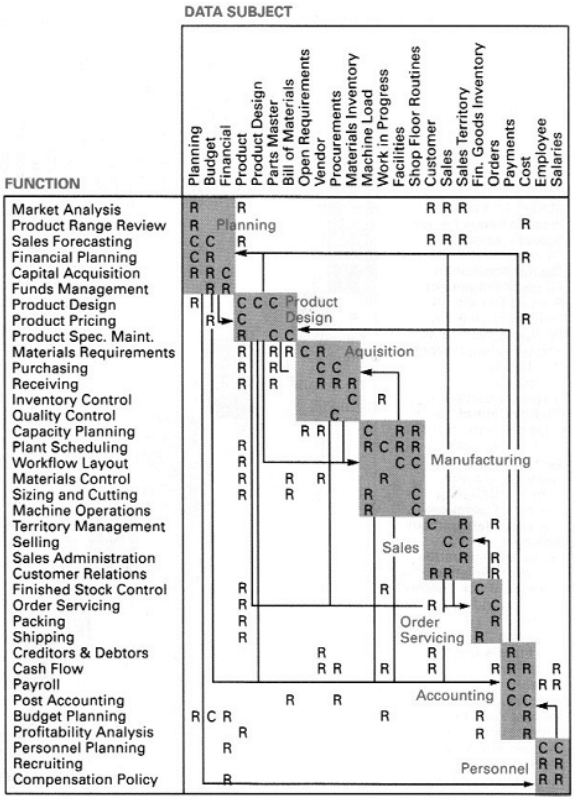

Figure 1 – Example of a CRUD Matrix from [Spewak93]. The application areas result from crossing the data entities with the business processes.

In this paper we propose an integration *methodology* based on the EA that aims to provide integration solutions through a specification model which guarantees the coherence of data entities and business processes. The specification model is based on standard techniques: Entity-Relationship diagrams and Data-flow diagrams [BatiniCeriNavathe92].

The methodology was applied to a practical test case achieving the desired results. This case is referred but not fully described in this paper for briefness reasons.

2. PROBLEM

The integration within an Organization begins with the identification of new functionalities that require data currently spread across several applications.

Typically, each application uses a unique representation of data. To share that data, it is not enough to make the means of communication available. It is necessary to make data transformations among the different representations available.

The following cases describe typical barriers that should be solved by integration:

a) A Windows application cannot communicate with a Mainframe application because the former uses proprietary network protocols;
b) An application represents the entity Product in a hierarchical XML structure and the other application uses several tables on a relational database;
c) An application represents the Date-of-Birth attribute of the Client entity in ISO format (year, month, day) and the other represents it in a national format (day, month, year);
d) An application of the Accounting department has a Client entity (people who have had transactions with the enterprise) and an application of the Marketing department has also a Client entity (prospective customers). Despite having the same name, they are different because they have different meanings;
e) An application has the Product entity with the attribute Price to represent price of the product with taxes included, and another application has an attribute with the same name but containing only the base price with the tax rate in a separate attribute.

An approach with *incremental integration* adds even more problems, for instance: the Billing application is integrated with the Accounting application. Next, it becomes strategically necessary that the Marketing application should be integrated with the previous two. Supposing the entity Client is used throughout these three applications, in the first integration we would have chosen either the Billing or the Accounting application to manage the Client entity. In the second integration, we would face the following problem: should the entity still be managed by the previously chosen application or should it be managed by the Marketing application?

These problems grow significantly with the number of applications involved. It rapidly becomes very difficult to make decisions on which applications should share, change and transform data.

3. PROPOSAL

Our proposal starts by classifying the *integration barriers* in three types:

- *Technological barriers* – when the different technologies used by applications require additional components to establish communication or to make transformations. Example: a).
- *Syntactical barriers* – when the entity's or attribute's internal structure is different between representations. Examples: b) and c);

- *Semantic barriers* – when the entity or attribute has different meaning between representations. Examples: d) and e).

As stated, barriers may appear at the entity or the attribute level.

The integration engines currently available in the market are more oriented to overcome technological and syntactical barriers and don't explicitly consider semantic barriers [Linthicum99].

The proposed methodology uses the Enterprise Architecture (EA) in an approach to applications integration, so that is possible to:

- Make the best possible use of the features in existing integration engines;
- Place applications in context and their responsibility over the data entities;
- Relate data semantics in production applications with the information in the EA;

Figure 2 represents the use of the EA-based data model to integrate applications.

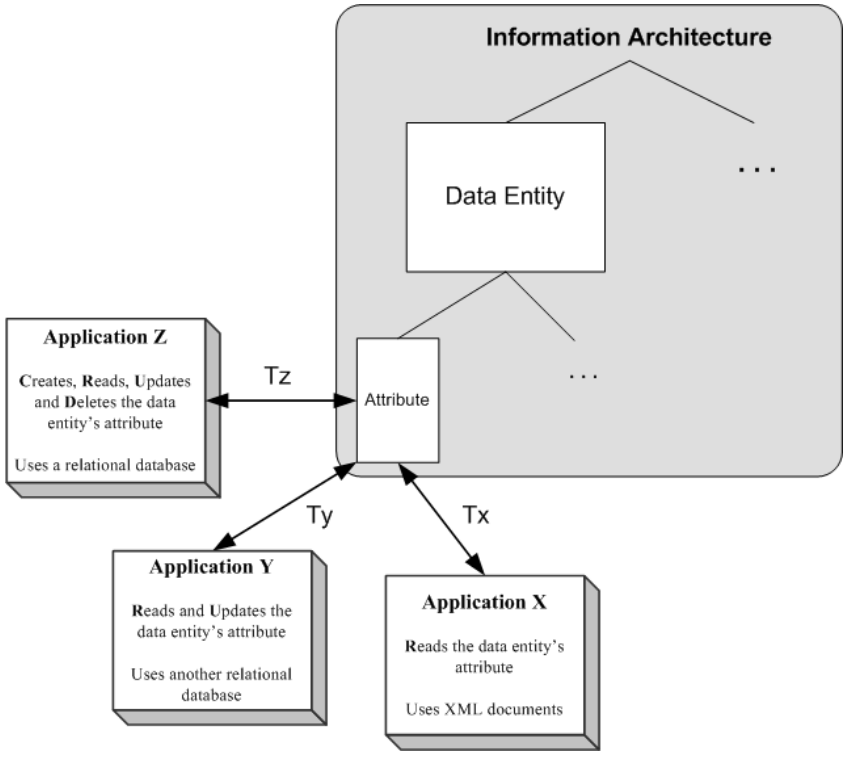

Figure 2 – Conceptual model for integration using the Enterprise Architecture

The EA defines the data entities and their attributes. X, Y and Z are different production applications that use data belonging to the same data entity's attribute. The data operations can be Create, Read, Update and Delete. The transformations Tx, Ty and Tz convert from each application's data schema to the EA attribute's data schema.

Two important *disclaimers* must be made on the use of the EA for the purpose of integration, due to the fact that it is model of reality with its own limitations:

- Uses an 80/20 conception logic – the reality is not represented with 100% accuracy and an 80/20 accuracy is often pursued [Spewak93];
- Becomes outdated – because the Organization must evolve and change to answer new challenges.

Phases of the Methodology

The integration methodology is structured in four phases:

A. Problem definition;
B. Application analysis;
C. Specification;
D. Implementation.

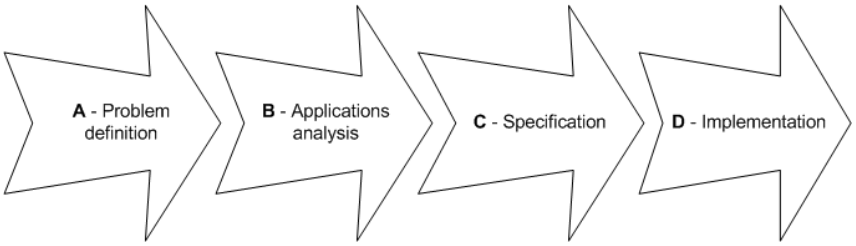

Figure 3 – Methodology phases

The problem definition (A) identifies the goal of the integration project in business and technological terms as well as the applications to be integrated. Each production application is then analysed (B) so that it is matched with the EA reality and the expected result. This is achieved through requirements analysis of production applications and semantically linking these applications to the EA applications, identifying aligned and non-aligned items. After the analysis of the production applications, the specification (C) is performed through data schemas, data flow schemas and schema mappings. In this phase, non-functional requirements such as security and availability are also defined. The implementation (D) includes development and test of the specified solution.

Phase A – Problem definition

This phase identifies the *applications* involved and describes the integration *goal*, which may be the management of a data entity or the support of a business process.

In this phase it is necessary to understand that the application integration drivers are people and organizational units within the Organization. The following factors are critical for success:

- Identify the integration project stakeholders;
- Obtain adequate top management support;
- Guarantee commitment from the people in charge of the production applications, so that the following means will be available: infrastructure, documentation, realistic data for tests in conformity with the Organization's security policy and personal data privacy laws.

Phase B – Application analysis

Interview

The analysis of each production application begins with interviews with the people in charge of the business area and the technological staff. The proposed approach for the interview is:

- *Presentation* – explain the integration project as a whole, how the production application is part of the project and define the meetings goals;
- *Inception and goals* – understand why the application was created and how it changed under the influence of different stakeholders;

- *Main data and functionality* – understand what is the core area of the application, with a brief description of main data and functions;
- *Technological platform* – identify the technologies and environments used;
- *Business/technology vision* – listen to the opinions of the interviewee concerning integrations in the production application;
- *Documentation* – define with the interviewee the documentation available, such as the physical data model, functionality and existing integration interfaces;
- *Finishing* – explain the next steps in the integration project previewing the need for further collaborations to access a stable version of the production application.

The main *results* of the interview are the physical data model, the functionalities, the existing integration interfaces and the technological platform.

Semantic linking

After the interview's results are observed, the semantic linking can start. It consists of a confrontation of the current production applications with the EA applications. For each production application:

- *Analyse the data model* – create data sets that are logically related;
- *Analyse the functionalities* – create functionality sets that contribute to the same goal or access the same data;
- *Analyse production integration* – identify the source and target production application, involved data and functionalities, direction of the data flow and operations (read and/or write);
- *Link data sets to data entities in the EA* – associate each data set identified in the production application to a data entity in the EA, checking for the data entity attributes if necessary;
- *Link functionality sets to business processes in the EA* – associate each functionality set to a business process in the EA, checking its description and its goals;
- *Identify the production application area* – mark the CRUD Matrix columns with the identified data entities and the rows with the identified business processes to obtain the production application area;
- *Link the production application to the EA* – relate the production application being analysed to the EA application with the greatest resemblance (with goals and compatible accessed data entities and business processes). This decision should be made with knowledge of the whole context, considering what is planned for the future of each production application as discussed in the interviews;
- *Identify the integration bridges in the EA using the CRUD matrix* – through observation of the data entities which are accessed (R) and created (C) in another EA application;
- *Link production integrations to integration bridges in the EA*.

The final *result* of this phase should include:

- Links between the production application and EA applications;
- Aligned items:
  - Data entities and linked data sets;
  - Business processes and linked functionalities sets;
  - EA integration bridges and linked production integrations.
- Non-aligned items:
  - Non-linked data sets;
  - Non-linked functionalities sets;
  - Non-linked integration production sets;
- Graphical representation of the CRUD matrixes used during the semantic linking to visualize the application areas.

Phase C – Specification

The specification model allows the description of integration solutions with enough detail to support its development, test and maintenance. The proposal has two modelling perspectives that are based on *entity-relationship diagrams* and *data-flow diagrams* [BatiniCeriNavathe92], extended with specific integration concepts.

Entity-Relationship for Integration

The *ER-I (Entity-Relationship for Integration)* diagrams allow building data schemas like the following example in Figure 4.

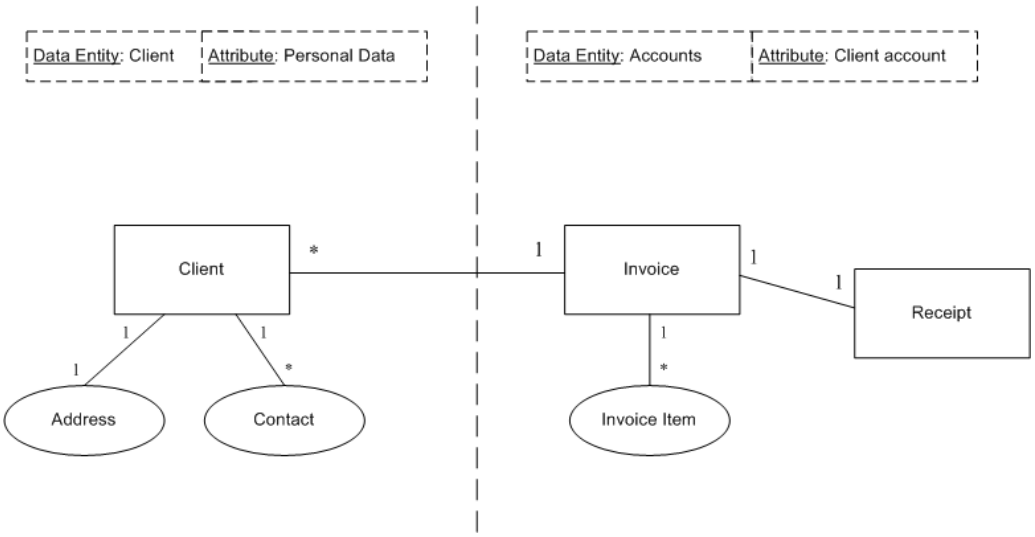

Figure 4 – ER-I schema example

The schema's graphical detail level doesn't include the simple attributes and the relationship's names. The details of the simple attributes of an entity or composed attribute are specified textually in *attribute lists*, like the one in Figure 5.

Each entity has also the *key* definitions (primary and candidates). It is important to identify all existing keys, because they can be very useful in the transformation of entities between different data schemas.

Attribute List
Entity: Client
EA Context [Data Entity: Client, Attribute: Personal Data]

PRIMARY_KEY(Number)
CANDIDATE_KEY(IDCardNumber)

Attribute	Type	Description
Number	NUMBER	Client number
FirstName	STRING	Client's first name
MiddleName	STRING	Client's middle names
LastName	STRING	Client's last name
IDCardNumber	NUMBER	Personal identification card number
RegistrationDate	DATE	Date when the client was registered
ClientType	ENUM	NRM - Normal, SPE – Special

Figure 5 – Attribute list example for the Client entity

An ER schema represents a set of entities and relationships and their respective attributes, which are subsequently translated into a single database. In an ER-I, the schema defines a part of all the information in the Organization, which can be spread across several production applications.

The *EA data dictionary* represented in Figure 6 stores the schemas in the information architecture tree, formed by the EA data entities and their respective EA attributes. Each EA attribute can contain ER-I schemas that describe a detailed view of its content.

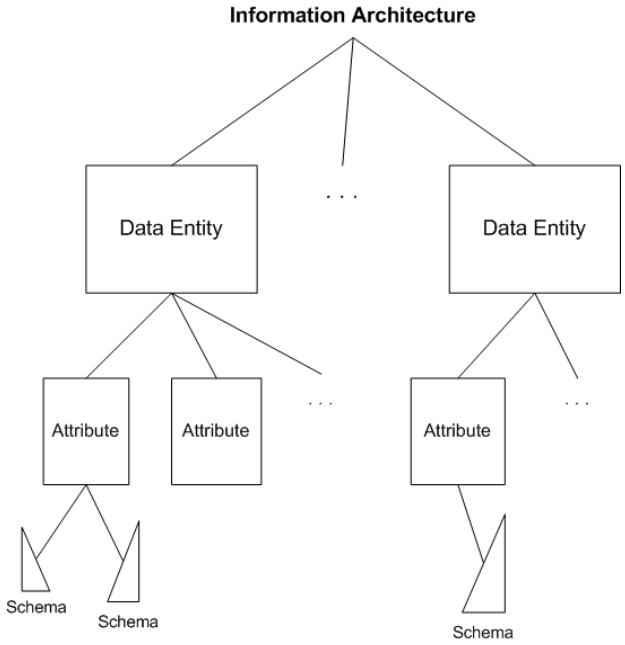

Figure 6 – EA Data Dictionary's tree structure

The EA data dictionary is never completely specified, because it contains only the schemas that represent information used in integrations and in EA data entity management.

The standard ER model [BatiniCeriNavathe92] was *extended* to support these concepts:

- EA context – EA data entity and EA attribute where the ER-I schema fits in;
- EA attribute borderline – border between different EA attributes (and possibly between EA data entities).

Each EA context in the dictionary has an *owner application* that manages the physical information of the schemas' instances. A borderline allows the representation of relationships that cross EA attributes boundaries, and therefore can be managed by different applications.

The EA data dictionary schemas are used as *intermediate data representation* in an integration solution. When data is read from an application, it's translated into an EA schema. When data is written to an application, it's translated from the EA schema.
Before doing the data modelling for a new integration problem, the EA data dictionary should be queried for existing ER-I schemas that can be reused, even if they need to be revised to include all necessary data attributes. If such a schema is not found, then it's necessary to create a new one.

In order to create a *new ER-I schema* it is necessary to:

- Choose an EA context (data entity and attribute) where the schema will fit in;
- Choose the owner application for the EA context – it should be the one that creates the data or performs most of the updates;
- Build the schema's first version with a top-down approach to define the main entities and relationships;
- Add detail to the schema with a bottom-up approach to define the entities' and relationships' attributes:
    - Start from the owner application's data representation;
    - Decompose the data attributes in its smaller parts;
    - Place the simple attributes in the most appropriate entity or relationship;
- Adjust the abstraction level, composing or decomposing entities and attributes. Group sets of related attributes in composite attributes;
- Iterate until an adequate representation is reached – one that correctly reflects the domain and that it is suitable for transformations.

The new schema is not directly influenced by the owner application's data representation, although conversion is guaranteed at the simple attribute level.

The heuristic *"it's easier to join than to split data"* should be considered during attribute composition. For example, choosing between having the Name attribute or having the FirstName and LastName attributes.

Data-flow Diagrams for Integration

The *DFD-I (Data-flow diagrams for Integration)* allows building functional schemas of data-flows, like the following example in Figure 7.

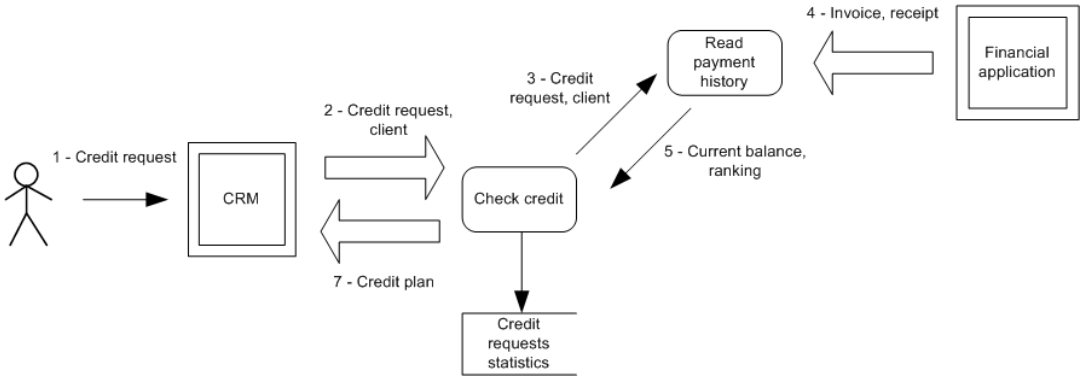

Figure 7 – DFD-I example

The data-flows and transformations have labels with sequence number and description. The sequence number represents the notion of order, while the description identifies the flowing data.

One of the assumptions in the original data-flow diagrams is that the underlying data schemas are coherent between themselves and based on the same model. In integration scenarios this is not true, because each application has its own data representation. For this reason, the standard DFD model [BatiniCeriNavathe92] was *extended* with:

- External interface – interface that belongs to an application with its specific data representation;
- Transformation – exchange and conversion of data between an external interface and an activity;
- User – used to represent human intervention when necessary.

The DFD-I core uses data specified in ER-I schemas. Its boundaries have transformations and external interfaces to applications. An application can have several external interfaces and each one is a partial view of its data representation.

A good approach to drawing the diagrams is *"outside-in"*, starting from the external interfaces towards the core.

The *transformation* is a functional pattern that can be applied to two situations: reading data from an external interface to a process or writing data from a process to an external interface. A read transformation is represented in Figure 8.

The *adapter* processes (application-side and integration-side) convert any integration interface – database access, API function calls, user interface – in a data interface, with queries and results. The adapters output application data instances, overcoming technological and some syntactic barriers.

The *transformer* process converts source schema data instances to target schema data instances, using *context information* if necessary. This process solves syntactical and some semantic barriers.

The conversion can generate *exceptions* that can abort or continue the transformation, according to the specific needs of the integration problem. Some exceptions may require human intervention to be properly handled.

The semantic barriers are solved using the ER-I schemas from the EA data dictionary.

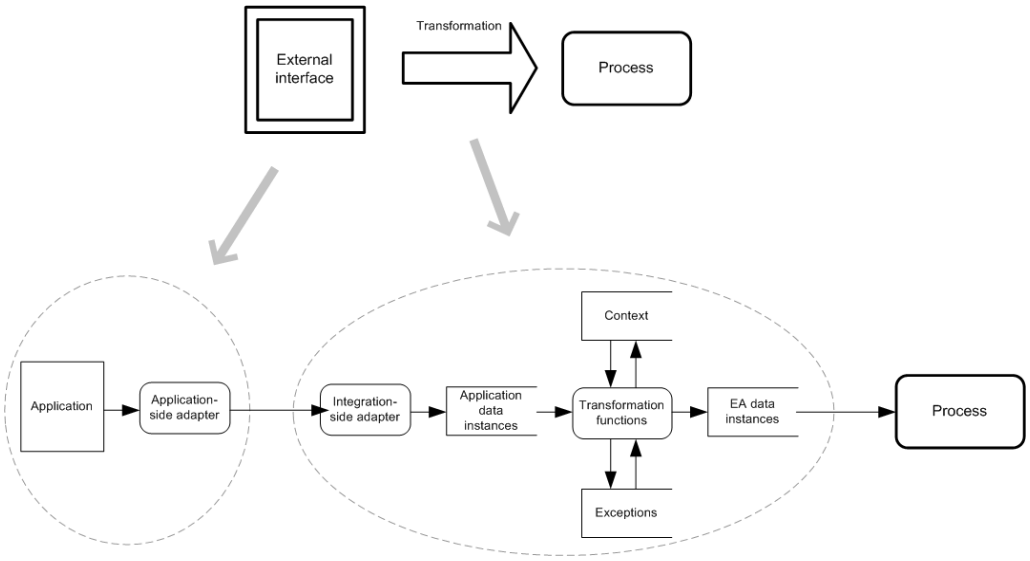

Figure 8 – Reading data from an application with transformation

A write transformation has the same components as a read transformation, but the data-flows are reversed.

Schema transformation

A schema transformation uses a set of transformation functions T to describe how to convert an instance of a source schema into an equivalent instance of a target schema. Each T function receives N source attributes and returns M target attributes.

The key transformation function (TK) maintains an entity's identity between two different schemas.

Transformation
Source - External interface: CRM_API, Application: CRM
Target - Entity: Client, EA Context [Data Entity: Client, Attribute: Personal Data]

Id	Source attributes	Target attributes	Description
TK	Cli_number	Number	Copy value
Ta	Cli_name	FirstName MiddleName LastName	Split name in parts
Tb	Cli_date	RegistrationDate	Date format conversion
Tc		IDCardNumber	empty
Td	Cli_type	ClientType	0 -> NRM 1 -> SPE

Figure 9 – Example of a transformation from an application schema to an EA schema

The DFD-I schemas identify which read and write transformations need to be specified.

A transformation function can execute at the *schema level*, considering only the data structure, and at the *instance level*, using the actual values.

The transformation function can be *stateless*, if it uses only the values of the source attributes, or *stateful*, if it also uses context information. The available context information can have different scopes:

- Data instance;
  - Current;
  - History;
- Data schema;
- Flow instance;
- Flow schema;
- Global.

Some examples of context information are:

- Table for code conversion (data schema scope);
- "Key-ring" that holds identifiers for the same entity in different schemas (data instance scope);
- Access to the transformation exceptions (flow instance scope).

The stateful transformation functions entail more configuration and administration effort, but they are indispensable to solve some integration problems. For example, to bind entities of different schemas that do not share key attributes because both use independent auto-incremented values.

The transformations are stored in the EA data dictionary in the same context as its source or target EA schema. The application schemas are not explicitly kept, instead being partially and implicitly defined in the transformations that refer to them. This decision avoids registering this information initially and then maintaining it up-to-date.

Phase D – Implementation

The implementation phase includes the development, test and maintenance of the specified solution.

The *development and execution platform* enables the creation and use of the integration solution in an efficient and scalable way. Most of the *Enterprise Application Integration (EAI)* products support the main features of an integration engine: communication capabilities, data transformation services and adapters for commercial software. However, the choice of an EAI product must be tailored to the effective needs of the Organization, excluding non-essential features that only increase its price and complexity. The investment decision must account not only with the current integration problem but also with the vision of future needs provided by the EA integration bridges.

An *adequate product* should combine the schema definition with their execution, linking the conceptual development to the operational environment. This way it's easier to keep the existential and referential integrity between the specification schemas and the implementation artefacts, promoting their maintenance and reuse.

After choosing the platform, the concepts and specifications will have to be mapped to the available tools in a standard way.

4. EVALUATION

The methodology was tested in an *integration scenario* at a Computer Science Department, for which an EA had already been developed.

The *applications* involved in the integration scenario were:

- Integrated School Management System (ISMS) – manages the back-office, teacher and student curricula;
- Student Portfolios – manages data about student extra-curricular activities;
- Teacher portal – displays teachers personal web pages.

The ISMS application belongs to the University, while Portfolios and Teacher portal belong to the Department.

The *integration goals* are to assure a coherent view of student data between Portfolios and ISMS; and to perform login into Teacher portal using each teacher's user name and password in ISMS.

The following conclusions were drawn in the methodology's *application analysis* phase:

- The student data should flow from ISMS to Portfolios, because ISMS was identified as the owner of Student data entity;
- The Portfolios' updates to student's personal data were considered non-aligned functionalities. The student entity is managed by ISMS, so updates in other applications should be performed by integration instead of requiring users to manually keep both copies up-to-date;
- The stated goal of login into Teacher portal through ISMS was confirmed to be an accurate business requirement, because the Teacher entity was owned by ISMS.

The integration was specified using ER-I and DFD-I. The Portfolios-ISMS was data modelling oriented (ER-I first) while Teacher portal-ISMS was data-flow oriented (DFD-I first). Modelling is done through successive refinements of both schemas so that the data and functionality perspective are coherent.

In this integration scenario, the methodology *proved beneficial* in the following aspects:

- Deciding data ownership by applications;
- Detecting non-aligned functionalities that were erroneously supported by existing applications;
- Helping to choose the best application to support a new functionality;
- Choosing EA dictionary data schemas that avoid many-to-many transformations, and only contain the detail strictly necessary for integration;
- Verification of mutual completion between data and functionality perspectives;
- Faster comparison of alternative solutions using graphical schemas that may also be used as a tool to enhance communication between the project team.

The integration scenario also revealed some *limitations* of the methodology:

- Prior to using this methodology, the Organization has to do a significant initial investment to develop an EA;
- Some EA assumptions can be hard to satisfy with reasonable costs;
- Communication with the people in charge of the production applications is a critical success factor for the integration project. The alignment of goals and the commitment by each part should be guaranteed – namely, in critical issues such as availability of source code and realistic data for testing;

- Production applications may need to be modified to achieve better user-interface results.

Overall, the methodology proved both useful and feasible in this practical integration scenario. It allows the integration problem scope definition in terms of business and technology, without resorting to partial visions.

5. RELATED WORK

The traditional integration methodologies do not assure the creation of a conceptual model of integration that relates data, processes and applications; both at the existing and future reality.

Britton

In [Britton00] the aspects of data modelling and associated data flows are process oriented. The business information is sorted into categories according to its use in the systems. This approach does not delimit the data entities and does not define the responsibility over its management, which makes it difficult to guarantee coherence.

Modelling is refined through prototyping, which is a pragmatic technique when the integration goals are poorly defined, and is also realistic since it may find problems, which are undetected through purely conceptual techniques. However, its cost is significant and its reuse is very limited.

Linthicum

In [Linthicum99] a semantic data model is proposed for identifying and cataloguing the Organization's data. The main difference is that it ignores different perspectives over the data – namely, which applications should be responsible by the entities creation and which applications are interested in managing each attribute. Using the EA allows us to cross the data entities with the business processes and check how the data entities are used by each process and each production application.

Web Services

Recently, Web Services [WS-I03] have been proposed as a means to integrate applications, gaining large support in the industry [HailstonePerry02]. However, they address mainly technical details and don't propose a methodology to specify the solution. In other words, they provide a way to surpass technical and syntactical barriers leaving the semantics undefined.

Service-oriented architecture (SOA) is an attempt to conceptualise the Web Services framework [Bloomberg03]. The approach is mainly functional (service-call oriented) and ignores data entity management issues.

Modelling techniques

Software engineering has produced several models and standard techniques for software development, which may be adapted to integration problems. Currently most of these proposals are not prepared to deal with detail level required to approach the entire Organization and, in this sense, are not scalable. An example of models extension was proposed in this paper, based on the reference [BatiniCeriNavathe92]. For instance, an alternative approach could be based on UML models [Fowler00].

6. CONCLUSION

The *main contributions* of this paper is a methodology that:

- Meets integration goals in the short-term, maintaining a long-term vision to ensure future coherence of present decisions;
- Has a work effort proportional to the problem's dimension – unnecessary elements are not integrated;
- Presents a semantic framing with the EA that allows the following benefits:
    - Avoids wrong integrations (misaligned with data entities, business processes and applications);
    - Checks alignment of existing applications;
    - Identifies integration needs not explicitly known during the problem's definition;
    - Structures a data dictionary containing data schemas and transformations, promoting maintenance and reuse of data assets;
- Use of widely used modelling techniques (ER and DFD), extended to represent integration problems in a familiar, concise and sufficiently objective way.

The methodology proposes solutions for some of the issues of applications integration. However, there are many subjects to explore in *future work*, such as:

- Widen the nature of the problem – with more complex data updates, business process support or more involved applications;
- Assess the need for additional concepts to represent process interaction with people using workflow systems;
- Define non-functional requirements – security, fault-tolerance, logging, etc. along with the specification;
- Explore the applicability of the methodology to bring about the convergence between the production applications and the EA applications;
- Design a set of procedures to validate existing integration solutions on production applications.

The integration specification model introduces concepts and definitions that could be improved with more test cases, preferably in different industries.

7. REFERENCES

[BatiniCeriNavathe92] Carlo Batini, Stefano Ceri, Shamkant Navathe (1992) Conceptual Database Design – An Entity Relationship Approach, Benjamin/Cummings.

[Bloomberg03] Jason Bloomberg (2003) The role of the service-oriented architect, ZapThink LLC.
http://www.therationaledge.com/content/may_03/PDF/bloomberg.pdf

[Britton00] Chris Britton (2000) IT Architectures and Middleware: Strategies for Building Large, Integrated Systems, Addison-Wesley.

[Fowler00] Martin Fowler (2000), UML Distilled – Second Edition, Addison-Wesley.

[HailstonePerry02] Rob Hailstone and Randy Perry (2002) IBM and the Strategic Potential of Web Services - Assessing the Customer Experience, IDC
http://www-306.ibm.com/software/solutions/webservices/pdf/May13_IBM_WebServices.pdf

[Inmon93] W.H. Inmon (1993) Data Architecture - The Information Paradigm - 2nd edition, QED Technical Publishing Group.

[Laudon02] Kenneth Laudon, Jane Laudon (2002) Management Information Systems, Pearson Prentice-Hall.

[Linthicum99] David Linthicum (1999) Enterprise Application Integration, Addison-Wesley Information technology series.

[MiraSilva03] Miguel Mira da Silva (2003), Integração de Sistemas de Informação, FCA Portugal.

[Spewak93] Steven Spewak, Steven Hill (1993) Enterprise Architecture Planning, John Wiley & Sons.

[WS-I03] Web-Services Interoperability organization (2003) Basic Profile 1.0a - Final Specification
http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html

Abbreviations

API	Application Programming Interface
CRUD	Create Read Update Delete
DE	Data Entity
DFD	Data-flow Diagram
DFD-I	Data-flow Diagram for Integration
EA	Enterprise Architecture
EAI	Enterprise Applications Integration
ER	Entity-Relationship
ER-I	Entity-Relationship for Integration
ISO	International Standards Organization
SOA	Service-Oriented Architecture
XML	Extensible Mark-up Language