A Meta-modelling Approach to Express Change Requirements

Anne Etien, Colette Rolland, Camille Salinesi

CRI - Université Paris 1 - Sorbonne
90, rue de Tolbiac, 75013 Paris - France
{aetien, rolland, camille}@univ-paris1.fr

Abstract. Organisations have to evolve frequently in order to remain competitive and to take into account changes in their environment. We develop a co-evolution approach to jointly make evolve the information system and the business processes. This approach relies on an explicit specification of change requirements defined with operators expressing gaps between the As-Is and the To-Be situations. However, such gaps based approach can also be used in an other evolution context, when a database or a workflow model evolves. Thus, instead of specifying new operators associated to the Map meta-model used in this co-evolution approach, we propose to define a generic typology of gaps to facilitate a precise definition of change requirements under the form of gaps. The paper presents the approach to generate a gap typology and illustrates it with the Map meta-model.

1 Introduction

Changes often affect an organization in its whole from business processes to information system. If they want to remain competitive, organisations have to react quickly to changes of their clients' needs or organization goals.

We propose the Alignment and Co-Evolution Method (ACEM) to help in jointly evolving the business processes and the system. In that method, the change movement is modelled from the current situation to the future situation as *gaps* between the *As-Is model* and the *To-Be model*. Intuitively a gap expresses a difference between these two models such as the deletion or addition of an As-Is element in the To-Be model. Gaps are related to *operators*, which transform elements of model.

We believe that an ad-hoc development of a gap typology for each project is error prone because: (i) it relies on the knowledge and know-how of some persons; (ii) it is not systematic and (iii) it can be influenced by the context of the project.

We thus could define a typology associated to the Map meta-model used in ACEM. However, such a typology would have been dependent of the used formalism.

Furthermore, developing a specific typology for each meta-model (e.g. XML DTD [1], DB meta-model [2], process meta-model [14], workflow meta-model [4]...), leads to a situation where the typologies depend on different specific meta-models and are difficult to compare [5].

In order to solve these issues, we propose to introduce a generic typology relative to a generic meta-model. This provides independence towards the project and the meta-model. The generic meta-model can be instantiated by each used meta-model. The generic typology associated to the generic meta-model is adapted to correspond to each specific meta-model. Such an approach allows to systematically identify the semantic and structural aspects that compose the specific meta-model and can be affected by a gap.

In the next section we provide an overview of the approach. In section 3, we present the generic meta-model and the generic gap typology. Section 4 outlines the process to generate a specific gap typology and illustrates it with the Map meta-model used in ACEM. Conclusions are drawn in section 5.

2 Overview of the approach

The approach, we propose to express change requirements, relies on a three levels structure: the model level, the meta-model level and the generic meta-model level, as shown in Fig. 1.

At the *model level*, are defined the models before and after evolution. At this level are also defined the change requirements (represented in the figure by the Greek letter Δ) under the form of gaps. In so far as in the ACEM As-Is model and To-Be model are defined with the Map meta-model, we make the hypothesis that the two models As-Is and To-Be are described in the same language. We are thus not interested in evolutions where As-Is and To-Be models are instances of two different meta-models as in [16] or [3].

The *meta-model level* contains the specifications of a specific meta-model and the associated gap typology. The specific meta-model specifies the type of elements used in the As-Is and To-Be models. From the same way, the specific gap typology specifies the type of gap operators defined at the model level. The gaps identified between the As-Is and the To-Be models are instances of the specific gap typology.

The *generic meta-model level* proposes a generic gap typology and a generic meta-model from which are respectively defined the specific typology and the specific meta-model. The generic meta-model identifies the generic concepts necessary to the definition of generic operators gathered in the generic gap typology. The generic meta-model allows to make explicit the elements and the structures of the specific meta-models.

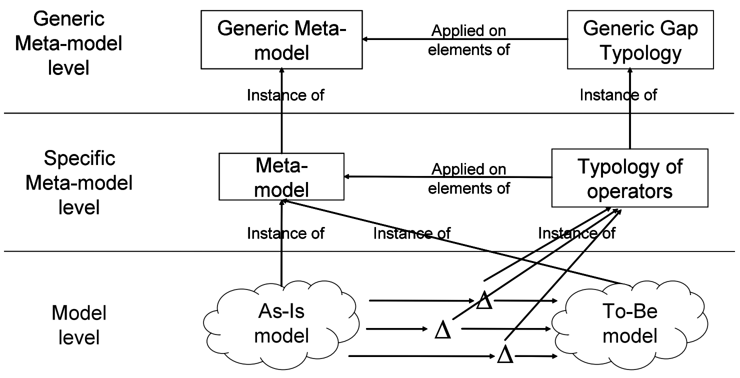

Fig. 1. Overview of the approach

For example, if the Entity-Relationship meta-model is used to represent the database, then the gaps are expressed at the model level between two Entity-Relationship models. The gaps between the As-Is and the To-Be models expressed what changes or should be adapted between the two situations. They instantiate the operators of the specific typology. They can express that the Reservation Entity type should be split into two Entities type Reservation and Demand and that the 'correspond' Relationship type (whose source is Reservation and target is Demand) should be added.

3 The generic typology

The generic gap typology takes the form of a set of operators applicable to generic elements that compose any model.

3.1 A meta-model for defining the generic gap typology

A number of attempts have been made to make explicit the elements that compose any model, i.e. to define meta-models [8], [10], [11]. There are different meta-models depending on the meta-modelling purpose. For example IRDS [8] is a standard to facilitate the evolution of model representation in CASE tools, Prakash [11] aims at a formal definition of a method and Marttiin [10] searches for a generic repository structure of meta-Case environments. The generic meta-model, that we propose, aims to identify the key elements and the structure of any meta-model having a graphic representation in order to define the elementary transformations that can occur on the elements of a meta-model.

This meta-model is drawn in Fig. 2 using UML notations. It shows that any model is made of *Elements,* every element having a *Name* and is characterised by a set of *Property*. In the E/R model for example, *Entity type, Attribute* and *Relationship type* as well as the *Is-A* relationship are *elements*. *Domain* is a *property* of *Attribute*.

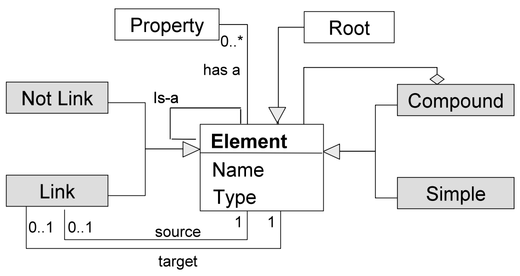

Fig. 2. The meta-model for gap typology definition

According to the generic meta-model, any meta-model is composed of a collection of elements that have properties. As shown in Fig. 2, *Elements* are classified into two clusters. First, a distinction between *Simple* and *Compound Elements* is made. Second, elements can be classified into *Link* and *NotLink*.

Compound elements are composed into elements that can be simple or at their turn compound. In particular, any model is a compound element.

Link Elements are connectors between pairs of elements. Links can be oriented; therefore one of the linked elements plays the role of *Source* and the other of *Target*. In the E/R model an *Entity type* is a compound element made of *Attributes*, which are simple elements. An *Is-A relationship* of the E/R model is a *Link*: it connects a source *Entity type* to a target *Entity type*. Vice versa, an *Entity type* is *NotLink*.

Fig. 2 shows that an element *is-a* another element, i.e. might inherit from another element Finally, any model is a compound element which can be reduced to the *root* element (such as the Object class in a class diagram).

3.2 The generic gap typology

The *generic gap typology* is composed of a set of *operators* applicable to *Element*. Each operator identifies a type of change that can be performed on an As-Is model. The operator identifies the difference between the As-Is model and the To-Be model.

Three types of change. The generic gap typology identifies three major types of change: *naming* changes, *element* changes and *structural* changes.

Naming changes are defined with the *Rename* operator. It only affects the way organisations want to refer to an element.

Element changes affect elements and are circumscribed to the elements themselves: adding an *attribute* to an *entity type* is an example of such localised change. Table 1 proposes four operators to specify element changes, namely *Modify*, *Give*, *Withdraw* and *Retype*.

Structural changes correspond to a modification of the set of elements which composes the model. There are nine operators to specify structural changes in Table 1: *ChangeOrigin*, *AddComponent*, *MoveComponent*, *RemoveComponent*, *Replace*, *Split*, *Merge*, *Add* and *Remove*. For example adding or removing *Relationship types* and *Entity types* in an *As-Is* E/R schema to form the *To-Be* schema is a structural change.

Table 1 sums up the *generic gap typology* composed of 14 operators classified according to the type of *Element* they are applied on.

Table 1. Meta-model elements and related operators

Object	Operator	Description
Element	*Rename*	Change the name of the element in the To-Be model
	Add	Add an element in the To-Be model
	Remove	Remove an element of the As-Is in the To-Be model
	Merge	Two separate As-Is elements become one in the To-Be model
	Split	One As-Is element decomposes into two To-Be elements
	Replace	An As-Is element is replaced by a different To-Be one
Link	*ChangeOrigin*	The source or target of the link is changed
Compound	*AddComponent*	A component is added in the To-Be element
	RemoveComponent	An As-Is component is removed in the To-Be element
	MoveComponent	A component is repositioned in the structure of the To-Be element
Property	*Give*	Add a property to the To-Be element
	Withdraw	Remove an As-Is property in the To-Be element
	Modify	Change the property of the To-Be element
	Retype	The As-Is and To-Be elements have different types

Structure of a generic operator. The definition of the operators relies on two concepts: a *signature* and a *predicate* as shown in Fig. 3.

The *signature* identifies the type of the elements involved in the As-Is model (before the operator is executed), and in the To-Be model (after the execution of the operator). The *predicate* is composed of two elements: a first order logic formula and eventually some parameters. The *formula* does not indicate how to modify the As-Is model but specifies the conditions that must be fulfilled in the To-Be model. It relies on the concepts of the specific meta-model (a concept being an *Element* or a *Property*). A *parameter* refers to a concept.

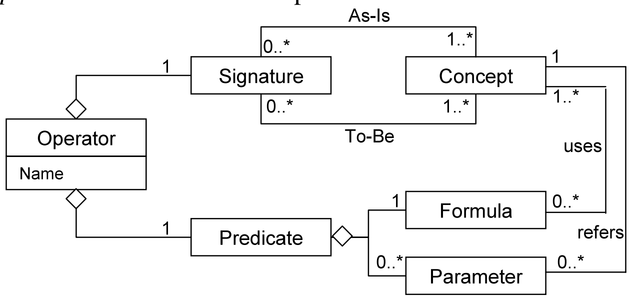

Fig. 3. Model of operator

In order to take into account the concepts of the generic meta-model and the links that exist between them, we introduce some functions that are used in the formula such as has-for-source() that is applied on a *Link* and that takes in parameter an *Element*. This function allows to specify the *Element* that is the source of the *Link* element. We can thus write L.has-for-source(E) where L is a *Link* and E is an *Element*.

From this structure, this function and four other ones, the fourteen operators of the generic typology can be formally defined.

The operator *Add* is differently defined depending on whether the element to add is a *Link* element of a *Not Link* element:

(signature) AddLink: $\text{NotLink}^2 \rightarrow \text{Link}, \text{NotLink}^2$

(predicate) AddLink (NL_1, NL_2) = L ∈ M ∧ L. has-for-source(NL_1) ∧ L. has-for-target (NL_2) | L ∈ Lien, NL_1, NL_2 ∈ NotLink, M ∈ Model

(signature) AddNotLink: Model → NotLink

(predicate) AddNotLink (M) = NL ∈ M | NL ∈ NotLink, M ∈ Model

The operator *AddLink* allows to add a *Link* L between two *NotLink* elements NL_1 and NL_2. After application of the operator, in the To-Be model, L is an element of the model M. L has for source NL_1 and for target NL_2.

The operator *AddNotLink* allows to add the *NotLink* element NL in the model M. After application of the operator, NL belongs to the model M.

The model is always present before and after the application of the operator. It appears as element in the signature, only when it is the only element specifying the As-Is or the To-Be situation, as in the definition of the *AddNotLink* operator.

All the other operators are described from the same way (more details can be found in [6]).

Properties of the generic typology. From the literature, we identify properties that a gap typology should fulfil: a typology is considered as (i) *complete* if any model can be derived from any other model [9]; (ii) *correct* if each operator is correct i.e. it does not leave the model in an incorrect state [2], (iii) *consistent* if the definition of its operators do not conflict each other [15], (iv) *semantically rich* if any type of change can be expressed using only one operator and (v) *minimal* if any operator can be considered as the composition of others [4]. These two last properties are contradictory and can not be fulfil at the same time.

The generic typology verifies each of these properties. Based on [2], we demonstrate in [12] and [6] that the generic typology is complete. The verification of the consistency and correctness relies on the formal definition of the operators. Finally, it is clear that the typology is semantically rich what allows to better answer to the customer requirements expressing, e.g. merger or replacement of elements.

4 Generation of a specific gap typology

We propose a process to generate a typology associated to a given specific meta-model from the generic typology. We then illustrate it by specifying a typology associated to the Map meta-model.

4.1 Description of the generation process

The process that allows to generate a gap typology associated to a specific meta-model, is composed of six steps:

1. *To choose the properties to reach*, particularly the minimality or the semantic wealth of the specific typology. Indeed, the set of operators to instantiate are not the same. To reach the minimality, only the generic operators *Give*, *Withdraw*, *Add*, *Remove*, *AddComponent* and *RemoveComponent* are instantiated. To satisfy the semantic wealth property all the operators of the generic typology are instantiated in the third steps.
2. *To instantiate the generic meta-model*. This step aims to build the specific meta-model by instantiation of the generic meta-model.
3. *To instantiate the generic typology*. This step uses the generic typology to generate, by instantiation a specific typology. According the choice made in the first step, all operators or only those forming the minimal set are instantiated for each concept according to its generic type *Link*, *NotLink*, *Composed*, *Simple* or *Property*.
4. *To remove the non-sense operators*. This step allows to prune the operators that would not have sense or would not be used in the context of the specific meta-model.
5. *To formally define the operators*. This step relies on the formal definition of the generic operators and on the knowledge of the specific model in order to formally define each specific operator.
6. *To verify the different properties*. This last step corresponds to the evaluation of the properties previously identified. During this step, the specific typology can be modified in order to satisfy the different properties.

4.2 Illustration of the generation process

The Map meta-model [13] used in ACEM provides an intentional representation of the system and the business processes. A map is a labelled directed graph from *Start* to *Stop* with *intentions* as nodes and *strategies* as edges. A map is composed of several sections; one section being an aggregation of two intentions linked through a strategy (cf. Fig. 4).

We chose to construct a semantically rich typology in order to better express the change requirements. For sake of space, we do not detail each of the six steps; we give the intermediary important results.

Instantiation of the generic meta-model. Fig. 4 shows the instantiation of the generic meta-model for the Map meta-model.

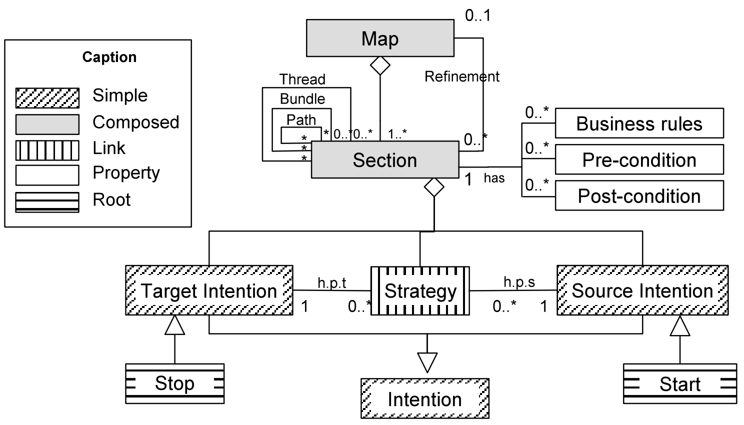

Fig. 4. Instantiation of the generic meta-model for the Map meta-model

An *intention* is a *NotLink* element corresponding to a goal that can be achieved by the performance of a process.

A *strategy* is a manner or a means to achieve an intention. In Fig. 4, a *Strategy* is shown as a *Link* element. As a link, a strategy has a source which is the *Source Intention* and a target which is the *Target Intention*.

A *section* is an aggregation of the *source intention*, the *target intention,* and a *strategy*. A section is thus a *composed* element. Furthermore, a section can be seen as the transition from an initial situation obtained by the realization of the source intention towards a final situation resulting from the enactment of the target intention by application to business rules linked to the section. These aspects are specified by three *Properties* associated to the section element: the *pre-condition* (characterising the initial state), the *post-condition* (reflecting the final state) and the *business rule*.

Sections are connected one another according to three different links: a *path* (establishing a precedence/succedence relationship), a *thread* (specifying that sections between a pair of intentions are alternative) or a *bundle* (when sections between a pair of intentions are mutually exclusive). These three elements are of type *Link*.

Finally, let us mention that it is possible to *refine* a section of a map at level *i* into an entire map at a lower level *i+1* to view an intention together with its strategy as a complex graph of intentions and their associated strategies. Refinement as defined here is an abstraction mechanism by which a complex assembly of sections at level i+1 is viewed as a unique section at level *i*. The refinement is a *Link* element.

Instantiation of the generic typology. The instantiation of the fourteen generic operators (Table 1) for the specific elements of the Map meta-model allows to obtain a table with eight columns corresponding to the number of elements in the Map meta-model (intention, strategy, section, map, refinement, path, bundle and thread).

The nature of the elements (Link, NotLink, Simple, Composed) helps in reducing the number of specific operators in the typology. For example, the operator ChangeOrigin can only be instantiate for the elements strategy, refinement, path, bundle, thread, relationship and alignment relationship that are *Link* elements.

Some operators have been removed from the typology insofar as they have no sense (step 4), as for example *AddComponentSection* or *RemoveComponentSection*. Indeed, the structure of a section is immutable: a source intention, a target intention and a strategy.

Finally some operators are removed in order to satisfy the chosen properties (step 6). Thus, for example, a typology containing the operators *AddSection* and *AddSectionMap* is not consistent since these operators have the same formal definition.

Table 2 shows an extract of the obtained table at the end of the generation process.

Table 2. Extract of the typology associated to the Map meta-model

Operator	Intention	Strategy	Section	Map pivot	Refinement
Rename	RenameIntention	RenameStrategy	N/A	RenameMap	N/A
Add	AddIntention	AddStrategy	N/A	AddMap	AddRefinement
Remove	RemoveIntention	RemoveStrategy	N/A	RemoveMap	RemoveRefinement
Merge	MergeIntention	MergeStrategy	MergeSection	MergeMap	N/A
Split	SplitIntention	SplitStrategy	SplitSection	SplitMap	N/A
Replace	ReplaceIntention	ReplaceStrategy	ReplaceSection	ReplaceMap	N/A
Change Origine	N/A (Non Applicable)	ChangeSourceIntention ChangeTargetIntention	N/A.	N/A.	N/A.
Retype	RetypeIntention	RetypeStrategy	N/A	N/A	N/A

This approach has been used in different industrial projects as for example with DIAC, the financial branch of the French constructor Renault. We developed a typology for the Map meta-model. The evolution based on gap elicitation allows to construct the To-Be model by focussing on change without defining again what remain unchanged.

Conclusion

System adaptation is done under intense time pressure: the new system must be put in place yesterday. Therefore, it is not possible to develop a To-Be model *from scratch*, given the time and resources involved. A workable strategy under these circumstances is to use and modify what is available, and add the remaining. This is the thrust of the gap drive proposed in this paper.

In this paper, we have proposed an approach to identify operators expressing change requirements. This approach relies on the existence of a generic meta-model and a generic typology.

The process that we have defined in this paper allows to systematically generate a specific typology satisfying the properties of completeness, correctness, consistency and semantic wealth. From this way: (1) any change can be expressed by the set of the typology operators; (2) the application of each operator let the system in a coherent state without introducing new errors; (3) the operators definitions are clear and non ambiguous and (4) each type of change can be expressed by using only one operator.

Furthermore, there are some advantages of proceeding following the proposed approach: (i) the generic typology serves as a guide to define the specific typology:

the latter is just an instance of the former and (ii) specific typologies are consistent with each other as they are generated from the same mould: this is important when several typologies are used in the same method.

The illustration of this process to define a specific typology associated to the Map meta-model has shown its relative simplicity and its systematic aspect. We have use this process in [12] to define a specific typology associated to the intentional Map meta-model and in [7], we generated typologies respectively associated to WIDE [4] and ORION [2].

References

[1] Al-Jadir L. (2003) "Once Upon a Time a DTD Evolved into Another DTD" Object Oriented Information Systems, Lecture Notes in Computer Science, Vol.2817, pp.226-237, 2003.

[2] Banerjee, J., Kim, W., Kim, H.-J., Korth, H. F.: Semantics and Implementation of Schema Evolution in Object Oriented Databases In Proc. of the ACM-SIGMOD Annual Conference, pages 311--322, San Francisco, CA, May 1987.

[3] J. Bezivin From Object Composition to Model Transformation with the MDA, Proceedings of TOOLS, USA, Santa Barbara, August 2001

[4] F. Casati, S. Ceri, B. Pernici, G. Pozzi Workflow Evolution. In Proc. of 15th Int. Conf. On Conceptual Modeling (ER'96), Cottbus, Germany, pp. 438-455, 1996

[5] J. Estublier and M. Nacer Schema Evolution in Software Engineering Databases -- A new Approach in Adele environment CAI Computer and Artificial Intelligence Journal. June 2000. Vol 19. pp. 183-203.

[6] Etien, A.: Ingénierie de l'alignement : Concepts, Modèles et Processus Thèse de Doctorat, Université de Paris I, Mars (2006)

[7] Etien, A., Salinesi, C.: Towards a Systematic Definition of Requirements for Software Evolution: A Case-study Driven Investigation. Proc of EMMSAD'03 Velden, Austria, 2003.

[8] Information Technology-Information Resource Dictionary System (IRDS) – Framework, ISO/IEC International Standard, 1990.

[9] M. Kradolfer. A Workflow Metamodel Supporting Dynamic, Reuse-based Model Evolution. PhD thesis, Department of Information Technology, University of Zurich, Switzerland, mai 2000, chap. 4, pp. 59-73

[10] Marttiin, P., Methodology Engineering in CASE shells: Design Issue and Current Practice, PhD thesis, Computer science and information systems reports, Technical report TR-4, 1994

[11] Prakash, N.: On Method Statics and Dynamics. Information Systems 24(8), pp. 613-637, 1999.

[12] C. Rolland, C. Salinesi, A. Etien, "Eliciting Gaps in Requirements Change". *Requirement Engineering Journal* Vol. 9, N°1, pp1-15, 2004

[13] Rolland, C., Prakash, N., Benjamen, A:.A Multi-Model View of process Modelling, Requirements Engineering Journal, (1999) 4 : 169-187

[14] Soffer P, Wand Y (2004) Goal-Driven Analysis of Process Model Validity. Proceedings of CAiSE'04, Riga, Latvia.

[15] Teeuw W. B., van den Berg H. (1997), On the Quality of Conceptual Models, Proceedings of the 16th International Conference on Conceptual Modeling (ER'97), Los Angeles, CA, November 1997

[16] M-N Terrasse, M. Savonnet, G. Becker, E. Leclercq, "UML-based Metamodeling for Information System Engineering and Evolution", Proceedings of OOIS'03, LNCS 2817, Springer Verlag, pp. 83-94, 2003