

An Adaptable ORM Metamodel to Support Traceability of Business Requirements across System Development Life Cycle Phases

Baba Piprani¹, Marlena Borg², Josée Chabot² and Éric Chartrand²

¹ SICOM Canada

² Transport Canada

babap@attglobal.net, {BORGME,CHABJOS,CHARTRE}@tc.gc.ca

Abstract. Enterprises launching IT development projects usually start off with establishing Use Cases or similar techniques to document functional requirements as a special directed effort. More often than not, the resulting information system has buried these and other newly discovered undocumented requirements into program code--losing the important link between business requirements, business rules and developed code. In reality, the business requirements are generally surfaced over several years in memos, e-mails, meeting minutes, consultant reports etc., and the requirements gathering effort starts all over again as a new project to capture these already stated requirements. Using an ORM based development life cycle approach, the supporting adaptable traceability metamodel enables the collection and tagging of the business requirements across a multitude of documents and across development phases, to provide traceability and the facility to develop transforms across an organized information system development effort in meeting with any established System Development Life Cycles.

Keywords: Requirements traceability, SDLC, ORM, Business requirements, metamodel

1 Why requirements traceability?

When people leave organizations, more often than not, corporate memory leaves with them. The remaining staff may not remember the need for a particular requirement or business rule, why it may have changed or why it may no longer be required.

Some systems take years to come into being for a variety of reasons. There may be lengthy lapses of time between life cycle phases due to higher priority projects, lack of resources, project staff changes, etc.

A requirements traceability metamodel would reduce the possibility of losing track of the requirement throughout the life cycle. It would also provide a more systematic way to analyze business requirements and rules, identify duplicate or conflicting requirements and rules, and reduce the possibility of conflicting results since every requirement is easily traceable and can be tracked.

Another important set of deliverables from a requirements traceability metamodel would be that it provides the capability to follow the lineage of the requirement and to

ensure that the necessary business requirements are captured at a point in time; are updated as required while keeping a history and rationale for changes; and are definitively addressed somewhere in some phase of the system development life cycle.

There are several benefits from establishing the requirements traceability metamodel.

For the Organization:

- provide continuity, promotes good governance of data and protection of corporate memory
- increased stakeholder confidence in the organization with respect to products offered
- increased productivity of project staff

For the Business Client (or stakeholder):

- eliminate or reduce time and effort to continually explain and rationalize requirements
- improve confidence in the system development process and that the final product will be fully reflective of their needs
- improve confidence in system output

For the System Developer:

- requirements and business rules would be documented in one place reducing the time and effort required to analyze and track the requirements
- should project staff change, or should there be a lengthy lapse of time between phases, it would eliminate the need to repeat locating, analyzing and confirming requirements
- duplicate or conflicting requirements would be more evident, thus reducing analysis effort and the risk of overlooking such duplications or conflicts at an early stage of the development process
- facilitate and accelerate verification that all requirements and rules have been addressed by the system (i.e. improved quality assurance)

The focus of this paper is to provide an ORM schema and its associated attribute based relational schema depicting an implementation of a requirements traceability metamodel, and explain its usage in a real-life scenario.

This paper addresses the need for requirements traceability, a high level overview of the currently used ORM based system development life cycle, the ORM metamodel as used for business requirements tracking and as implemented, some usage scenarios, and the mapping to the corporate development life cycle model.

2 An ORM based System Development Life Cycle

Figure 1 below depicts a typical ORM based System Development Life Cycle initiated by propositions involving business requirement statements, that incorporates multiple phases of analyses incorporating Business Activity Modelling (IDEF0[1]), Process Modelling (BPMN[2] or BPWin and IDEF3[3]), ORM Modelling, Attribute Based Modelling (IDEF1X[4], ERWin), Event Modelling, Control Sequence

Modelling, and being realized or implemented in a Service Oriented Architecture using an Oracle physical schema and supporting BI toolsets like Crystal Reports, Business Objects---using the standard data modelling, process modelling and implementation tool sets at Transport Canada.

The following paragraphs briefly describe a summary description of each phase and the identifying concepts that are involved in the business requirements traceability for the purpose of the metamodel as depicted in this paper.

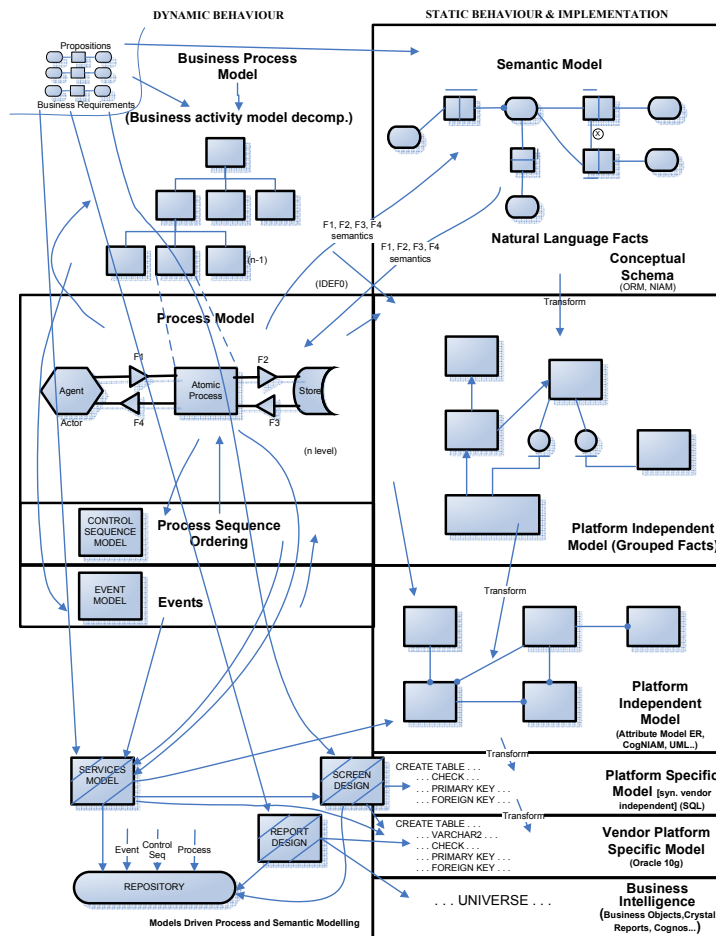


Fig. 1. An ORM based System Development Life Cycle at Transport Canada.

As can be seen from Figure 1, the progression of systems development is not a waterfall approach, nor a spiral discover-as-you-go approach, but instead, a services based architecture that has its analysis taking place in a zig-zag fashion, pretty much mimicking the way a human brain thinks. Humans tend to multi-thread and connect

several frames and concepts in their thought processes. Similarly, the analysis exercise as depicted essentially is initiated from a statement of business requirements.

2.1 The Business Activity Model to the Semantic Modelling phase

An initial forest-level picture pegging the boundaries of the set of stated requirements is defined through a form of functional business decomposition. A business activity contributes to the achievement of an objective of the business. Each business activity in the hierarchy is decomposed until the lowest level activities (called elementary business activities or atomic processes---cannot be split further) that comprise it have been identified. This lowest level activity may contain a mix of automatable and non-automatable activities where the automatable part cannot be split any further without encountering database input / output operations like Create/Retrieve/Update/Delete, or, purely an automatable process that again encounters database input / output operations like Create/Retrieve/Update/Delete, or lastly, a purely business non-automatable process that would simply contain a sequence of operations that are external to the automatable domain.

This last (or lowest) level is denoted as level "n", i.e. an automatable process. The business decomposition stops at level "n-1", i.e. the level when the activity involves an "automatable part" and still maintaining a "purely business part", and where a further decomposition results in a purely automatable process involving computer facilities input / output, or, the consumption or generation of a product. The lowest level n of a decomposition becomes a process that involves some form of input or output in an automated data processing system.

The procedures for defining business activities and the procedures for decomposition may be done differently by different people. In other words, two or more persons defining business activities and decompositions of the same business may arrive at different answers. This is quite acceptable and it does not matter, as long as all the business activities are being covered.

Why does this not matter? A business activity model is not a formal model i.e. there is not a formal grammar to support the business activity model. What matters is the data or information that is to be identified and formalized in the data usages of information flows from the lowest level process.

It is important that the lowest atomic processes represent a complete elementary task activity that cannot be split any further without losing meaning, and that these elementary tasks, while they may contain a processing sequence to accomplish that elementary task, may not be connected or sequenced with other elementary tasks---since this sequencing actually is a service and is depicted by an independent stand-alone sequence model that controls the sequence of atomic processes.

A semantic data model is derived from the data usages in the information flows of these atomic processes. A semantic data model is a formal model with formal grammar associated with it.

What this means, is that it does not matter how the business activities are organized, as long as the data usages have been recorded. No matter which alternate approaches of business activity modelling or decomposition is used---be it organizational based, product association based, business functionality based---the

data usage information flows from the lowest level processes will ultimately result in a single verifiable formal data grammar / semantic schema. This is because the final implementation is essentially being supported by a Services Model to achieve the business deliverables of the enterprise based on these agreed upon semantics. The decomposition of business activities is only a means of achieving the formalization of the semantic data model required to support the enterprise. It is the Services Model that will bring the necessary atomic processes, their necessary sequences along with pre and post conditions to enable the carrying out of the necessary services for the enterprise as derived from the requirements.

2.2 Transforms from Semantic Data Model to Neutral and Physical Data Model

It is this ORM schema that is used as the foundation for the development of an attribute based model (IDEF1X in ERWin) and further transforms to the neutral data model, and physical data model (Oracle SQL). These are generally published mappings provided by tool manufacturers or other previously published materials.

It is important to note that the data model transforms need to carry with them the maximal scope of integrity rules and constraints.

While the initial scope of the semantic data model was being defined by the data usage information flows of each elementary business activity or atomic process, one will note that this “Process Model connection” appears to be “left behind” over the transforms to the physical SQL schema.

2.3 Bringing the Processes together

Recall that the elementary business activity or an atomic process, while it may have its own internal sequence to complete the elementary task (e.g. change reservation date of hotel guest), it should not be associated with another process in any sequence except if that process is calling another process to complete its task. For example the atomic process “change reservation date of hotel guest” will require a re-usable atomic process “select hotel guest folder” which will simply fetch the current reservation and other account details of the given hotel guest.

The sequence of processes to be performed is determined by a Services Model which has a set of processes that is driven by events and in turn uses a control sequence model that determines which process is to be performed for that particular service.

Of course, the processes may require data from multiple database sources or URIs.

3 Harmonizing the Business Requirements to the SDLC model Suite

Many System Development Life Cycles (SDLC)[5] typically have a separate requirements collection phase along with identified deliverables along the way, and many embark upon this “Business Requirements Collection” journey as a search for the Holy Grail, hoping to receive a “Wal-Mart package style” set of requirements on a plate. While this is generally a useful exercise, it must be noted that the users are essentially visualizing some services-process based on a business requirement, at the root of which is a data model that can be formally defined and supported by a set of semantic models and associated supporting models.

Enterprises launching IT development projects usually start off with establishing Use Cases that determine system behaviour or similar techniques to document functional requirements as a special directed effort. More often than not, the resulting information system has buried these business requirements and other newly discovered undocumented requirements into program code, losing the important link between functional requirements, business rules and developed code---where business rules can be defined as being conformance requirements for the operations, definitions and constraints that apply to an organization in achieving its goals. In reality, the business requirements are generally surfaced over several years in memos, e-mails, meeting minutes, consultant reports etc., and the requirements gathering effort starts all over again as a new project to capture these already stated requirements.

So the question is, a) how to relate these business requirements collected over time---no matter how assembled---to the various multitude of models, and, b) how to track and define an updated lineage that is easy to trace and maintain?

- **The Business Requirements Metamodel**

Figure 2 depicts an ORM metamodel for business requirements tracking as has been implemented at Transport Canada in the Inspection Information System for the Transportation of Dangerous Goods.

Business requirements have accumulated over a number of years in the form of e-mails, meeting minutes, memos, reports, consultant studies etc.---with each set represented as a separate ‘document’ for ease of referencibility. A document is identified by a unique document identifier. A document may contain one or more requirements, each identified by a uniquely identified sequence within the document. A document may have a document current or past date or is given an approximate current or past date of being recorded. A document must have one or more authors. A document must have a unique title, and may have notes associated with it. A requirement must be identified by the originating document and a sequence number within that document. A requirement as noted in a document is re-worded as a statement of requirement, i.e. as a proposition and noted as a requirement statement description, while the original text is maintained via a document reference.

Admittedly, the requirement statement as extracted may appear to be loosely represented, and not formal since the requirement could span several areas like User Interfaces, business rules, access requirements etc. As such, a requirement here is a

association i.e. the fact type and not to the tables, columns, constraints etc. There could be other transforms that may be directly mappable to a set of tables and columns, e.g. the Z900 series of tables (see Figure 1) which are essentially an extended Information Schema tables containing metamodels to track implementation artefacts like which screen uses which column and which report uses which column, which process belongs to which service etc.

The ORM metamodel for business requirements tracking as transformed to the ER attribute based model using IDEF1X in ERWin---the Transport Canada Data Modelling standard toolset---displaying definitions and example populations for categories, status and types, is shown in Figure 3.

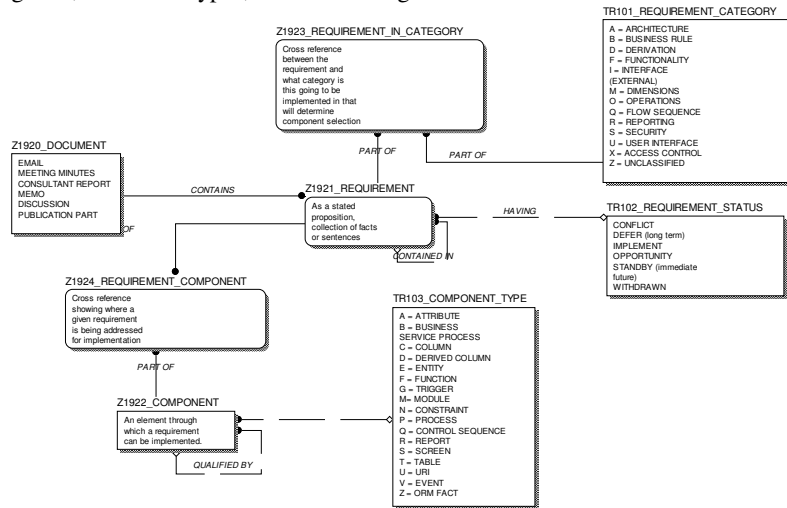


Fig. 3.. ER based metamodel for Business Requirements transformed from ORM (Definitions).

The ORM metamodel for business requirements tracking as transformed to the ER attribute based model using IDEF1X as the implemented model is shown in Figure 4.

It is important to note that for the trackability of the mappings across the multitude of models and implementation components, each model artefact and implementation component artefact is supported by a standardized form of identifier. Examples include Business Requirement identifier, Process identifier, process information flow identifier, fact type identifier, fact based constraint identifier, table identifier, column identifier, SQL constraint identifier, etc.

In its currently implemented form, the on-going collection of business requirements consists of reviewing over 150 documents collected over a span of 10 years. The makeup of the document set consists of previously defined business requirements documented in consultant reports, emails, memos, forms, graphs as requirements, project documentation, and reference materials like Acts, Regulations, guidelines that need to be conformed to. So far, we have extracted over 500 business requirements which are trackable based on the requirements traceability metamodel as implemented in Oracle 10g, a sample of which is shown in Table 1 .

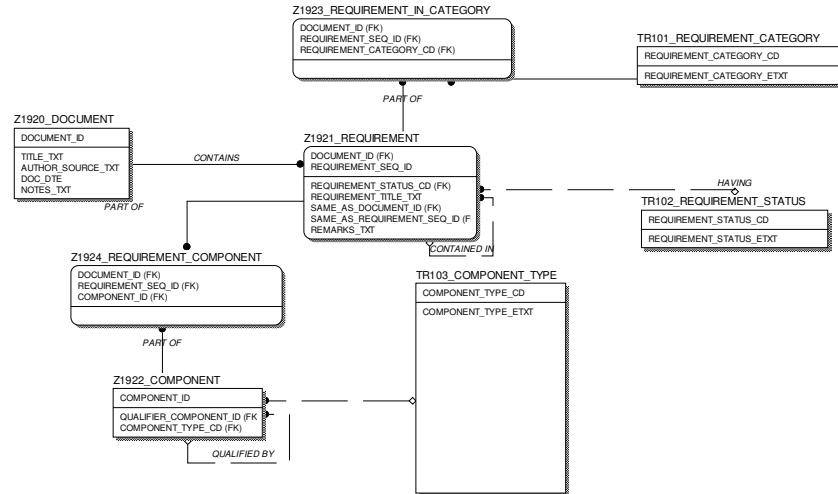


Fig. 4. ER based metamodel for Business Requirements transformed from ORM (as implemented).

Table 1. Example extract from on-going requirements tracking.

DOCUME NT_ID	REQUIR EMENT_ SEQ_ID	REQUIRE MENT_ST ATUS_CD	REQUIREMENT_TITLE_TXT
IIS-004	40		Allow searching by manager name
IIS-004	41		Allow searching by company name
IIS-004	42		Allow searching by province
IIS-004	43		Capture contact's e-mail address
IIS-004	44		Produce a graph of companies having high violations
IIS-004	45		Produce a graph of companies having high instance of certain violations
IIS-005R1	1		Allow user to view and edit tombstone data
IIS-005R1	2		Capture business type
IIS-005R1	3		Capture dangerous good handled
IIS-005R1	4		Capture means of containment used
IIS-005R1	5		Capture means of transport used
IIS-005R1	6		Keep a historical record of previous data values

4 Corporate SDLC Correlation Mapping

Transport Canada uses Macroscopic Productivity Centre [5] as the SDLC standard set of documentation and practices in support of their IT system development projects. Each life cycle phase is supported by one or more sets of deliverable documents that

contain specifications at major life cycle stages. The above ORM model as shown in Figure 2 already contains a construct identified with the fact type 'Document contains Requirement'. A Macroscopic document identifier is created and is associated with related requirements. In this way, the business requirements from various sources are captured and stated as propositions, and these propositions are captured and documented in the format and style as required by the stated template for Macroscopic documentation, thus meeting Transport Canada corporate standard requirements.

5 Conclusion

The business requirements traceability metamodel provides the much required requirements lineage and more importantly, captures all facets and incarnations of business requirements 'as-they-happen'. The metamodel enables the tracking of documents, tracking of actual requirements, projected into the realization and implementation of the stated requirements. The metamodel allows navigability across multiple models involved in the systems development life cycle for an organization, and supports the zig-zag or other development processes like agile, waterfall, prototyping, spiral etc.

References

1. Mayer, R.: IDEF0 Functional Modeling. Knowledge Based Systems, Inc., College Station, TX (1990).
2. Object Management Group (OMG): Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification (2006).
3. R. Mayer, C. Menzel, M. Painter, et al.: Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report, Knowledge Based Systems Inc. (KBSI) (1995).
4. Appleton Company, Inc.: Integrated Information Support System: Information Modeling Manual, IDEF1 – Extended (IDEF1X), ICAM Project Priority 6201, Subcontract #F33615-80-C-5155, Wright-Patterson Air Force Base, Ohio(1985).
5. Macroscopic ProductivityCentre, by Fujitsu Consulting, USA, see <http://www.fujitsu.com/us/services/consulting/method/macroscopic/prodcentre/index.html>