Fitting Business Models to System Functionality Exploring the Fitness Relationship

C. Salinesi and Colette Rolland

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

Abstract. One of the major concerns in Requirements Engineering is to establish that the 'whys' of the system to be developed fit the 'whats' of the delivered system. The aim is to ensure a 'best fit' between organisation needs (whys) and system functionality (whats). However, systems, once developed, undergo changes and it is of prime importance that the changed need and the changed system functionality continue to preserve the 'best fit'. We explore the fitness relationship to reveal its nature and its engineering process. We identify major issues that must be addressed in this process to arrive at the best fit. We also consider the preservation of this relationship in the face of change and discuss some issues specific to this scenario. The results presented are founded in our experience in about a dozen industrial and research European projects.

1 Introduction

Requirements engineering is seen as a way of establishing a relationship between the "whys" and the "whats" of the system under development [Lamsweerde01][Yu01]. The latter deals with the system functionality whereas the former provides its rationale. The "whys" are captured in the *Business Model* (*BM*) whereas the "whats" are available in the *System Functionality Model* (*SFM*). When system development starts from scratch, then establishing the relationship between the "whys" and the "whats" is a means by which requirements engineering ensures that system functionality matches organizational requirements. We refer to the relationship between *BM* and *SFM* as the **fitness relationship** [Potts97]. When the system undergoes a change then, just as there is a fitness relationship between the *As-Is BM* and the *As-Is SFM* reflecting the current state, requirements engineering must ensure the fitness relationship between the *To-Be SFM* and the *To-Be BM* expressing the future state.

In this paper we consider some of the common issues involved in the twin problems of *establishing the fitness relationship* and *preserving it in the face of change*. We will use examples extracted from four European and industrial projects that we have been involved in [Rolland00a][Rolland00b][Rolland01][Rolland03]. These projects considered change management in different change contexts, and the main problem was that of ensuring a proper fit between the *BM* and the *SFM*.

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 647–664, 2003.

The broad framework used in this paper is in Fig. 1. It accepts the prevalent view of change as a move from the *As-Is* to the *To-Be* situation [Jackson95]. However, it departs from the traditional view in highlighting the fitness relationship itself and its engineering through the change process.

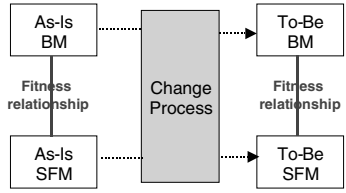

Fig. 1. The broad change framework

Thus, the framework raises two main points of interest: (a) *understanding* the fitness relationship, and (b) *engineering* the fitness relationship.
The former deals with characterizing, conceptualizing, and modeling, the fitness relationship whereas the latter deals with establishing and preserving it in the face of change. Issues relating to (a) above represent the static point of view whereas those of (b) above follow the dynamic viewpoint.

It has been recognized that there is a *conceptual mismatch* between the *SFM* and the *BM* levels [Arsajani01][Alves02]. It is considered necessary to minimize this mismatch. Since the fitness relationship deals with the system as well as the business viewpoints, *modelling 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. 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.

To sum up, it can be surmised that an understanding of the fitness relationship requires a position to be adopted on the following issues

Issue 1: Conceptual mismatch;
Issue 2: Modelling the relationship;
Issue 3: Refining the relationship;
Issue 4: Refinement Quality.

Moving to the engineering aspect, the framework suggests that there are two engineering contexts, with and without the change process. The former corresponds to the case of development from scratch where the fitness relationship has to be freshly established, and the latter which involves the production of the fitness relationship through change. This latter corresponds to the preservation of the fitness relationship. The process of establishing or preserving the fitness relationship necessarily involves the exploration of alternative ways by which elements of the *BM* can be related to one or several elements of the *SFM*. Thus, one common issue to both these contexts is the *exploration* and subsequent selection from a set of alternatives. Thus, we have the fifth issue as follows: **Issue 5**: Exploring alternatives.

Our experience in a wide range of change handling projects shows that change arises in a number of different ways. Each of these can be characterized by a

specialization of the framework of Fig. 1. For example, in the case of ERP system customizing, the *As-Is BM* is an *As-Wished* model expressing the business requirements for the *To-Be* situation. The *As-Is SFM* is the *Might-Be Model* expressing the set of functionalities that could be installed in the *To-Be SF*. In this case, the production of the fitness relationship is through a matching process which aim is to find the best fit between the ERP functionality and the wished *BM*.

The next issue in engineering the fitness relationship, namely **Issue 6**, is the *identification of the different engineering classes.* Our experience, suggests that there are four classes :

- Direct change propagation;
- Customization from product family;
- Adaptation of a baseline product; and
- Component assembly.

Each class has its own engineering process. Given the body of opinion which supports modeling of engineering processes, we believe that these four processes have to be modeled and additionally, suitable *guidance* mechanisms should be developed to direct/monitor their enactment.

Despite the diversity of processes, we found some common underlying strategies. The customisation and assembly processes ask for an identification of *similarities* between the old and the new situation whereas direct propagation and adaptation highlight the *differences* between the old and the new. This leads us to believe that there are two strategies underlying these four cases, namely gap driven and match driven strategies.

Thus, we have three issues

Issue 7: Modelling and guiding the engineering process; and

Issue 8: Engineering strategies

In the rest of this paper we consider these eight issues. The next section deals with the four issues that arise in understanding the fitness relationship. Section 3 of the paper considers the four issues of engineering the fitness relationship.

2 Understanding the Fitness Relationship

In this section we consider the four first issues dealing with the understanding and characterization of the fitness relationship.

Issue 1: Conceptual Mismatch

BMs and *SFMs* are typically expressed in different languages. Business models use concepts such as goal, process, actor and role whereas system functionality models deal with objects, operations, events and the like. The distance between these two sets of concepts is referred to by [Arsajani 01] as the "*conceptual mismatch*" between the business and software models. We experienced this issue in ERP installation projects, where we found that ERP experts and organisational stakeholders had difficulty to match each other requirements [Rolland01]. Indeed, the customisation process typically focuses on the ERP functionality and on its customisation. The functionality is expressed in low level details such as data to be maintained and operations to be carried out whereas organisations think in terms of their goals and objectives. This

results in a language mismatch between ERP experts and organisation stakeholders. This mismatch exposes the ERP system installation to the danger of failing to meet the requirements of organisations.

One way to obviate this issue is to leverage the functional 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 [Yu01][Lamsweerde01]. Our experience is based on the use of the *Map* representation system for a *uniform representation of business goals and system functionalities*. A detailed description of the notion of map can be found in [Rolland01]. A brief overview is as follows:

A map is a labelled directed graph (see Fig. 2) with *goals* as nodes and *strategies* as edges between goals. 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. Since, there can be multiple edges entering a node, the map is capable of representing the many strategies that can be used for achieving an goal.

A *goal* can be achieved by the performance of a process. Each map has two special goals, *Start* and *Stop*, to start and end the process respectively.

A *strategy* is an approach, a manner to achieve an goal. The strategy S_{ij} characterizes the flow from the source goal G_i to the target goal G_j and the way G_j can be achieved once G_i has been achieved.

A *section* is the key element of a map. It is 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.

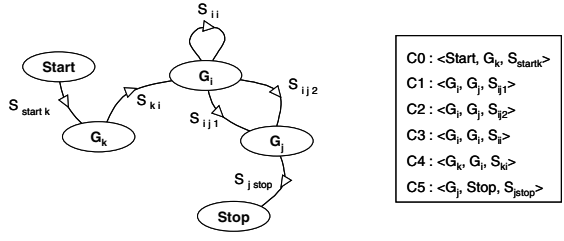

Fig. 2. A map

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 task can be performed 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.

As an example, the map of Fig. 2 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.

All the projects which serve as the basis for discussion in this paper used the map representation system as a means to unify the business view and the system functionality view.

Issue 2: Modelling the Relationship

Despite the fact that the fitness relationship is mentioned in many approaches, it is rarely a concept per se. For example, most of recent object oriented methodologies argue in favor of relating business models to software models but adopt a process tracking approach leading to a *loose indirect*, *coupling* by which a business case is related to a software component through a number of development process steps. The relationship between a use case and a component in Catalysis [D'Souza99] or the RUP [Jacobson99] are examples of such loose coupling.

In contrast goal driven approaches help in establishing a more *direct coupling* as a goal is operationalised in a system functionality. For example, in a goal driven requirements engineering approach such as KAOS [Lamsweerde01] an operationalisable goal is directly related to a system constraint expressed in terms of system objects, actions and events that will contribute to this goal achievement.

Our experience is that in the current prevalent context of rapid change a *direct and strictly defined coupling* is required. This shall facilitate the propagation of a business change onto a system functionality.

This coupling is achieved in the map formalism by *simply relating each section of a map to a system functionality.* Therefore, 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.

For example, the map in Fig. 3 shows how the SAP Material Management (MM) module can be abstracted into sections of a map [Rolland01]. Every section in the map represents both (i) a SAP's MM function, and (ii) the business goal that can be satisfied by using this function.

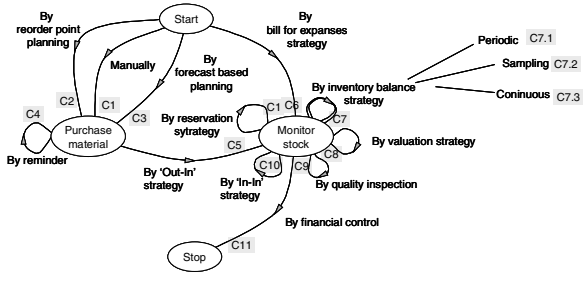

Fig. 3. Map describing how to *Satisfy material needs efficiently* with SAP's MM module

From the *business viewpoint*, material management deals with supplying materials in the right quantity, at the right place and time, and at the minimum cost. The map identifies through two goals that this involves two issues: to *Purchase material*, and to *Monitor stock*. It also make explicit the different manners by which each goal can be achieved. For example there are four strategies to *Purchase material*, namely *Manually*, *By forecast based planning*, *By reorder point planning* and *By reminder*.

From the *system viewpoint*, the map indicates which SAP function helps to achieve the *Material purchase* and *Monitor Stock* goals, and how. For example, the SAP MM module contains a "Create purchase order" function (or 'transaction' in SAP's terms). At the operational level, this function entails the identification of material requirements. The material requirements are defined by references to the needed materials, their vendors, their prices, the dates and plant at which they should be delivered, etc. At the business level, the issue is the one of purchasing material to satisfy material needs of the organization. The function contains variants depending of the purchase situation. These are referred to in the four sections C1, C2, C3, and C4, as documented in Table1. For each of the four sections, the table outlines the variant of the SAP function that is to be used.

Table 1. Documenting map sections

Code	Name	Description
C1	Purchase material	Create a purchase order based on a purchase requisition manually defined with information about the material, vendor, date, price, etc. If the information is correct the purchase order is created with a unique identification number.
C2	Purchase material based on reorder points	Automatically generate purchase requisitions any time a stock event that causes the stock of a given material to fit the reorder point criteria occurs. The purchase requisitions can then be transformed into purchase orders.
C3	Purchase material based on forecast	Automatically generate purchase requisitions at the dates defined in the forecast scheduling the purchases that shall be made for a given material. The forecast is computed based on former purchases of the material. Once generated, the purchase requisitions can be transformed into purchase orders.
C4	Purchase material by reminder	Automatically remind of a purchase order for which no delivery has been noticed within due date.

As illustrated in the SAP example above, our experience showed that the multi-thread topology of the map is useful to reasoning about alternative fitness relationships. The multi-thread (a) makes explicit the different business strategies to achieve a goal and (b) identifies the variants of the SF that can be selected depending on the situation at hand, thus highlighting the alternative (ORed) fitness relationships. We found that when eliciting the desired state (To-Be model) the multi-thread helps envisioning multiple business strategies and identifying the corresponding required SF. In a customizing process the multi-thread helps exhibiting the panel of business strategies embedded in the product family and their related software variants.

Issue 3: Refining the Relationship

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. Our belief is that such *refinement mechanism* is required for *handling the fitness relationship* in *a systematic, 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 Table2) is around 20, meaning that a relationship, initially seen as a whole, finally leads to a complex organization of about 20 sub-relationships.

From our 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 [Lamsweerde01] and goal decomposition ([Bubenko94] [Nurcan02] [Finkelstein02]) 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.

In the map approach we defined a refinement mechanism in order 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.

Let us exemplify this mechanism by refining the section C5, *Monitor stock by Out-In strategy* of the SAP map presented in Fig. 3. The C5 refined map is shown in Fig. 4. From the *business viewpoint*, this map explains how the *Monitor Stock by 'Out-In' strategy* is refined as a graph of goals and associated strategies. The map tells us that stock entries shall not be permitted unless they have been checked. This is reflected in Fig. 4 by the ordering of the two goals *Accept delivery* and *Enter goods in stock*. The map also shows that there are several ways to achieve each of these two goals. For example, there are four strategies to *Accept delivery*: the *Okay* strategy (when the delivery conforms the order) and three reconciliation strategies, namely the *Reconciliation by PO recovery*, the *Reconciliation of unit difference*, and the *Reconciliation of under/over delivery*. Each of these provides a way of accepting deliveries that don't match the order but are within specified tolerances, missing order reference, unit difference, under/over quantity, respectively. From the *system viewpoint*, this map explains how the *complex function C5* is made of other functions and in which way these functions co-operate to achieve collectively the C5 goal. The map shows that there are seven sub-functions C5.1 to C5.7 of the C5 function that shall co-operate as indicated by the multi-thread and multi-path topology of the M5 refined map.

Since refinement results in a map, it produces a multi-thread, multi-path structure at level *i+1*. As a result, for a given section at level *i*, not only (1) multiple threads describe *alternative sub-sections* at level *i+1*, but also (2) the multi-path structure introduces several different *combinations of sub-sections*. Therefore, section refinement is a more complex structure than a simple composition structure such as AND/OR goal decomposition Indeed, it provides at the same time (a) several alternative decompositions of the initial fitness relationship into its constituents, and (b) different alternatives to its constituents themselves. We found this mechanism useful in practice as a means to fine tune the selection of the adequate system sub-functions in a customizing process.

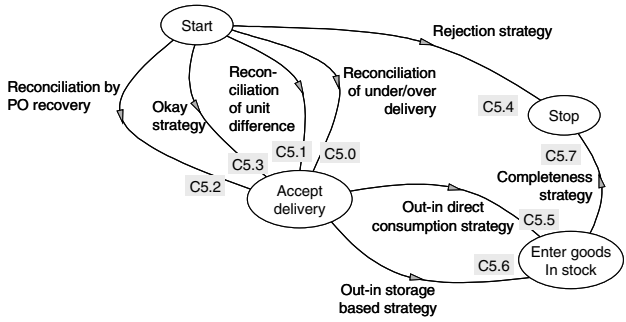

Fig. 4. Refined map M5 of section C5 of the SAP MM map shown in Fig. 3

As an example, let us consider two companies who have selected the *Out-In strategy* of the SAP map to *Monitor stock*. The refined map M5 (Fig.4) helps each of them to refine their selection in different ways. For example, while one company may want to *Accept delivery* by *reconciliation of over/under delivery*, the other may be driven by the *reconciliation by purchase order recovery* strategy. Similarly, one company may want to *Enter goods in stock* using the *Out-in direct consumption* strategy that allows one to consume goods even though they are not yet entered in stock, whereas the other company may select only the *Out in storage* strategy, i.e. systematically enter goods in stock before they are consumed. Therefore, while the two companies have the same fitness relationship at the level of abstraction of the MM map, the refinement mechanism allows us to differentiate the sub-relationships relevant for each company.

Issue 4: Refinement Quality

If refinement is necessary as a way to master complexity, it also generates its own difficulties. One is to control the *level of abstraction* in a given refinement. This has been found for example, in using refinement in use case driven approaches [Weidenhaupt98]. Cockburn reported [Cockburn00] that the application of his refinement mechanism (by which an action in a scenario can be seen as a goal attached to a new use case and associated scenarios) led to mix up of abstraction levels in the same scenario. It seems that several levels of abstraction are source of difficulty, in particular with respect to consistency of the entities belonging to the same level. This issue is related to the *black box/white box* principle [Weidenhaupt98] [BenAchour99] [Kamsties02]. According to this principle, it is useful to see an entity as a black box at a level of abstraction *i* and then as a white box at level *i+1*. When a system is seen as a black box, its internal properties are hidden and the emphasis is on the relationship between the system and other systems. When it is seen as a white box, the internal of the system is on the contrary apparent. The problem arises when the white box analysis shows that the content of the box covers different levels of abstraction.

We encountered this problem when applying the map refinement mechanism in projects, and we believe that complementing the *refinement mechanism* of the fitness relationship with *refinement quality assurance* is an issue.

In a project with the Renault company [Rolland03], we defined and experienced the use of a set of *refinement quality rules.* The aim was to ensure a unique level of abstraction in a given level of map refinement. In essence rules helped, given a map at a refinement level *i*: (a) to detect and move up sections into maps at level i-1; (b) to detect and move down sections into maps at level i+1 and, (c) to improve sections at the same level. We briefly illustrated (a) and (b) in Fig. 5. Rules help detecting the following in the initial map (m) :

- The goal *Evaluate risk* is a means to *Decide on the offer.* It is thus a sub-goal to *Decide on the offer*, and belongs to a lower level of abstraction. This led to the introduction of a map (c) that refines the section <*Gather offer data, Decide on the offer, by semi-automated risk evaluation*>; the goal *Evaluate the risk* was moved down.
- Map (b) as resulting of the above transformation can be abstracted into a single section of a higher level map (a). The section relates to the goal *Gain the customer*, and offers a strategy, namely *By contracting* which is an alternative to the two other strategies already identified : *By keeping customer loyalty*, and *By prospecting*.

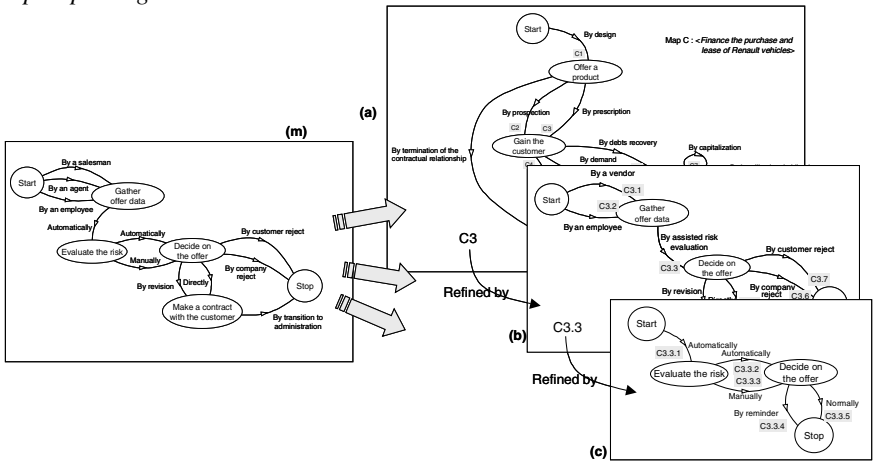

Fig. 5. Applying rules to control refinement quality

In order to reinforce the importance of the *refinement quality checking* issue, we present in Table 2 data gathered from three projects in industry [Rolland03] [Rolland01] [Rolland00b]. The table shows that a top-down approach was followed in the three cases, starting form one single map with 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 2. 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

3 Issues in Engineering the Relationship

Whereas the four previous issues were dealing with understanding the fitness relationship we are now moving to the issues related to its production.

Issue 5: Exploring Alternatives

This issue relates to the exploration of the different ways to link a business goal and a system functionality; in other words to investigate the *alternative fitness relationships* for the problem at hand. In a change perspective this is crucial for the envisionment of the future system.

Most of the goal based RE approaches recognize the usefulness of goals in exploring alternative designs [Anton98] [Yu01] [Lamsweerde01] [BenAchour99] [Paech02]. This is generally achieved using AND/OR refinement where "alternative goal refinements [expressed with OR links] allow alternative system proposals to be explored." [Letier02]. Our experience with goal analysis [Rolland98] is that providing automated support to alternative goal generation is useful because manual search for alternative is far to be exhaustive [Rolland99].

However, if alternative goals help reasoning about alternative system functionalities to achieve the parent goal, the issue of exploring fitness relationship alternatives raises the question of reasoning about alternative combinations of functionalities across the entire AND/OR goal graph.

We found that maps, as a means for describing alternative complex assemblies of functionalities, can help in this exploration and in the discovery of the ones that best fit the business goals. The multi-thread topology of maps corresponds to OR structures in a goal graph. In addition, the multipath map topology helps reasoning and evaluating alternative assemblies of functionalities. Such assemblies give rise to a *payoff analysis*. The result is the selection of sections that show the combination of the functionalities required.

For example, the map shown in Fig. 6 identifies seven different functionalities for the management of electricity supply in a utility company. Each functionality is identified by a section in the map. The <*Start, Sell electricity, with credit strategy*> section identifies C4 for selling electricity in a conventional way, which provides IT support to manage the process chain of conventional meter reading, electricity consumption billing and payment collection with seven sub-component as map M4 shows it.

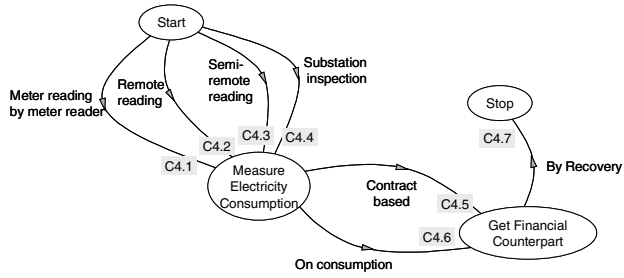

Fig. 6. *Manage electricity supply* and *Sell electricity by credit strategy* maps

Once C4 has been selected, one has to decide on how electricity should be measured and how the financial counterpart should be obtained. Each sub-component selection has however a payoff that can be analysed in the view of its combination to another component. The pay-off analysis for C4 sub-components is summarized in the Table3 below.

Table 3. Pay-off summary for the selection of C4 sub-components

		Get financial counterpart	
		Contract based	**On consumption**
Measure electricity consumption	**Meter reading by meter reader**	Can be envisaged at sustainable cost if visits are achieved at a low frequency (e.g. once or twice a year)	Excluded because too difficult to organise all visits at the required pace.
	Remote reading	Cost effective combination that can be done in real time. However, remote reading is not completely secure. A complimentary check of electricity measurement is thus needed, e.g. by meter reader, or by substation inspection.	
	Semi-remote reading	Cost effectiveness is a linear function of the number of contracts per cluster of semi-remote reader.	Very costly if the number of customers paying on consumption, per cluster of remote reader is low
	Substation inspection	Only possible if the connected meter readers relate to single contract. Otherwise, calls for individual reading.	Cost effective way to handle the verification of consumers invoiced by remote reading clustered on the same substation.

Let us consider the case where it is necessary to get financial counterparts both contract based and on consumption. The table shows that remote readings are a cost effective way to handle electricity measurement in both cases. Indeed, it is real time and therefore adapted to payment on consumption. Besides, the cost of installing remote readers can be included in the contract prices and recovered in the long term. However, the payoff table also says that remote reading, as it is automated, is not fully reliable and should be double-checked, e.g. by using subsection inspection. New opportunities emerge from further analysis. For example, remote reading makes it possible to analyse consumption in real time; new types of contracts can also be proposed to automatically adapt electricity production to consumption. This would be useful when electricity provision should be reliable, e.g. for refrigeration warehouses, hospitals, telephony service, etc.

Issue 6: Identifying Engineering Classes

As already mentioned in introduction, our experience showed that there are different change situations leading to specific ways to engineer the fitness relationship. Our 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. We propose four engineering classes

- Direct change propagation;
- Customization from product family;
- Adaptation of a baseline product; and
- Component assembly.

Fig. 7 is the customized version of the generic change framework presented in Fig.1. It corresponds to the case of *direct change propagation*. In this situation, the change is led by the move from the *As-Is BM* to the *To-Be BM*. The relationship between the *As-Is BM* and the *As-Is SFM* is used to propagate the business changes onto system functionality changes and to produce the *To-Be SFM* fitting the *To-Be BM*.

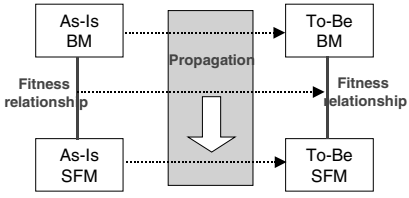

Fig. 7. Direct change propagation

This first engineering class corresponds to the traditional evolutionary change of an 'in house' system that is driven by organisational change. However today, with an increase of the 'to-buy' policy over the 'to-do', the fitness relationship is established through rather more complex engineering processes shown in Figures 8, 9, and 11 respectively.

Fig. 8 displays the case of *customising a product family*. Here, 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 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 what is provided by the Product Family, (*Might-Be SFM*).

The third engineering class shown in Fig. 9 corresponds to a case of *system adaptation*. Such an adaptation is caused by changes in the organizational context in which a legacy software system is now to operate. This typically occurs because of mergers/take-overs, globalisation, standardisation of practices across branches of a company etc. Several legacy software systems are already running when such events occur. In this context, it is out of question to develop a new system from scratch. However, it is possible to integrate the legacy systems or to select one of these for adaptation and uniform deployment across the organization. *The Is-Baseline FM* (Is-Baseline Functionality Model) models the functionality of the selected system

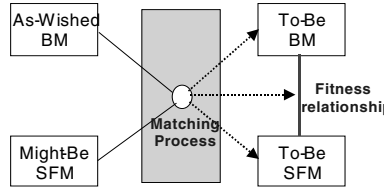

Fig. 8. Customization from product family

for uniform deployment across the organization. The *As-Wished BM* expresses the requirements of the organization who wants to adapt its requirements to the new situation at hand. The *To-Be BM* and its counterpart the *To-Be SFM* result from a gap *centred process* that focuses on eliciting the gaps between models. Indeed, eliciting the differences (or gaps) between the current baseline-functionality model and its future version seems to be the most efficient way to perform the change.

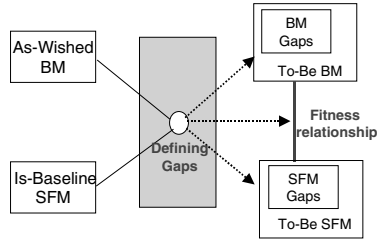

Fig. 9. Adaptation of a baseline product

Finally, Fig. 10 displays the case of *COTS based system development*. Here the process of establishing the fitness relationship is centred on the retrieval and assembly of system component matching organizational requirements. The *Might-Be SFMs* model the different available COTS components. The *As-Wished BM* reflects the organizational needs. The *To-Be BM* and its counterpart the *To-Be SFM* result of a process where component matching the organization needs are retrieved and then assembled together.

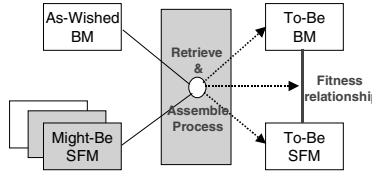

Fig. 10. Component assembly

Issue 7: Modelling and Guiding the Engineering Process

The importance of having quality engineering processes is recognized in the RE, SE and IS communities. This is also true for most recent challenges such as the ones raised by COTS selection and assembly [Maiden98] [Brons00] or [Boehm02]. In this

area, modelling processes is still necessary to document, guide, and improve them. For example, [Maiden02] underlines that "there is little guidance available on how to acquire requirements for COTS software, how to select COTS software compliant with these requirements, or how to interleave requirements acquisition and COTS software selection to provide the most effective process guidance". We identified the same needs in each of the projects that we conducted for this class of engineering problem.

Our experience in these projects is that people have specific expectations and requirements about these process models. First, they are facing an issue and have a goal in mind and would like process models to let them easily situate both and to suggest different alternative paths to achieve the goal and solve the issue. Second, they want freedom and flexibility in their ways of working; one single imposed way-of-working is not acceptable. They expect to learn about the different ways by which each of their goals can be achieved and each issue can be solved. Third, they want advice on how to choose between the different alternative solutions that shall be proposed to solve a given issue.

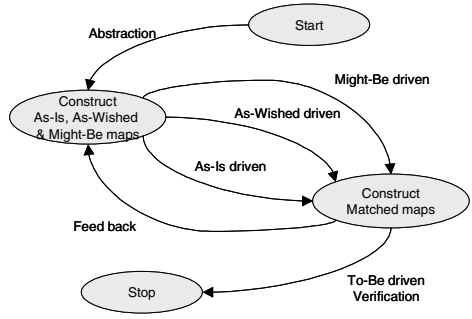

Fig. 11. The ERP customisation requirements process model

Fig. 11 shows a process model that was developed for an ERP customisation project. As the figure shows, the process model was developed under the form of a map. Indeed, our various experiences [Assar00], [Ralyte01], [Benjamen99], [Tawbi01] showed us that maps are useful to specify both methodological goals and the various strategies to achieve those. Besides, guidelines can be specified in maps to guide the selection of methodological goals, as well as to guide strategy selection, situation identification, and section achievement. For example, the process model identifies that ERP customisation involves a matching between the *BM* and *SFM* that can be achieved in three different ways:

1. If the context is that of a well-defined business requirements to which the system should fit, and in-house development is not a problem, then the As-Wished driven matching strategy can be used.
2. If on the contrary, the system is less likely to change than the business (e.g. because customising the system has become too expensive [Rolland01]), or if the system customisation is an opportunity to change the business (e.g. because it allows to generalise its associated best practice in the business) then, the matching process should be driven by the system. This is what the Might-Be driven strategy proposes.

If it is particularly important to preserve the functionality provided by the existing system in the To-Be SFM, then an As-Is driven matching is required. We encountered such functional non regression requirements when we studied the introduction of software components for selling electricity in the PPC company at the occasion of European electricity market deregulation [ISE01].

Issue 8: Engineering Strategies

Despite the diversity of processes dealing with each of the four classes of engineering problems, our 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 (Fig.7) and component assembly processes (Fig.10) call for a matching technique whereas a gap measurement technique is more suitable for processes of the type 'adaptation from a product line' (Fig.9).

Similarity modelling and gap modelling 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. For example, we developed *Goal semantic similarity* metrics and *Goal structure similarity* metrics to automate the measurement of similarity between maps [Ralyte01]. This is typically an issue that is met in the component assembly class of engineering problem. Indeed, looking in a collection of components (specified with Might-Be SFMs) for the ones that best fit a given business (specified with As-Wished BM) necessitates a systematic measurement of similarity between each map of the collection of Might-Be SFMs and the As-Wished BM.

We also experienced in industry the importance of modelling gaps in a baseline product adaptation project. In the DIAC project [Rolland03], it was not desirable to specify the future system functionalities as would be done if they were developed from scratch. Indeed, most of them already existed in the baseline system. Rather, the project owners wanted a documentation of the transformations that should be made on the baseline system to get the future one. These were specified under the form of a collection of gaps expressed as operations to perform on the Is-Baseline SFM maps to obtain the To-Be SFM maps. In the end, 98 of the gaps identified were specified in the requirements documentation. Fig. 12 presents some of these by organising them in a table and illustrating the resulting To-Be map. For each goal, strategy and section involved in a gap, the operation achieved to define the To-Be map is quoted in a separate line (operations are named in the left column).

The table shows for example that the baseline system offered the opportunity to shift the focus from a traditional contract oriented way of dealing with customers to a more customer centric one. This is specified by the operation of replacement of the goal *Define an offer* by the goal *Prepare a contract*, and by adding of the goal *Collect a demand*.

It is very common that a baseline system also brings new technologies with it. For this reason, the business experts involved in the project required new system functions fully exploiting the Internet technology. The introduction of these new functions into the system was specified by adding the strategies *By a customer via internet* and *By pull* in the To-Be map.

Operator	Goal	Strategy	Section
Rename	*Make a contract with the customer* as *Enter into a contract with the customer*		
Add	*Collect a request*	*By a customer via Internet* *By pull* *By modification request* *By customer retraction*	*<Collect a request, Prepare a contract, By assisted risk evaluation>*
Remove		*By an Employee*	
Split		*By company reject* into *By company reject* and *Bu deadline expiration*	
Change Origin		*By a vendor* target intention to *Collect a request*	
Replace	*Define an offer* by *Prepare contract*		

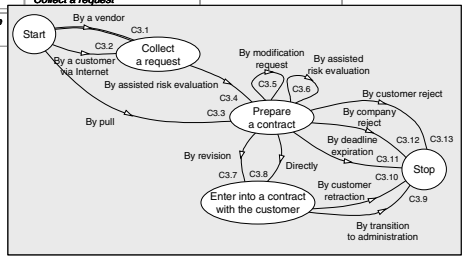

Fig. 12. Example of gaps and resulting map

The table emphasizes other gaps, such as renaming goals, splitting strategies, introducing sections, etc. Each of them is the result of a discussion about the fitness of the baseline system with the business vision. The gaps resulting from this discussion show how the Is-Baseline SFM (and sometimes the As-Wished BM) was to be transformed to get the To-Be SFM. The resulting map specifies a To-Be SFM that fits to the To-Be BM.

4 Conclusion

Our experience of Requirements Engineering and Information System Engineering stands in a number of industrial projects and European research projects. The common point we found in all these projects was the prime importance of establishing and preserving the fitness relationship between businesses and systems. A number of issues directly derive from this point. However, the prevalent view is that of a system and a business that change in a concurrent way. We depart from this traditional view by proposing to specify the fitness relationship itself and to engineer it through the change process. This entails a number of issues that were reported and discussed in this paper. Each issue was discussed individually with respect to the state of the art literature and with references to our own experiences. We believe more references should be introduced and we plan to do so in order to (i) validate/invalidate our belief that these are important issues that should be dealt with, (ii) help improve the classification of engineering processes, (iii) help us identify and/or confirm the common strategies that are used in these processes and finally, (iv) validate these strategies by comparison with the other existing ones.

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.

[Anton98] A.I. Anton C. Potts. *The Use of Goals to Surface Requirements for Evolving Systems*. In Procs IEEE International Conference on Software Engineering, IEEE Computer Soctety. 1998

[Arsanjani 01] 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, 2001, Pittsburgh, PA, USA, pp 186–191.

[Assar00] S. Assar, C. Ben Achour, S. Si Said. *Un modèle pour la Spécification de Processus d'Analyse des Systèmes d'Information*. Proc. 18ème Congrès INFORSID , Lyon France, May 2000.

[BenAchour99] C. Ben Achour. *Extraction des Besoins par Analyse de Scénarios Textuels*. Phd Thesis, Univ. Paris 6 – Pierre et Marie Curie. 1999.

[BenJamen99] A. Benjamen. *Une approche multi-démarches pour la modélisation des démarches méthodologiques*. Phd thesis. Université de Paris I – Panthéon Sorbonne. 1999.

[Bubenko94] J. Bubenko. *Enterprise Modelling*. In Ingenierie de Systèmes d'Information, Vol 2, Num. 6, pp. 657–678.1994.

[Cockburn00] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley. 2000.

[D'Souza99] D. F. D'Souza, A. C. Wills. *Objects, Components, and Frameworks with UML The Catalysis Approach.* Addison-Wesley, Object Technology Series. 2001.

[Finkelstein02] D. Bush ., A. Finkelstein. *Requirements Elicitation for Complex Safety Related Systems. In Procs London Communication Symposium. London, UK, Sept. 2002*

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

[Jacobson99] I. Jacobson, G. Booch, J. Rumbaugh. *Unified Software Development Process.* Addison-Wesley, Object Technology Series. 1999.

[Kamsties 02] E. Kamsties, A. von Knethen, R. Reussner. *A Controlled Experiment on the Understandability of Different Requirements Specifications Styles*. In Procs of REFSQ'02, Essen, Germany, 2002.

[Kawalek 99] P. Kawalek, P. Kueng. *The Goal in the Organization*. In Procs of BITWorld'99, Cape Town, South Africa, July 1999.

[Lamsweerde01] A. van Lamsweerde. *Goal-Oriented Requirements Engineering: A Guided Tour.* Invited Paper for RE'01 – 5th IEEE International Symposium on Requirements Engineering, Toronto, August, 2001, pp. 249–263

[Letier02] E. Letier, A. van Lamsweerde. *Agent-Based Tactics for Goal-Oriented Requirements Elaboration. In Procs* de ICSE'02 – 24th International Conference on Software Engineering, ACM Press. May 2002.

[Nurcan02] S. Nurcan, C. Rolland. *A multi-method for defining the organizational change*. To appear in Information & Software Technology, 2002.

[Paech02] B. Paech, A.H. Dutoit, D. Kerkow, A. von Knethen. *Functional Requirements, non-Functional Requirements, and Architecture Should not Be Separated.* In Procs. of REFSQ'02. International Workshop on Requirements Engineering : Fundation for Software Quality. Essen, Germany, Sept 2002.

[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.

[Ralyte01] J. Ralyté, C. Rolland. *An assembly process model for method engineering*. In Procs of CaiSE'01. Lecture Notes in Coputer Sciences. 2001

[Rolland00a] C. Rolland, N Prakash. Bridging the Gap Between Organisational Needs and ERP functionality. Requirtements Engineering Journal, Springer Verlag. 5:180.193. 2000.

[Rolland00b] C. Rolland. Intention Driven Component Reuse. Information Systems Engineering (S. Brinkkemper, E. Lindencrona, A. Solvberg, eds). Springer, pp. 197, 208. 2000.

[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.

[Rolland03] C. Rolland, C. Salinesi. *Eliciting Gaps in Requirements Change : An Industrial Experience*. To appear in Requirements Engineering Journal. 2003.

[Rolland98] C. Rolland, C. Souveyet, C. Ben Achour. *Guiding Goal Modelling Using Scenarios*. IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol 24, No 12, p. 1055–1071. December 1998.

[Rolland99] C. Rolland, G. Grosz, R. Kla. *Experience with Goal-Scenario Coupling in Requirements Engineering. In Procs* of the Fourth International Symposium on Requirements Engineering, RE'99. Limerick, Ireland, June 1999.

[Tawbi01] M. Tawbi. *CREWS-L'Ecritoire: un Guidage Outillé du Processus d'Ingénierie des Besoins*. PhdThesis, Université de Paris I – Panthéon Sorbonne. 2001.

[Weidenhaupt98] K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer, CREWS Team. *Scenario Usage in System Development : a Report on Current Practice*.Proceedings of ICRE'98, 3rd International Conference on Requirements Engineering, Colorado Springs USA. 6–10 April 1998.

[Yu01] E. Yu. "Agent Orientation as a Modelling Paradigm". *Wirtschaftsinformatik*. 43(2) April 2001. pp. 123–132.