Fitting Business Models to System Functionality: Alignment issues and challenges

C. Rolland

Centre de Recherche en Informatique
Université Paris 1 – Panthéon Sorbonne
90, rue de Tolbiac 75013 Paris – France
rolland@univ-paris1.fr

1. Introduction

Alignment between IT systems and the business they support has been considered as a key issue for several years by both researchers and professionals. As [Lufman04] [Reich03] [Tallon02] [Watson97] or [Brancheau96] already showed it, strategic alignment between IS and business objectives is a top priority for CIOs and IT executives. There is also, a large corpus of empirical and theoretical evidence that more tactical alignment between information systems and the business processes they support improves organizational performance (e.g. [Chan97], [Kefi05], [Chan02], [Sabherwal01], [Croteau01]). Furthermore, studies highlight the lack of alignment as a major cause for business processes failure in providing return on investments. If alignment clearly appears as desirable, a number of issues still remain unsolved as for example: (i) the achievement of BP-IS alignment and, (ii) its management over time [Hirschheim01], but also (iii) the identification of non fit and (iv) its evaluation [Soffer04].

In advocating to establish the relationship between the "whys" and the "whats" in order to ensure that system functionality matches organizational requirements, Requirements Engineering (RE) inherently deals with alignment, but not often in an explicit manner.

My position it that System/Business alignment at the strategic level as well as a more operational/tactical level is a challenge for RE that we could discuss at the Workshop.

In order to introduce my position on the subject, I (partly) explore in the following what I will refer to (inspired by Colin Potts [Potts97a]) as the *fitness relationship* to reveal its nature and its engineering process. In this exploration, I shall consider the twin problems of establishing the fitness relationship and preserving it in the face of change. The presentation is organized around two sets of issues related to (a) *understanding the fitness relationship and (b) engineering the relationship.*

I will also take the position that intentional modeling can help resolving some of these issues and shall use the MAP representation formalism that we experienced in a number of large size projects, as an example.

2. Understanding the fitness relationship

Issue1: **Conceptual mismatch**

It has been recognized that there is a *conceptual mismatch* between the business and the system languages [Arsajani01][Alves02] and that it is considered necessary to minimize this mismatch. One way to obviate this issue is to leverage the functional system view to a requirements view and to express both with the *same language*. This shall allow to express the fitness relationship in a more straightforward manner. Goal centred languages seem to be the most adequate to this purpose as they explicitly capture the *why* and *how* of both system functionality and business process. With this position, the mismatch resolution requires on one hand, to abstract intentionality from the functionality and, on the other hand, to formalize the business in intentional terms.

Our experience is based on the use of the *Map* representation system for a *uniform representation of business goals and system functionalities*. See [Rolland01] for a detailed description. A brief overview is as follows: A map is a labelled directed graph (see Fig. 1) with *goals* as nodes and *strategies* to achieve them as edges. The directed nature of the graph shows which goals can follow which one. An edge enters a node if its strategy can

be used to achieve the corresponding goal. The key element of a map is a section defined as a triplet $<G_i, G_j, S_{ij}>$ and represents a way to achieve the target goal G_j from the source goal G_i following the strategy S_{ij}. Each section of the map captures the condition to achieve a goal and the specific manner in which the process associated with the target goal can be performed. Sections of a map are *connected* to one another :

(a) when a given goal can be achieved with different strategies. This is represented in the map by several sections between a pair of goals. Such a map topology is called a *multi-thread*.

(b) when a goal can be achieved by several combinations of strategies. This is represented in the map by a pair of goals connected by several sequences of sections. Such a topology is called a *multi-path*. In general, a map from its *Start* to its *Stop* goals is a multi-path and may contain multi-threads

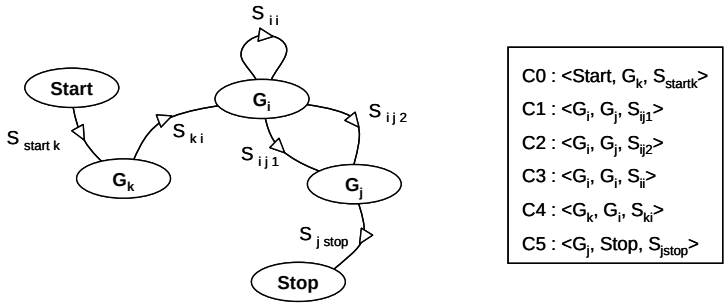

Fig. 1. A map.

As an example, the map of Fig. 1 contains six sections C0 to C5. It can be seen that C1 and C2 together constitute a multi-thread whereas {C4, C1} and {C4, C3, C2} are two paths between G_k and G_i constituting a multi-path.

Issue2: Modeling the relationship

Despite the fact that the fitness relationship is mentioned in many approaches, it is rarely a concept per se. My position is that is should be. This will allow to clarify the concept, make it purposeful and reason with it. Besides, since the fitness relationship deals with the system as well as the business viewpoints, *modeling the fitness relationship* should provide two faces, one for understanding the system viewpoint and the other for the business viewpoint. Thus, the representation should integrate these two viewpoints suitably.

A survey of literature shows that most approaches use links adopting the Regev's view [Regev04] of alignment as "the correspondence between a set of components". Different types of links have been proposed. *Traceability* links like in [Clarke99] or [Pohl94] to facilitate the impact of changes are one option leading to a *loose coupling* between entities. Another variation are *derivation links* such as in [de Landtsheer03] who proposes a technique for deriving event-based specification from KAOS specifications. Similarly [Krishna04] identifies links between concepts of i* and those of Z in order to transform changes of an i* model into modifications of a Z specification. Derivation rules also familiar to GORE approaches in RE correspond to a more *direct coupling*.

A more advanced solution completes links with *metrics*. For instance, [Bodhuin04] proposes to define consistency using *metrics*. Soffer[Soffer04] suggests that identification of unfit requires the application of a fit measurement method. In [Etien05b] Etien proposes criteria and associated generic metrics to quantify to which extent there is a fit between related business and system models.

Finally, some authors advocate the use of a *common language*. This approach is recommended by Clarke to address the misalignment of design and code [Clarke99] and used by SysML [sysml.org] in order to understand relationships between different UML domain dependent models. This is the position we adopted using the MAP formalism as the common language. The ideal fit between system and business occurs when there is a perfect match between the business map and the system map. In that case, any section can be regarded from two viewpoints : the *business viewpoint* and the *system viewpoint*. As a result, a map section expresses a direct relationship between a system function and a business process. As the world is not always ideal, we defined in [Etien05a] distance measures to evaluate to which extend two sections of a system map and business map (which define a fitness relationship) deviates one from the other. In dealing with strategic alignment we defined in [Thevenet07] links such as is necessary to, is useful to, is sufficient to, is constrained by, is contradictory with.

Issue3: **Refining the relationship**

It may be inconvenient to view the fitness relationship as one monolithic flat structure. A layered approach may facilitate understanding. Therefore, in order to present the fitness relationship at different levels of detail, it is necessary to have a *refinement* mechanism as well as a means to control the *quality* of the refinement.

Refinement is an abstraction mechanism by which a given entity is viewed as a set of interrelated entities. Refinement is known as a means to handle complexity. My belief is that such *refinement mechanism* is required for handling the fitness relationship in a systematic and controlled manner. Indeed, it would be inconvenient to view in one shot, a fitness relationship as one monolithic, flat structure. A layered approach may help mastering progressively the complexity of the relationship. This confirms our experiences which show that the refinement ratio (see Table1) is around 20, meaning that a relationship, initially seen as a whole, finally leads to a complex organization of about 20 sub-relationships.

From my knowledge, there are very few attempts to provide a refinement mechanism of the fitness relationship [Potts97b]. In goal driven approaches, it is known that goals can be used to capture the objectives of a system at various levels of abstraction and goal decomposition is traditionally used to relate high level goals to low level, operationalisable goals. These are leaves of the goal graph that point out to the required functionalities of the system. One can therefore see that goal decomposition does not support a top-down reasoning about the fitness relationship. Instead goal decomposition is a mechanism leading to the establishment of the fitness relationship as the link between a system functionality and a leave of the goal graph.

The possibility of refining a goal graph was one of our motivation for defining MAP. In the map approach it is possible to *refine a section* of a map at level *i* into an *entire map* at a lower level *i+1*. Therefore, a fitness relationship (captured in a section of the map) is refined as a complex graph of sections, each of them corresponding to sub-relationships between the business and the system. Therefore, what is refined by the refinement mechanism offered by maps is in fact the fitness relationship itself. We found this mechanism helpful to understanding the fitness relationship at different levels of detail. Table 1 provides data gathered from four projects in industry [Rolland01] [Rolland99] [Rolland04] [Rolland06]. The table shows that the top map has a limited number of goals and strategies. The table also reflects the fact that systematic section refinement could rapidly lead to a combinatorial explosion of the number of maps to document. There is therefore, a need to control the refinement. It was also found necessary to identify when the refinement is needed and when it is not. In the DIAC project for example, we achieved the latter through a consensus based process : each section was subject to a vote and the refinement was considered unnecessary when the fitness between the business requirement and the selected product was agreed upon by the stakeholders.

Table 1. Practice data

	Goals in top level map	Sections in top level map	Refinement levels	Total number of maps	Number of transactions or screens
PPC	2	6	3	37	200 transactions,
DIAC	3	14	3	36	2000 screens
SAP MM	2	11	2	14	About 50
SNCF	4	27	5	55	300 transactions

3. Issues in engineering the relationship

Whereas the three previous issues were dealing with understanding the fitness relationship I am now moving to the issues related to its production, particularly in an evolutionary perspective.

Issue 4 : Identifying change engineering classes

The position here is that there are different requirements change situations leading to specific ways to engineer the fitness relationship. My suggestion is to *classify* these change situations and change processes as a means to better understand the different engineering ways to produce the fitness relationship. At the moment, I would like to propose two change engineering classes :

- (a) *By construction* and,
- (b) *By matching*

Class (a) accepts the prevalent view of change as a move from the *As-Is* to the *To-Be* situation [Jackson95]. The change process focuses on the *construction* of the To-Be requirements of the Business Model (BM) and the System Functionality Model (SFM) while keeping a good fit between them (Fig.2).

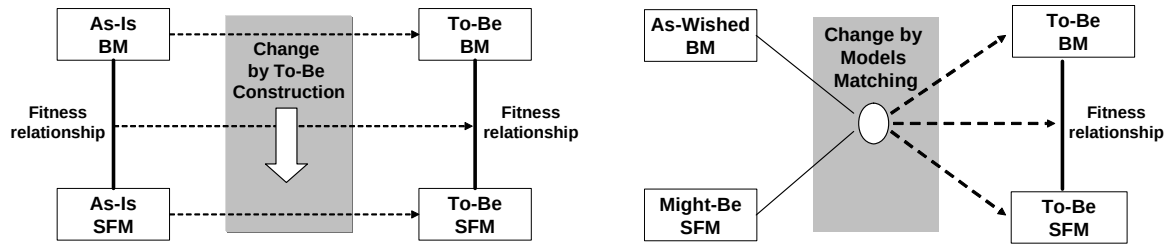

Fig.2 : Change engineering classes

However today, with an increase of the 'to-buy' policy over the 'to-do', the change is carried out through a different engineering processes in nature. In Class (b) requirements change processes are mainly *matching* processes (Fig.2). For example, in the case of an ERP system installation or the customisation of a product family, the requirements of the organisation are expressed in the *As-Wished BM*. The *Might-Be SFM* (Might-Be System Functionality Model) reflects the functional capability of the ERP or product family. The *To-Be BM* and its counterpart, the *To-Be SFM* result from a *model-match centred process* which searches for the best match between the organisational requirements (expressed in *As-Wished BM*) and the requirements that are fulfilled by the ERP or Product Family, (*Might-Be SFM*). The situation is similar in the case of COTS based system development except that the matching process has to deal with several *Might-Be* SFM to be composed to construct the To-Be SFM.

Issue 5 : Specifying change requirements

In order to carry out the change it can be consider necessary to specify *change requirements*, i.e. requirements for change. This is my own position that is far to be adopted by the majority. An analysis of the literature seems to show that there are two way-of-doing:

(a) *implicitly* and,

(b) explicitly

What I mean by *implicitly* is that change requirements are not formulated per se. The focus is (i) on the formulation of the To-Be requirements or (ii) on a new release of the requirements document. Many methodologies such as Merise, i*, Kaos or the RUP do not require an explicit description of changes of current models. They help to construct new ones focussing on the To-Be situation. Type (i) reflects an had-hoc practice inherited from the way legacy systems have been maintained over time. In (ii) type of approaches the concern on the change transition leads to keep track of requirements that have been removed, added, changed using a configuration mechanism of requirements documents.

I am in favour of an *explicit* specification of change requirements. Reflection from experiences led us to identify two common underlying strategies. On one side of the spectrum we found that *similarities* between pairs of models are useful to support the matching process whereas at the other end of the spectrum focusing on differences or *gaps* is the most relevant technique. For example, customising and component assembly processes call for a similarity based technique whereas a gap measurement technique is more suitable for processes of the type ' adaptation from a baseline product'. Similarity modeling and gap modeling are not easy tasks if they are not achieved in a systematic way. The challenge is to maximize the knowledge gained, while limiting the amount of effort needed to gain it.

Different gap and similarity based languages have been defined in different domains and could be adapted to the fitness context; for example, similarities formulae to reuse source code [Prechelt00], or find resemblances among object oriented models [Sarireta97] or UML models[Blok98]; languages for expressing requirements of database schema evolution [Delgado03], [Lauteman97], and [Banerjee87] or workflow models evolution [Casati96], [Kradolfer00], or [Reichert03]. We proposed in [Rolland04] a generic typology of gap operators that can be specialised for any modeling formalism. Once specialised for the MAP formalism, our language can be used to express how gaols and strategies shared by the business and the system shall change, e.g. a goal can change of name, a strategy can be replaced by another one, sections can be merged or the other round split, etc. Combined with maps of the current situation, change requirements expressed as gaps can be used to automatically control the consistency of the requirements actually released for the future system under the form of goal/strategy maps.

Issue 6 : Propagating change requirements

The question raised through issue 6 is that of propagation of change requirements taking into account that several entities have to evolve at the same time. Fig. 3 proposes four classes, namely *independence*, *interdependence*, *dependence*, and *double dependence*. Each class reflects a different ordering to handle the co-evolution of involved entities.

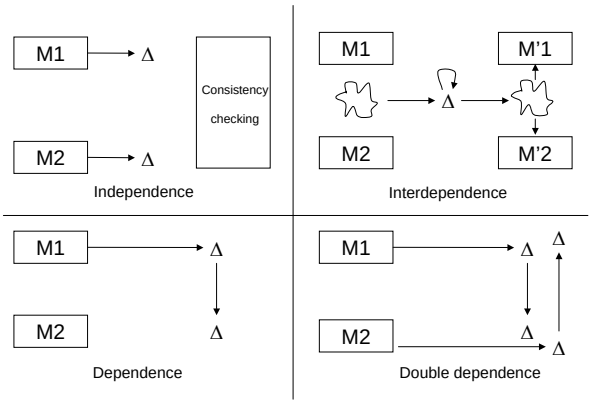

Fig.3: Change propagation classes

Co-evolution is engineered *independently* (left upper corner of Fig. 3), when there is no dependency between the change engineering processes of each evolving entity. Independent co-evolution implies the need for retrospect check of alignment.

In a *dependent* approach to propagation of change requirements, the change requirements of one entity are inferred from the change requirements elicited for the co-evolving entity. Changing the software system as a consequence of organisational changes is a typical example of this class. Similarly, projects portfolio mangers select often master projects which imply changes on other dependent projects. Rule based dependence as in [Krishna04] is a way to maintain alignment.

There is a *double-dependence* when each co-evolving entity can play the role of master in the propagation of change requirements. For example, [Kardasis98] proposes rules to evaluate the impact of change requirements specified at the business level on the system-level and vice-versa. A double dependence approach can be considered as the combination of two one-way dependence approaches.

An *interdependent* approach is more balanced. Each change requirement specifies how all co-evolving entities shall evolve simultaneously. Then, propagation is performed with a single collection of change requirements. The gap approach that we developed using maps as a means to represent both business and system belongs to this category. Change requirements are expressed as gaps and are propagated simultaneously on process and system models.

4. References

[Alves02] C. Alves, A. Finkelstein. *Challenges in COTS-Decision Making: A Goal-Driven Requirements Engineering Perspective.* In Procs of Workshop on Software Engineering Decision Support, in conjunction with SEKE'02. Ischia, Italy, July 2002.

[Arsanjani01] A. Arsanjani, J. Alpigini. *Using Grammar-oriented Object Design to Seamlessly Map Business Models to Component-based Software Architectures*. In Procs of the International Symposium of Modelling and Simulation, May 16-18, Pittsburgh, PA, USA, pp 186-191, 2001.

[Banerjee87] J. Banerjee, W. Kim, H.J. Kim and H.F. Korth. *Semantics and Implementation of Schema Evolution in OODB.* In Procs of ACM SIGMOD Conference on Management of Data, ACM, pp. 311-322, 1987.

[Blok98] M.C. Blok and J.L. Cybulski. *Reusing UML Specifications in a Constrained Application Domain*. In Procs of the 5th Asia Pacific Software Engineering Conference pp. 196-202. Dec. 2-4, 1998.

[Bodhuin04] T. Bodhuin, R. Esposito, C. Pacelli and M. Tortorella. *Impact Analysis for Supporting the Co-Evolution of Business Processes and Supporting Software Systems*. In Procs of BPMDS'04, Riga, Latvia, 2004.

[Brancheau96] J.C. Brancheau, B. Janz, J.C. Wetherbe. *Key issues in ISs management*. 1994-95 SIM Delphi results. MIS Quarterly, Vol. 20, n°2, 225-242, 1996.

[Casati96] F. Casati, S. Ceri, B. Pernici and G. Pozzi. *Workflow Evolution.* In Procs of the Conference On Conceptual Modeling (ER'96), Cottbus, Germany, pp. 438-455, 1996.

[Chan02] Y.E. Chan. *Why haven't we mastered alignment? The importance of the informal organization structure.* MIS Quarterly Executive, 1(2), 97-112, 2002.

[Chan97] Y.E. Chan, S.L. Huff, D.G. Copeland, D.W. Barclay. *Business strategic orientation, ISs strategic orientation and strategic alignment*. ISs Research, 8, 2, 125-150, 1997.

[Clarke99] S. Clarke, W. Harrison, H. Ossher and P. Tarr. *Subject-Oriented Design: Towards Improved Alignment of Requirements, Design and Code.* In Procs of Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), 1999.

[Croteau01] A.M. Croteau, F. Bergeron. *An information technology trilogy: business strategy, technological deployment and organizational performance*. Journal of Strategic IS, Vol. 10. Elsevier, 77-99, 2001.

[de Landtsheer03] R. de Landtsheer, E. Letier, A. van Lamsweerde. *Deriving Tabular Event-Based Specifications from Goal-Oriented Requirements Models.* In Procs of RE'03, IEEE International Conference on Requirements Engineering, Montgomery Bay, USA, September 2003.

[Delgado03] C. Delgado, J. Samos and M. Torres. *Primitive Operations for Schema Evolution.* In ODMG Databases Proceedings of Object Oriented Information Systems, Springer-Verlag Lecture Notes in Computer Science, Vol. 2817, Geneva, Switzerland, pp. 226-237, 2003.

[Etien05a] A. Etien, C. Rolland. *A Process for Generating Fitness Measures.* In Procs of CAISE05, Springer Verlag, Portugal, 277 – 292, 2005.

[Etien05b] A. Etien and C. Rolland. *Measuring the fitness relationship.* Requirements Engineering Journal (REJ), No10, p184-197, 2005.

[Jackson95] M. Jackson. *Software Requirements and Specifications*. Addison-Wesley. 1995.

[Kardasis98] Kardasis, P. and Loucopoulos, P. "Aligning Legacy Information Systems to Business Processes", Proceedings of CAiSE*98, Pisa, Italy, 1998, pp. 25 - 40

[Kefi05] H. Kefi, M. Kalika. *Survey of Strategic Alignment Impacts on Organizational Performance in Int. European Companie.* Hawaii Int. Conf. on System Sciences, 2005.

[Kradolfer00] M. Kradolfer. *A Workflow Metamodel Supporting Dynamic, Reuse-based Model Evolution.* PhD thesis, Department of Information Technology, University of Zurich, Switzerland, 2000.

[Krishna04] A. Krishna, A.K. Ghose, S. Vilkomir. *Co-Evolution of Complementary Formal and Informal Requirements.* In Procs of International Workshop on Principles of Software Evolution (IWPSE'04), Kyoto, Japan, pp. 159-164, September 2004.

[Lauteman97] S. E. Lauteman. *Schema Versions in Object-Oriented Database Systems*. In Procs of the Fifth International Conference on Database Systems for Advanced Applications, Melbourne, Australia, April, 1997.

[Luftman04] J. Luftman, E.R. Maclean. *Key issues for IT executives*. MIS Quarterly Executive, 3, 89-104, 2004.

[Pohl94] K. Pohl, S. Jacobs. *Concurrent Engineering: Enabling Traceability and Mutual Understanding.* Journal of Concurrent Engineering Research and Application, Special Issue on Concurrent Engineering and Artificial Intelligence, Vol.2, No.4, 1994, pp. 279-290

[Potts97a] C. Potts. *Fitness for Use: The System Quality that Matters Most.* In Procs REFSQ'97: Third Int.Workshop on Requirements Engineering: Foundation for Software Quality. Barcelona, Spain: June 16-17, 1997.

[Potts97b] C. Potts, I. His. *Abstraction and Context in Requirements Engineering: A Synthesis of Goal Refinement and Ethnography*. Annals of Software Engineering, Vol. 3 pp. 23-61.1997.

[Prechelt00] L. Prechelt, G. Mahpohl, and M. Phlippsen. Jplag. *Finding Plagiarisms among a set of Programs*. Technical Report 2000-1, Universitat Karlsruhe, March 2000.

[Regev04] G. Regev, A. Wegmann. *Remaining Fit: On the Creation and Maintenance of Fit*. Proceedings of BPMDS'04, Riga, Latvia, 2004.

[Reich03] B. H. Reich, K.M. Nelson. *In Their Own Words: CIO Visions About the Future of In-House IT Organizations*. The DATA BASE for Advances in ISs, 34, 2003, 28-44.

[Reichert03] M. Reichert, S. Rinderle and P. Dadam. *A Formal Framework For Workflow Type And Instance Changes Under Correctness Constraints*. UIB-2003-01, April 2003.

[Rolland199] C. Rolland, G. Grosz, and R. Kla, *Experience with Goal-Scenario Coupling in Requirements Engineering*, Requirement Engineering (RE), Limerick, Ireland, June 1999.

[Rolland01] C. Rolland, N. Prakash. *Matching ERP System Functionality to Customer Requirements*. In Procs of the 5th IEEE International Symposium on Requirements Engineering, Toronto, Canada. August 27-31, 2001.

[Rolland04] C. Rolland, C. Salinesi, and A. Etien. *Eliciting Gaps in Requirements Change.* Requirement Engineering Journal, Vol. 9, 2004, pp1-15.

[Rolland06] C. Rolland, *Aligning Business and System Functionality Through Model Matching*, Systèmes d'Information et Management (SIM), Editions ESKA, 2006

[Sabherwal01] R. Sabherwal, Y.E. Chan. *Alignment between business and IS strategies: A study of prospectors, analyzers and defenders.* ISs Research, 12(1), 11-33, (2001).

[Sarireta97] A. Sarireta and J. Vaucher. *Similarity Measure in the Object Model*. In Procs of ECOOP.97, Jyvaskyala, Finland, 1997.

[Soffer04] P. Soffer. *Fit Measurement: How to Distinguish Between Fit and Misfit.* Note for BPMDS'04, Riga, Latvia, 2004.

[Sysml] http://www.sysml.org

[Tallon02] P.P. Tallon, K.L. Kraemer. *Executives' Perspectives on IT: Unraveling the Link between Business Strategy, Management Practices and IT Business Value.* ACIS2002, USA.

[Thevenet07] L.H. Thevenet, C. Salinesi. *Aligning IS to organisation's strategy : the Instal method.* Accepted for publication in CAISE'07, Trondheim, 2007.

[Watson97] R.T. Watson, G.G. Kelly, R.D. Galliers, J. Brancheau. *Key issues in ISs management: an Int. perspective*. Journal of Management ISs 13, 91-115, 1997.