A Knowledge Model for Business Re-engineering Methods and Tools

Stan Jarzabek and Tok Wang Ling
Department of Information Systems and Computer Science
National University of Singapore

Abstract

What do we need to know about the business in order to understand and, eventually, to improve business operations? Many business modeling methods have been described in literature and applied in business re-engineering projects. We feel that current business modeling methods do not have a precise enough model of the underlying business knowledge. A model should be comprehensive enough to allow for a systematic study and precise formulation of re-engineering methods. It should also provide a conceptual framework for designing tools to support business re-engineering projects. We identified information requirements for business re-engineering based on the commonly used business re-engineering methods and case studies published in the literature. We formalized these requirements within the *business knowledge model* that is described in this paper. The business knowledge models vary from company to company and from one business re-engineering project to another. Therefore, we build a generic model first and then we customize the generic model to needs of a given company and to a business re-engineering project in hand. We build the core of a tool environment for business understanding and re-engineering around the generic business model. The business knowledge model defines conceptual schema for the business knowledge to be acquired and stored in a tool repository. We derive the physical schema for tool repository and generate customized tools from the customized business knowledge model specifications. Tools built around our model can support business knowledge acquisition, business process modeling, performance/quality analysis and analysis of alternative business process re-engineering solutions. We use frames to model business features such as a customer, business process or event. Frames are organized into an inheritance network. Frame slots describe properties of features in the mixture of formal and informal specifications. In the paper, we describe the generic business knowledge model, explain how we use the model and discuss user-level views of the model.

1. Introduction

Operational transformation of business is a hot topic on the agenda of corporate management. Trends such as business re-engineering, business process re-engineering, business redesign, process improvement and process innovation all emphasize the need for transformation of business operations in order to achieve improvements in productivity and quality. Such improvements are required for companies to remain competitive in the market. The scale of required changes varies from company to company and ranges from incremental improvement scenarios [4,8] to radical rethinking of the way business operates [6]. Operational changes are oriented on the customer

and may involve, among others, simplification of the work flow, decentralization of responsibilities, standardization of procedures, reduction of redundant procedures and automation. Better use of information systems and technology often makes such changes possible [4,5,6,8,16]. In this paper, we are concerned with issues of business understanding for effective transformation of business operations, independently of the approach used to achieve the business transformation. We use the term *business re-engineering* (which seems to be the most common term) to mean all the types of radical and incremental business transformations described in the literature and practiced in industry [4,5,6,7,8,15].

We identified information requirements for business re-engineering based on the business re-engineering methods and case studies described in the literature. We formalized these requirements within the business knowledge model that is described in this paper. We feel that such a model is needed to allow for a systematic study and precise formulation of re-engineering methods. Our model also provides a conceptual framework for designing tools to support for business re-engineering projects. Tools that we are talking about support business re-engineering analysts in business knowledge acquisition, in business process modeling, in analysis of business data and in formulating alternative business re-engineering solutions. Fig. 1 positions the business knowledge model in the context of business re-engineering and the follow-up software development efforts.

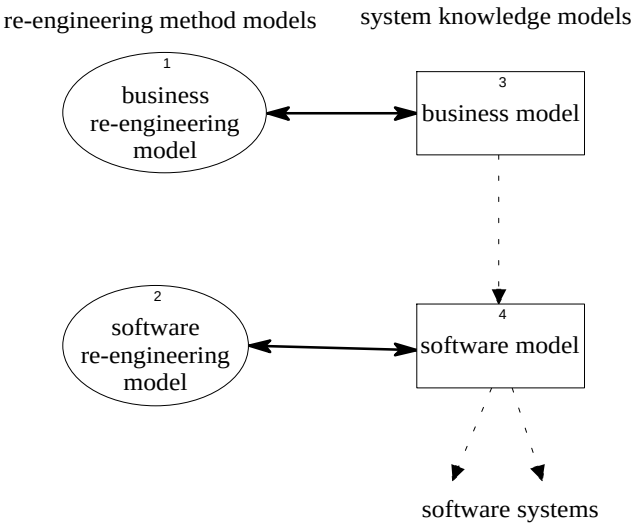

Fig. 1. Business and software models

System knowledge models (in rectangular boxes) describe business and software systems in the form suitable for re-engineering. These models provide the conceptual schema for the information that must be collected about the current state of business and software systems in order to proceed with a re-engineering project. Re-engineered business and software systems are also ex-

pressed within knowledge models. Dotted arrows in Fig. 1 depict mappings between models. Method models (in ovals) define re-engineering methods (i.e., how to re-engineer business and software systems). These models refer to business re-engineering methods (such as simplification, automation, etc.) and techniques (such as interviews with company staff, groupware, selecting candidate processes for re-engineering, modeling current and future processes and impact analysis). The business knowledge model serves as a reference model for business re-engineering method model. Method models use, create and update information stored in corresponding knowledge models (solid arrows). The purpose of models of Fig. 1 is to facilitate precise definition of detailed re-engineering methodologies and to aid in designing tools to support these methodologies. In [9], we described a framework called strategic re-engineering for addressing business and software re-engineering issues in an integrated way. In this paper, we shall concentrate on the contents and representation for a business knowledge model (box 3 in Fig. 1).

We model business features (such as a customer, business process or event) by frames [1]. Frames are organized into an inheritance network. Frame slots describe properties of features in the mixture of formal and informal specifications. A business knowledge model is bound to be complex. Therefore, business analysts are not supposed to work directly with the model. Tools are needed to provide an easy interface to the model and to enable the model to play a role of an active and useful assistant in business re-engineering projects. The business model defines an internal mechanism for tools that we build around the model. The role of tools is to provide an easy, intuitive and diagram-based access to the business knowledge.

The premise of our approach is that business process re-engineering is a creative thinking process (like software or engineering design) and as such cannot be fully automated. But tools can assist business analysts in business re-engineering, just like CASE tools assist software engineers in software design. Many business process re-engineering planning techniques (interviews, business modeling workshops, performance analysis, etc.) can then be effectively supported by tools. Our business knowledge model is semi-formal and reflects this human-centered nature of the business re-engineering process.

In the remaining part of the paper, we describe the generic business knowledge model, explain how we use the model, discuss some of the user-level views of the model and comment on tools that we build around the model. In the accompanying paper [10], we describe in detail a model-based design of a tool environment for business re-engineering.

2. Relation to other work

Many sources discuss motivation for business re-engineering, describe detail business process re-engineering methods and discuss tools that can support business process re-engineering activities [4,6,7,8,15]. CASE-like tools exist to help analysts in flow charting business processes as well as in many types of business data analysis [15]. Some tools allow analysts to simulate and do performance analysis of business processes [7,15]. This is possible due to a well defined underlying model (usually based on a variant of Petri nets). Yu and Mylopoulos [22] show that by capturing goals, rules and methods we can formally reason about implications of proposed business process changes.

We agree with authors of simulation and reasoning tools that a possibly precise internal information model is the first necessary step towards building successful tools. We also think that to effectively support business analysts, the underlying model should comprehensively cover many aspects of business structure and dynamics. In addition to executability, the model should record many kinds of dependencies between business entities that have to do with understanding of business operations. We hope the model described in this paper is a small step towards achieving this end. Model-based approaches have been applied before in domains such as CASE tool design, software design, software reuse, software maintenance and software re-engineering. In each of these domains, a proper model defines the structure and some semantics of entities that must be understood, analyzed and manipulated in the context of a given domain. The role of a well-defined model in business re-engineering is similar to roles of models in other domains.

3. The scope of a generic business knowledge model

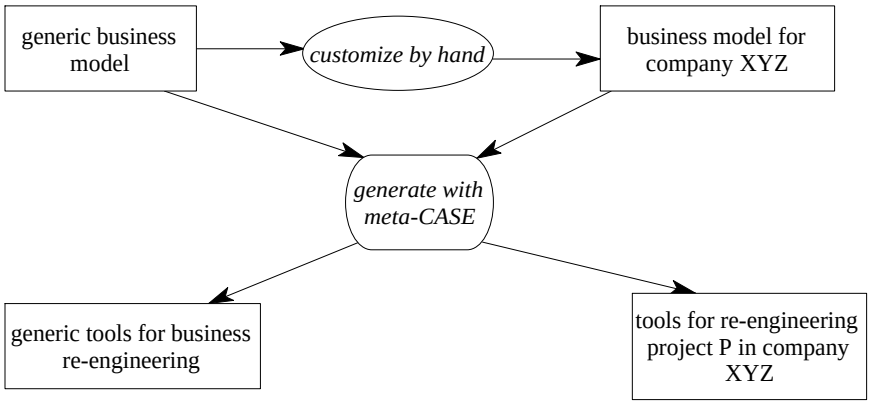

Fig. 2. Business model and tool customization

The scope and detailed schema of a business knowledge model depend on the business re-engineering goals, on re-engineering methods to be used and on specifics of a company to be re-

engineered. However, we can try to characterize a common core of business knowledge that, after customizations, can be reused across a range of companies and re-engineering projects (Fig. 2). In the next section, we shall propose such a *generic business knowledge model*.

We assume an approach to business re-engineering in which the assessment of current business processes is done prior to the design of improved business processes. (Some other approaches advocate design of new business processes from scratch, based on company goals, to allow for radical changes without any limitations that may be implied by the current way of conducting business processes.) The actual re-engineering of physical business processes follows business re-engineering planning. Fig. 3 depicts steps in business re-engineering planning and examples of the re-engineering methods that our business knowledge model addresses.

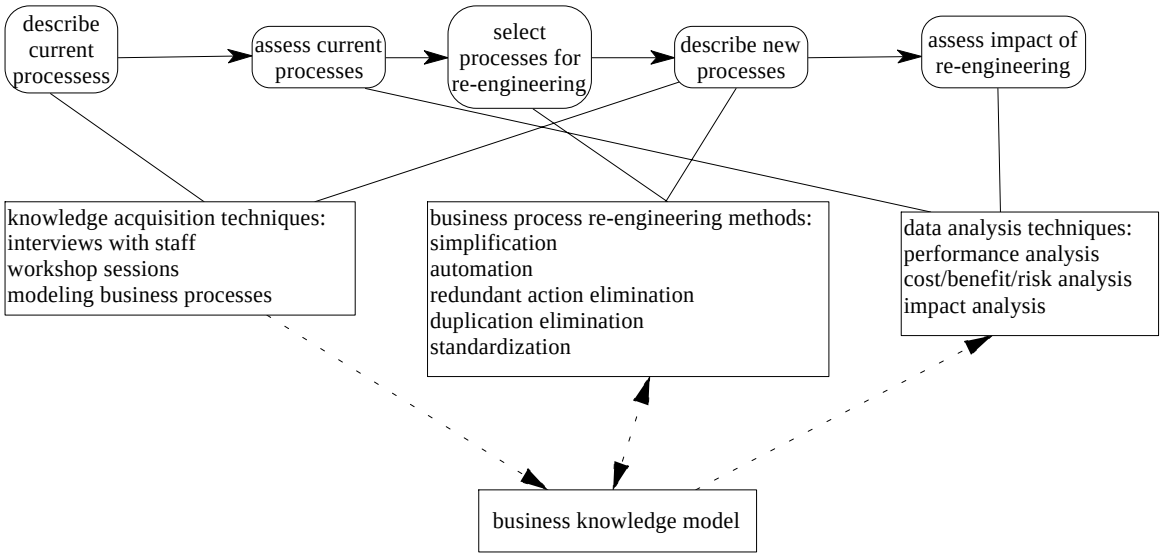

Fig. 3. Business re-engineering planning steps

We use the *business re-engineering impact diagram* (Fig. 4) to determine the boundaries of the business knowledge pertinent to business re-engineering. The impact diagram demonstrates potential benefits of proposed improvements of business processes. For a selected business process, the impact diagram depicts how re-engineering will affect the process itself, other business processes and the company as a whole. In our model, business processes are composed of actions. Similar diagrams are also built to analyze the impact of changes at the action level. The business re-engineering impact diagram is the final product of the business re-engineering planning phase. The business knowledge model should facilitate demonstration of business re-engineering benefits and risks in the form of such a diagram. In addition to that, the boundaries of the business knowledge model are determined by information requirements of business re-engineering methods that we plan to use.

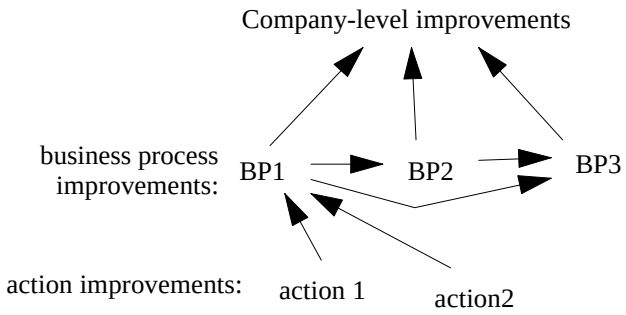

Fig. 4. The business re-engineering impact diagram

4. How do we build a generic business knowledge model?

Many business modeling methods, both formal [2,12] and informal [8,15], have been described in the literature. Our approach is based on modeling concepts used by others. We model business in terms of *features*. Features are types of business entities that we want to analyze during business process re-engineering and about which we want to store information. Fig. 5 shows a snap-shot of basic feature types that we model.

Process	Company	BPR-goal	Customer	Object
CompanyProcess	Strategy	Problem	External	File
SubProcess	CompanyGoal	Reason	Internal	Information
BusinessProcess	BusinessArea	Solution		Material
Action	Department			
BusinessRule	Role			
Event	Staff			
Quality-Measure				
ProcessInterface				

Fig. 5. Types of features in a business knowledge model

Features are described by properties. For example, the feature *BusinessProcess* has properties such as process owner, cost, work flow, efficiency and many others. (For readability, in the remaining text feature and relationship names are in italic and start with a capital letter.) Features can be related one to another using inheritance, aggregation and association. Relationships may also have properties. Types of property values include usual basic types, formal expressions, faceted classification schema [13] and informal text. In either case, each feature property identifies a piece of knowledge about the business that must be collected during interviews with the company staff, workshop sessions or through monitoring of the ongoing business operations. A

syntactic construct to describe properties of features is a *frame*, the term used in knowledge representation [1]. Frame slots contain properties of features.

4.1. Modeling features and feature relationships

We use Rumbaugh notation OMT [14] to model features and their relationships. We chose OMT because it allows us to express the macro structure of the business knowledge model in a natural way. Figures 6-9 show the general scope of the knowledge that we model[1] . Features are represented by rectangular boxes, triangles represent inheritance and lines between features represent binary relationships. Dots stand for 'many' connectivity in a relationship link. The meaning of a relationship link is clarified by the name attached to a link (in italic). In the inheritance link, a parent feature appears above the triangle and derived features appear below. We believe models are intuitive, so we only briefly comment on them.

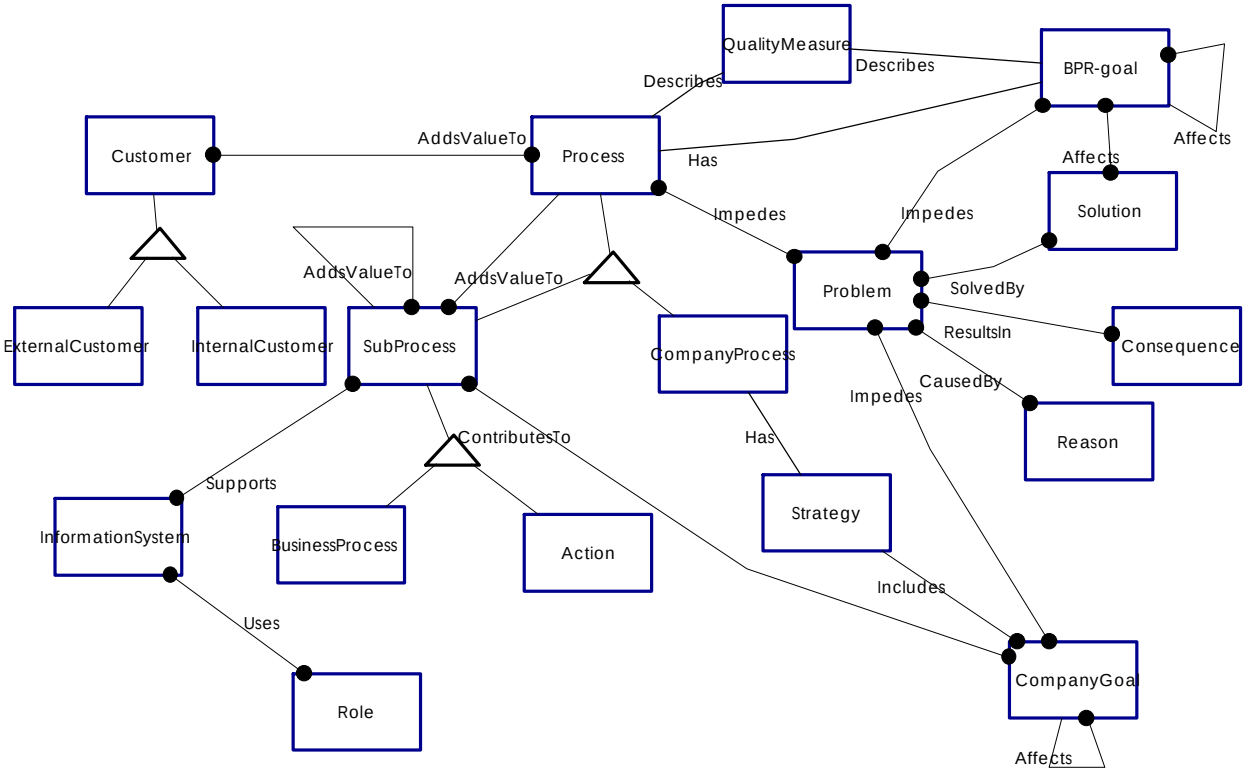

Fig. 6. Major features in a generic business knowledge model

Fig. 6 depicts three types of processes derived from abstract feature *Process*: *CompanyProcess*, *BusinessProcess* and *Action*. We model company *Strategy*, *CompanyGoals* and *Problems* that impede *CompanyGoals* and *Processes*. We show how *Solutions* to *Problems* affect goals of business re-engineering (feature *BPR-goal*). Value-added impact of *Processes* on each other and

[1] Business model diagrams are produced by MetaEdit, a trademark of MetaCase Consulting Oy.

on *Customers* are modeled by a family of relationships *AddsValueTo*. (Remark: A relationship defined for feature *Process* applies to all the features derived from *Process*.) The model also shows which *BusinessProcesses* and *Actions* contribute to which *CompanyGoals*. *Company-Goals* and *BPR-goals* are organized into a hierarchy by relationship *Affects*. Based on feature *QualityMeasure*, *Process* quality can be judged. Feature *QualityMeasure* can describe either the quality of current *Processes* or quality requirements to be met by re-engineered *Processes*. Target values for *Process* characteristics are set up based on company standards and world-class operation benchmarks [8]. Some of the *BPR-goals* are expressed in terms of *QualityMeasures*. By examining *QualityMeasures*, we select candidates for business re-engineering and evaluate the improvements of business re-engineering to see how business re-engineering goals are met. Features *Solution* and *BPR-goal* reflect the application of business re-engineering methods. We discuss these and other features used in modeling business re-engineering solutions later in the paper.

Fig. 7 depicts a taxonomy of process quality measures.

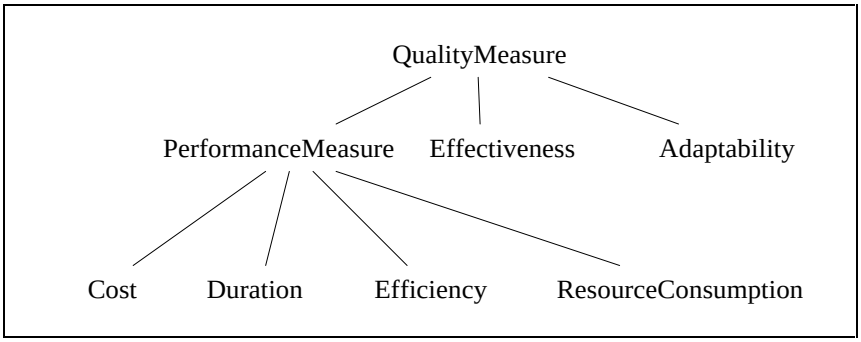

Fig. 7. Process quality measures

A model in Fig. 8 depicts the company structure, interactions between *Departments* and *Roles*.

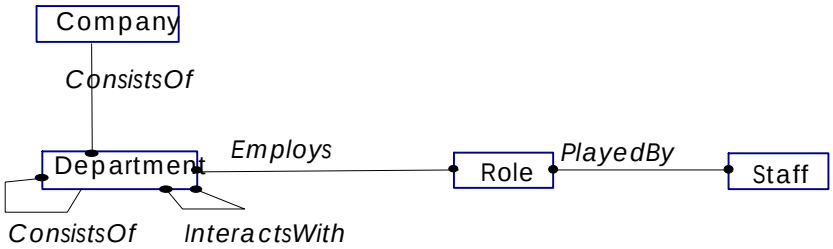

Fig. 8. A company structure model

4.2. Modeling business processes and business rules

Fig. 9 depicts the process structure. *BusinessProcesses* are grouped into *BusinessAreas*. *CompanyProcess* may consist of a number of *BusinessAreas*.

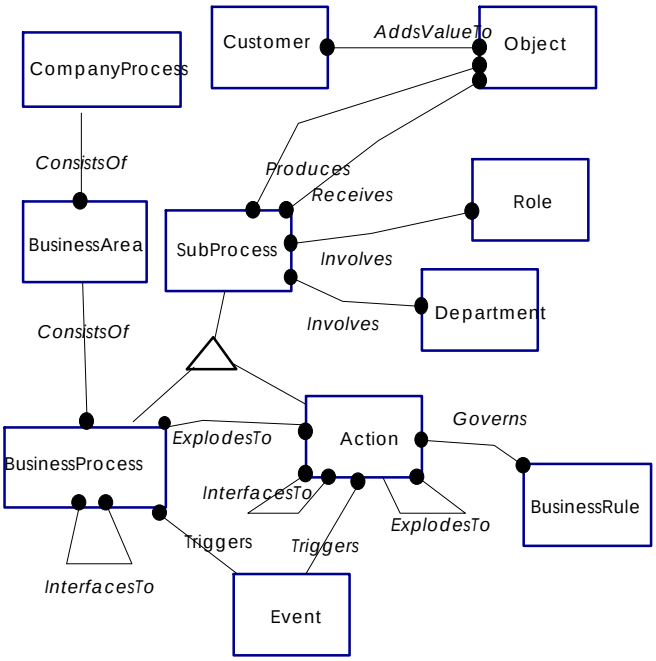

Fig. 9. A business process model

Actions describe detailed steps involved in a *BusinessProcess*: a *BusinessProcess* is exploded into level 1 *Actions*, each level 1 *Action* can be further exploded into more detail level 2 *Actions*, etc. One *Action* can be involved in many *BusinessProcesses*.

A *BusinessProcess* describes an end-to-end, usually cross-departmental, chain of business activities that deliver a product or service to a customer. *BusinessProcesses* are often triggered by external events, for example an arrival of a customer order. The *BusinessProcess* triggered by this event involves activities that start by customer order processing and end with product delivery to a customer. *BusinessProcesses* may also be initiated at fixed time intervals or when a certain condition arises. Feature *Event* models all situations that have to do with activation of a *BusinessProcess*. *Events* may also occur during *BusinessProcess* execution and influence execution of *Actions* that compose a *BusinessProcess*. *Action* execution is governed by *BusinessRules.*

The *BusinessProcess* structure can be conveniently depicted by a flow chart of *Actions* (Fig. 10). Each *Action* can be further exploded into a more detail flow chart. In Fig. 10, diamonds mark decision points at which alternative *Actions* can be taken. Issues that affect the decision are modeled as conditions. *BusinessRules* describe what happens at decision points or indicate chains of *Actions* that can be performed in parallel. Timing constraints (such as “wait for 1 hour before sending a report”) are also modeled as *BusinessRules*. In Fig. 10, Event1 triggers a business process whose first level of decomposition involves eight actions. After Action1 is completed, depending on condition ‘Cond1’, either Action2 or Action4 is initiated; Action3 is initiated once Action2 is completed; when Action4 is completed, Action5 may be performed in parallel with Action6 and Action7. In the diagram, small circles mark the end points of chains of alternative or

concurrent *Actions*. Fat arrows depict flows of business objects (information, documents, materials, etc.) between *Actions*.

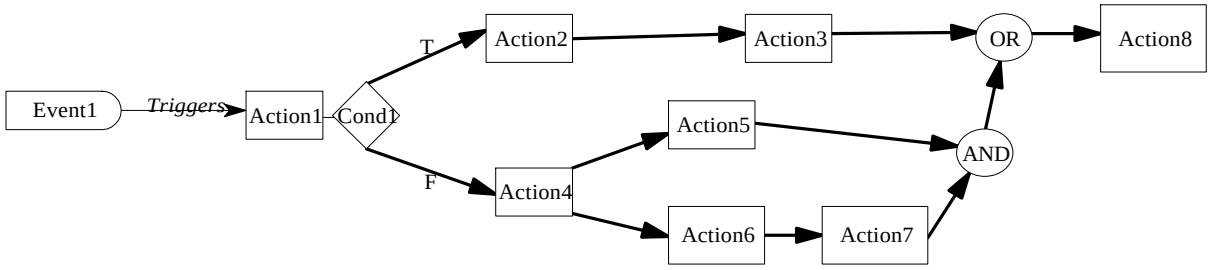

Fig. 10. An business process flow chart

Action flow charts and other diagrams form a user interface to the business knowledge model. They enable business analysts and company staff involved in a re-engineering project to communicate with the model in an easy and intuitive way. However, internally within the business model, the structure of a *BusinessProcess* is described by *BusinessRules* (see Fig. 9) that have the following formats:

Action1 NEXT Task1

meaning: Action1 must be completed before Task1 can be initiated.

ON Event1 Task1

meaning: when Event1 occurs, initiate Task1

[ON Event1] IF condition THEN Task1 [ELSE Task2]

meaning: when Event1 occurs and if condition is TRUE, then initiate Task1; if condition is FALSE, initiate Task2. In the above rule, ON clause and ELSE clause are optional.

In above rules, Task1 and Task2 may be substituted either by *Action* or by *BusinessRule*. Within the business model, the *BusinessProcess* of Fig. 10 would be represented by the following rules (BR is the name of the third *BusinessRule*):

ON Event1 Action1

Action1 NEXT BR

BR: IF Cond1 THEN Action2 ELSE Action4

Action2 NEXT Action3

Action3 NEXT Action8

Action4 NEXT Action5

Action4 NEXT Action6

Action6 NEXT Action7

Action7 NEXT Action8

Often problems occur at the interface points between *Actions* or between *BusinessProcesses*. Therefore, we explicitly model process interfaces (relationship *InterfacesTo* in Fig. 9). A process interface is a relationship between two *Actions* that, according to *BusinessRules*, communicate with one another. We also identify and model interfaces between *BusinessProcesses*. Process interfaces are described in terms of objects that are transferred among processes. An OBJECT is a family of features such as *Material, Document, Information, File, Database,* etc. OBJECTs are produced and received by processes (relationships *Receives* and *Produces* in Fig. 9). OBJECT properties include attributes, methods and constraints. Some of the *Actions* can be often mapped into OBJECT methods. For example, *Action* "print invoice" is likely to be a method "Print" of document "Invoice". OBJECT models are important as they integrate the business model with a model of software systems that we have to build to support re-engineered business processes. As the actual members of feature family OBJECT are domain-specific, they are defined in customized business models. We shall discuss OBJECT modeling later in the paper.

4.3. Modeling properties of features and relationships

Features and feature relationships have properties. A frame [1] is a syntactical structure to define feature properties. For each feature, we have a corresponding frame (with the same name) that defines feature's properties. Each property has a name, value type and an optional value. Property values can be defined at any level of feature inheritance network and are subject to inheritance. Properties can be grouped into specification sections. Property names are unique within a section. Section names are unique within a feature's frame. Star '*' following a property or the specification section name denotes multiple occurrence of this property or section in the frame.

Property value types include usual basic types such as TEXT (free text), INTEGER, (n:m) (a range of integers between n and m), REAL, BOOL, CHAR, DATE and MONEY. Built-in types are written in capital letters. Often used property value type is an enumeration. Enumerated keywords appear in round brackets, for example, QM-TYPE = (Cost, Duration, Resource, Efficiency, Effectiveness, Adaptability) describes various types of quality measures or TIME-UNIT = (minute, hour, week, month, year) describes time units. Value types also include constraints and assertions expressed in a formal notation (value type FORMAL). We show examples of FORMAL terms later in the paper.

User-defined value types include enumeration and faceted classification scheme [13]. Faceted classification consists of a set of typed keywords. Type name consists of feature name followed by keyword FACETS, such as CompanyGoal-FACETS in the example below. Company-

Goal-FACETS might specify the nature of a *CompanyGoal*. For example, *CompanyGoal* "increase revenue from sales" might be specified by types of issues involved: [Activity : Sales], [Measure : Money], [Objective : Increase], etc. If lists of keywords are used uniformly across features, faceted classification scheme enables one to selectively extract features relevant to a certain aspect. For example, one could retrieve all the *CompanyGoals* relevant to sales. Facets provide a flexible mechanism for extending pre-defined feature descriptions, whenever necessary.

Each property value domain includes a special symbol: NULL representing an undefined value. When a new frame is created, all properties are initialized to NULL, unless within the frame of this feature (or within any parent frame in the inheritance network), a property is given a default value.

Below we describe frames for some of the features.

CompanyGoal

goal-name : TEXT ident

goal-description : TEXT

goal-specs : CompanyGoal-FACETS

priority : (1:5)

current-rating : (1:5)

target-rating : (1:5)

evaluation-method : TEXT

due-date : DATE

end CompanyGoal

Explanations: Underlined ident indicates that a *CompanyGoal* is identified by a unique goal-name. Other properties describe *CompanyGoal's* priority on the scale between 1 and 5, assess current level of *CompanyGoal's* satisfaction, explain how to evaluate the level of *CompanyGoal's* satisfaction and provide the date by which the *CompanyGoal* should be met.

SubProcesses are related to *CompanyGoals* (Fig. 6). Properties of relationship *ContributesTo* include:

ContributesTo (SubProcess, CompanyGoal)

description-of-contribution : TEXT

evaluation-method : TEXT

current-rating : (1:5)

target-rating : (1:5)

end ContributesTo

QualityMeasures describe quality-related characteristics of *Processes* such as efficiency, effectiveness, adaptability, time/resource requirements, cost, etc. Feature *QualityMeasure* has the following frame structure:

QualityMeasure

qm-name : TEXT ident

qm-type : QM-TYPE

qm-description : TEXT

qm-unit : QM-UNIT

current-value : INTEGER

target-value : INTEGER

target-value-description : TEXT

calculation-method : TEXT

importance-rating : (1:5)

source : TEXT

end QualityMeasure

Explanations: QM-TYPE is a built-in enumeration that identifies the type of the BusinessProcess: QM-TYPE = (Cost, Duration, Resource, Efficiency, Effectiveness, Adaptability). QM-UNIT is also a built-in enumeration that includes units of measure such as TIME-UNITs, Money/TIME-UNIT, Man/TIME-UNIT, (n:m), etc. The range (n:m) defines the rating scale with no particular unit of measure.

Processes are described by *QualityMeasures* (relationship *Describes* in Fig. 6). Often the business process re-engineering goals can be expressed in terms of expected improvements of *Process* quality characteristics.

The reader can find more examples of frames in the appendix.

5. Modeling alternative re-engineering solutions

Fig. 11 shows how we model alternative business re-engineering solutions and the impact of proposed improvements on the business. We use inheritance to model commonalties (*BP-common*) and differences between *BusinessProcesses* before (*BP-before*) and after re-engineering (*BP-after*). Next, we model *Problems* that impede a *BusinessProcess*, *Reasons* that cause *Problems* to occur and *Solutions* to *Problems*. Application of *Solutions* results in re-engineered *BusinessProcesses*. The level of process improvement is evaluated through corresponding *QualityMeasures* related to a given *BusinessProcess*. Attributes of relationship Improves (Solution, BusinessProcess) model the overall improvement rate. This relationship also

describes the way proposed *Solutions* improve other *BusinessProcesses*. *Solution's* impact on a company as a whole is captured by relationships Affects (Solution, business re-engineering-goal) and Affects (Solution, CompanyProcess). Improvement at the level of *Actions* is modeled in a similar way. The business re-engineering impact diagrams (Fig. 4) can be developed based on the business re-engineering solution models of Fig. 11.

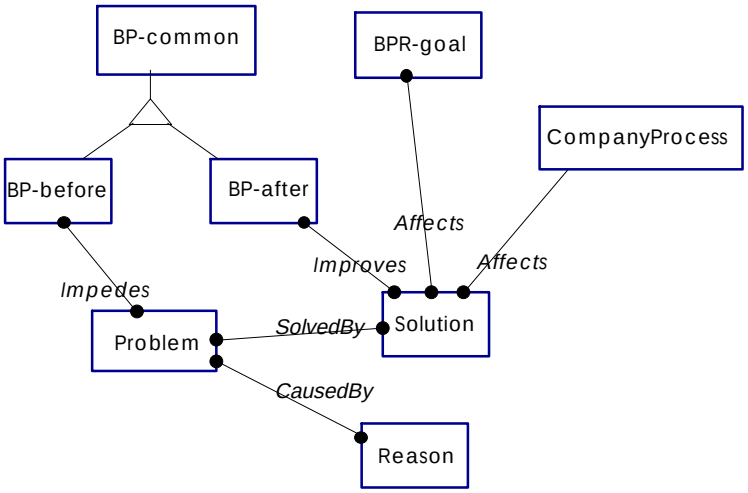

Fig. 11. Modeling improvements of business processes

Notice that in modeling features, we do not distinguish between a class and its instances: derived features inherit all the properties from parents together with values of any properties initialized at the superclass level. New features can be derived from existing ones. We use inheritance rules that allow us to hide, redefine and refine inherited property values. We described modeling rules useful in organizing business and software specifications in [11].

6. Modeling objects

OBJECT is a family of features such as *Material*, *Document*, *Data*, *Information*, *File*, *Database*, etc. OBJECT properties include attributes, methods and constraints. Process interfaces are described in terms of OBJECTs. We find OBJECT models useful in integrating business models with models of software systems that one needs to build to support re-engineered business processes.

OBJECT frames contain three sections describing OBJECT attributes, methods and constraints:

FeatureName : OBJECT

attributes { }

methods { }

object-constraints { }

end FeatureName

The detail properties of OBJECTs are domain-specific. They are identified during analysis of a given business domain and specified in a customized business model. Therefore, we do not standardize OBJECT frames any further. An example below shows a frame for OBJECT *Book* in a library system:

Book : OBJECT
attributes {
CatalogNo : Cat-No ident
Title : TEXT
#copies : INTEGER
Status : (borrowed, reserved, available) }
methods {
RegisterNew() : Action
CheckOut() : Action
CheckIn() : Action
Reserve() : Action
INTEGER IsOverdue() : Action }
object-constraints {
Reserve().pre-condition = (Status=available)
CheckIn().post-condition = (Status=available)
CheckOut().post-condition = (Status=borrowed) }
end Book

'Cat-No' is a set of valid catalog numbers for *Books*. The value of attribute 'Status' is a user-defined enumeration. Notice that, unlike properties of non-OBJECT features, OBJECT attributes are rarely assigned values. This is because the frame *Book* describes the whole collection of books rather than instances of this collection. Methods are identified by method signatures that include method name and optional returned type and argument types. Detail *properties* of methods are described by separate *Action* frames that are linked to the frame *Book*. Method 'IsOverdue()' says how many days a *Book* is overdue. Constraints describe OBJECT's properties in terms of rules and equations. The first constraint says that only books not available can be reserved. The second constraint says that once checked in, a book is available for loan and the last one - that after checking out, a book is not available for loan.

As during business re-engineering attention is focused on processes, some of the authors do not consider modeling of objects essential [15]. However, we feel that modeling objects offers the following three advantages to business re-engineering:

1. With objects we can better describe interfaces among processes.

An interface between Action1 and Action2 can be computed as intersection of two sets:

$$\{x : \text{Produces (Action1, x)}\} \cap \{x : \text{Receives (Action2, x)}\}$$

where *Produces* and *Receives* are relationships indicated in Fig. 9. Furthermore, the results of *Actions* can be described in terms of *Action's* impact on object's attributes.

2. During object analysis we can capture rules that characterize business operations in general.

During *BusinessProcess* modeling, *BusinessRules* are captured at instance level. For example, the first constraint in the frame *Book* explicates such a general rule. This rule might be never recognized if attention was not paid to objects. We include rules that involve only one object into the 'object-constraints' section of object's frame. Here are more examples of rules:

$$0 \leq \text{Book.\#copies} \leq 10$$

Meaning: There should be no more than 10 copies of a given title in a library.

if LibMember.Type=student **then** $0 \leq$ LibMember.#booksBorrowed ≤ 5

Meaning: This rule says that student library members cannot borrow more that 5 books. Here, *LibMember* is an OBJECT feature with attributes 'Type' (user-defined enumeration) and '#booksBorrowed' (INTEGER).

Many interesting rules involve more than one object. We specify such rules outside object frames:

if Book.IsOverdue > 10 **and** Borrowed (LibMember, Book)

then LibMember.SendReminder (Book)

Meaning: If a *Book* is overdue for more that 10 days, send reminder to a *LibMember* who borrowed the *Book*. *Borrowed* is a relationship between *LibMember* and *Book*.

We describe rules and constraints such as above in both formal and informal way. In this paper, we do not discuss the details of formal notations for constraints and rule specification. The reader is referred to other papers [2,12] for more details. Capturing rules in a formal notation during business modeling can be beneficial, particularly if the notation is intuitive and understandable to business analysts.

3. Object models form a natural link between business models and software systems that we must build to support business operations after re-engineering.

Some of the objects correspond to entities that will be represented in future software systems. Software requirements can be formulated in terms of such objects and the first-cut software architecture can be derived from them. In particular, a conceptual model of the central database for software systems can be derived from object attributes and relationships. Some of the object methods form sound candidates for software modules. Consequently, with object models, business analysis, business process re-engineering and future software development efforts can be addressed within a common modeling framework.

7. A user interface to a business knowledge model

A generic business knowledge model must be customized before it can be used on a specific re-engineering project. We customize the model by modifying properties of features and relationships, by modifying domains and values of properties and by defining company-specific OBJECT features. A customized business model is instantiated when we insert the actual business knowledge into the repository in the course of the business re-engineering project. As indicated in Fig. 2, a re-engineering expert must customize a generic model by hand. However, many non-experts will be interacting with the business knowledge model during a business re-engineering project. It is the role of the business re-engineering tools to hide the complexity of the business model and to enable non-experts to easily interact with the model. In this section, we discuss examples of user-level views of the business model displayed by such tools. Our examples are based on the generic business knowledge model. Examples of customized models and views derived from customized models will be described in another publication.

7.1. A company structure view

An organization structure view (Fig. 12) shows how authority, responsibilities and activities are delegated down into the organization. This diagram also reflects the communication paths in the organization. The main features that we represent are *Departments* and *Roles*. Different types of links are used to distinguish the inter-department communication from hierarchical relationships. *Roles* can be modeled separately from *Departments*. However, this may make it difficult to model relations between *Roles* in different *Departments*. Hence, we also model *Roles* at the same level as *Departments*.

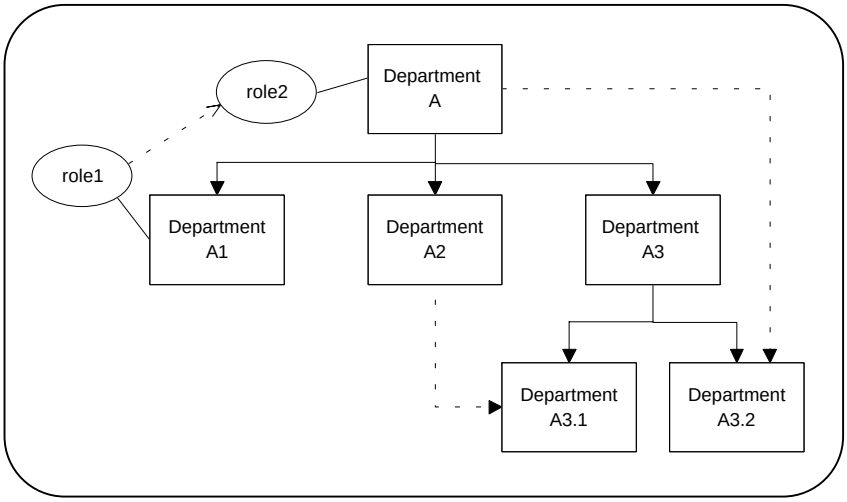

Fig. 12. An organization structure view

7.2. Process views

Process views depicted in Fig. 13, 14 and 15 are derived from process models of Fig. 6 and 9. At the top level (Fig. 13), the *CompanyProcess* is modeled together with the *Customers* who supply OBJECTs (i.e. materials, information, services, etc.) to the company, and those who buy OBJECTs (i.e. products, services, etc.) from the company. This gives the highest level view of the company as a whole.

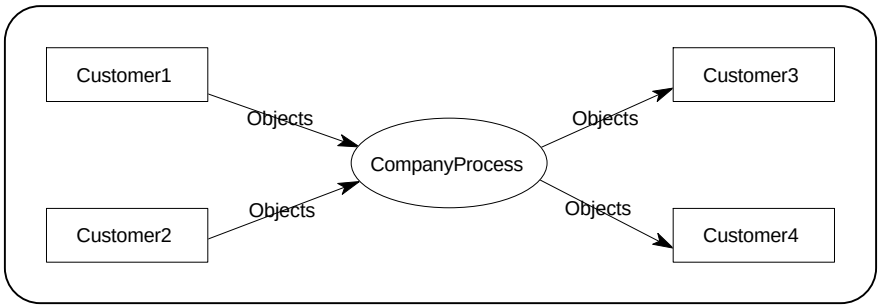

Fig. 13. A company process view

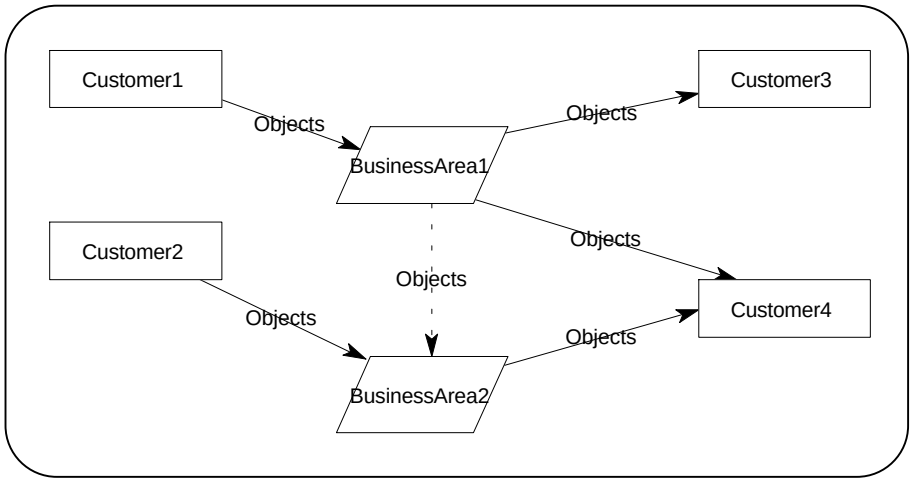

Fig. 14. A business area view

The company process view is linked to the business area view (Fig. 14). The business area view shows *Customers* that provide inputs to and get outputs from different business areas. Interactions between business areas are also shown.

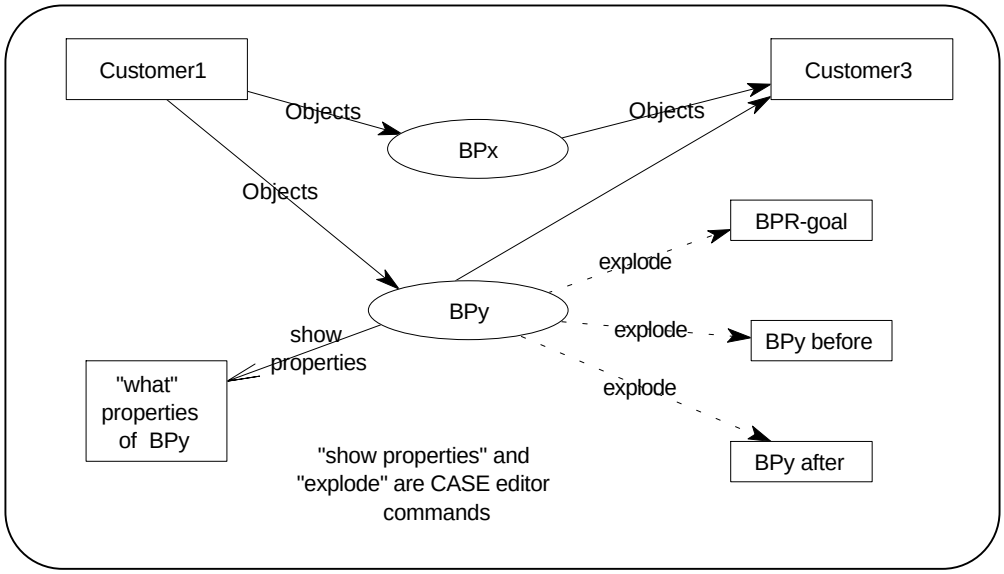

Fig. 15. A business process view

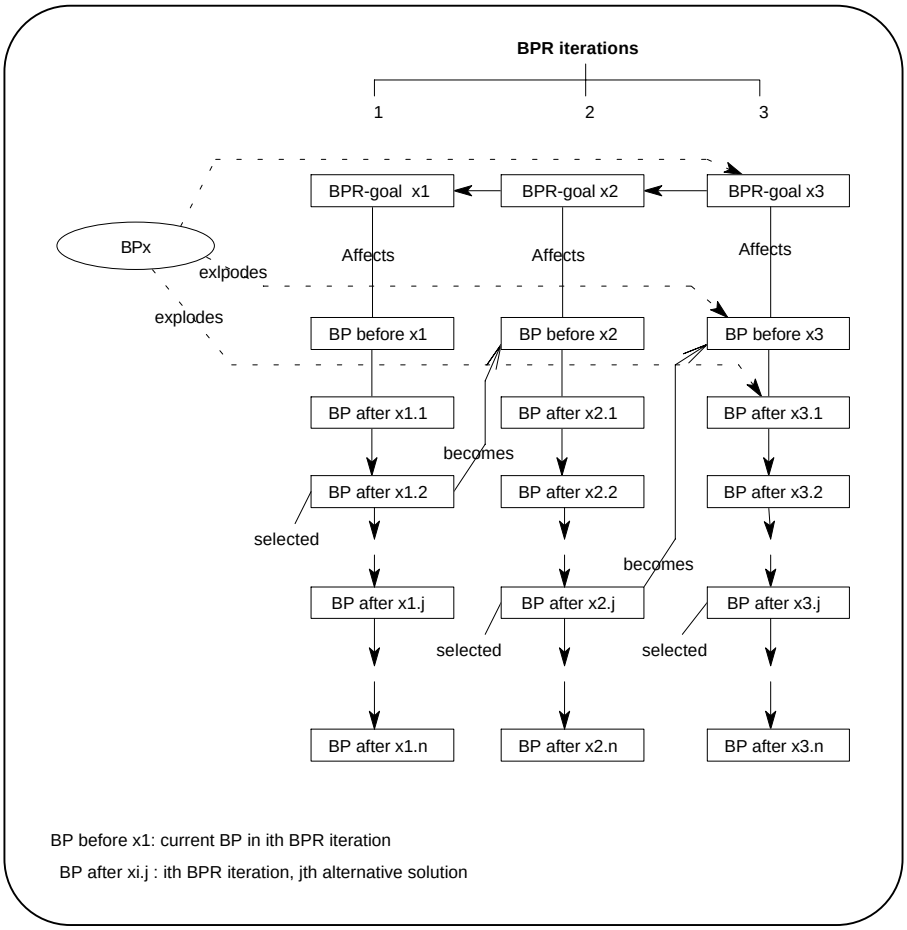

Fig. 16. Business re-engineering alternatives and iterations

Each business area can be further exploded to its component business processes (Fig. 15). Each business process can be linked to process vision (feature *BPR-goal*), current process model (*BP-before*) and re-engineered process (*BP-after*).

A view of Fig. 16 depicts alternative re-engineering solutions and iterations over business re-engineering cycles. Business process "BPx" evolves over three re-engineering iterations. In the first iteration, "BPx" has "BPR-goal x1". This goal affects the current business process ("BP before x1"). And it provides a direction for the design of some alternative business processes which will meet the targeted improvements. These alternatives are depicted by "BP after x1.1" to "BP after x1.n". From these alternatives, one is selected ("BP after x1.2") for implementation. In time, business environment and technology evolves, so *BPR-goals* can evolve too ("BPR-goal x2" evolves from "BPR-goal x2"). Hence, in the next iteration, "BP after x1.2" becomes the current business process ("BP before x2"), and the cycle repeats. Each *BusinessProcess* is exploded into an Action flow chart such as one in Fig. 10.

8. How do we use a business knowledge model?

Let us start by stating who and how is going to use the business knowledge model. Two different stakeholders use the model, namely *business analysts* and *designers of tools* for business understanding and re-engineering.

- *To business analysts*

  the model drives the process of business re-engineering planning that precedes the actual re-engineering of physical business processes. In particular, the business knowledge model:

  1. defines a common vocabulary to talk about business,
  2. identifies types of business knowledge that must be collected,
  3. defines conceptual schema for that knowledge,
  4. defines parameters by which the quality of business processes can be judged,
  5. provides a reference model for business re-engineering methods and techniques which extract and use business knowledge,
  6. captures enough semantics about the business to support the many types of business analysis required in business re-engineering,
  7. maintains a good balance between formal and informal knowledge to reflect various modes of business analysts operation (informal, creative "soft" thinking and formal, analytic "hard" thinking)
  8. can be customized to realities of a given company and to needs of a specific business re-engineering project.

The business analysts are not supposed to work directly with the model. Tools are needed to provide an easy interface to the model and to enable the model to play a role of an active and useful assistant in business re-engineering projects. The business model defines an internal mechanism for tools that we build around the model. The role of tools is to provide an easy, intuitive and diagram-based access to the business knowledge.

- *To the designers of tools for business understanding and re-engineering*
  the business knowledge model provides a basis for automation of some of the routine tasks involved in business process re-engineering. In particular, the business knowledge model defines conceptual schema for a repository where all the business knowledge will be stored and through which various business re-engineering tools can be integrated. The physical repository schema and customized tools for a given company and a specific business re-engineering project can be automatically generated from the customized business knowledge model.

As the business knowledge model cannot not be fixed once for all, an important requirement for an automated business re-engineering environment is flexibility. The tools should be adaptable to versions of a generic business model customized to specific business re-engineering projects. To satisfy this requirement, we use a CASE generator that can construct editors from specifications of modeling methods. Business model views described in chapter 7 are supported by graphical editors. Business knowledge is extracted from diagrams and stored in the repository. We build the repository according to the business knowledge model schema. The repository is implemented in PROLOG.

We also implemented a business knowledge acquisition tool to support interviews with company staff. An interview assistant tool supports business process re-engineering facilitator in conducting interviews and in feeding knowledge acquired during question/answer sessions into the repository. Finally, we experimented with types of impact, performance and quality analysis that are routinely done during business re-engineering planning. We refer the reader to [10] for a detailed description of our tool prototypes.

9. Conclusions

In this paper, we formalized information requirements for business re-engineering within a business knowledge model. We use knowledge representation methods to build the business model. Our model plays the following two major roles:

1. The business model provides a reference model for business process re-engineering methods. Business re-engineering techniques (such as interviews, groupware, business process modeling, performance analysis) and methods (such as process simplification and re-designing

process interfaces) can be related to the model and organized around it. In particular, for each business re-engineering method/technique, the information that must be collected, the impact on business, etc. can be expressed in terms of the business knowledge model.

2. The business model defines the schema for a repository of the business knowledge. Tools can be implemented around the repository to provide a simple, user-friendly access to the business knowledge acquired in the course of a business re-engineering project. We implemented tool prototypes that help in business knowledge acquisition, display information in the form of diagrams and support various types of business knowledge analysis [10].

We started a project with the Temasek Polytechnic in which we shall evaluate, customize and refine our business model and tools. In the future, we plan to address business re-engineering methodology issues. If heuristic rules reflecting knowledge about re-engineering methods could be identified, then tools could also provide assistance for decision making and methodological guidance through out the business re-engineering project.

10. Acknowledgments

This work was supported by National University of Singapore Research Grant RP920613. Thanks are due to the students who implemented various components of tool prototypes. Seow Mui Leng refined the business model, implemented tools for graphical modeling, contributed diagrams to this paper and did a case study. Lau Ai Leng, Kuan Sook Peng and Catherine Tan implemented an interview assistant.

References

[1] Adeli, H. (editor) *Knowledge Engineering*, vol I, McGraw-Hill, 1990

[2] Berztiss, A. (1990) 'The Specification and Prototyping Language SF' Report No 78, SYSLAB, The Royal Institute of Technology, Sweden

[3] Chen, P. 'The Entity-Relationship model -- toward a unified view of data' *ACM Transactions on Database Systems*, vol. 1, no. 1, 1976, pp 9-36

[4] Davenport, T.H. *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press, Boston, Massachusetts, 1993

[5] Eliot, L. *Information System Strategic Planning*, first edition, Computer Technology Research Corp., Charleston, S.C., USA, 1991

[6] Hammer, M. and Champy, J. *Re-engineering the corporation: A manifesto for business revolution*, Nicholas Brealey Publications, 1993

[7] Hansen, G. *Automating Business process Re-engineering*, Prentice Hall, Englewood Cliffs, 1994

[8] Harrington, H. *Business Process Improvement*, McGraw-Hill, Inc., 1991

[9] Jarzabek, S. 'Strategic re-engineering of software: lifecycle approach' *6th Int. Workshop on CASE, CASE'93*, Singapore, July 1993, pp 211-220

[10] Jarzabek, S. and Ling, T.W. "Model-based Design of Tools for Business Understanding and Re-engineering," *Technical Report TRD3/95*, Department of Information Systems and Computer Science, National University of Singapore, March 1995

[11] Jarzabek, S. and Tan, C.L. “Modeling multiple views of common features in software re-engineereing for re-use” *Proc. 6th Int. Conference on Advanced Information Systems Engineering, CAiSE’94*, Utrecht, June 1994, *Lecture Notes in Computer Science*, no. 811, Springer-Verlag, pp 269-282

[12]McBrien, P. *et al* ‘A Rule Language to Capture and Model Business Policy Specifications’ *Proc. 3rd Int. Conference on Advanced Information Systems Engineering CAiSE’91*, Trondheim, May 1991, *Lecture Notes in Computer Science*, no. 498, Springer-Verlag, pp 307-318

[13] Prieto-Diaz, R. “Classification of reusable modules,” IEEE Software 4(1), 1987, pp. 6-16

[14] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. *Object-Oriented Modeling and Design*, Prentice-Hall, 1991

[15] Spurr, K., Layzell, P, Jennison, L. and Richards, N. *Software Assistance for Business Re-engineering*, John Wiley & Sons, 1993

[16] Strassmann, P. The Business value of computers, *The Information Economics Press*, 1990

[17] Ulrich, W. ‘Re-development engineering: formulating an information blueprint for the 1990’s’ *CASE Outlook*, No. 2, 1990, pp 15-21

Apendix A. Examples of feature frames from the generic business knowledge model

The following frame describes properties of feature *BusinessProcesses*:

```
BusinessProcess
    BP-name : TEXT ident
    BP-description : TEXT
    BP-specs : BusinessProcess-FACETS
    pre-condition : TEXT
    post-condition : TEXT
    cycles-per-time-unit {
        #cycles : INTEGER
        time-unit  : TIME-UNIT
        variance : (n:m) }
    business process re-engineering-indicators {
        BP-owner : Role
        (AddsValueToCustomer : Customer)*
        (AddsValueToProcess : BusinessProcess)*
        accumulated-value-added {
                overall-rating-of-added-value : INTEGER
                target-rating : INTEGER
                added-value-description : TEXT
                added-value-specs : AddsValueTo-FACETS }
        #staff-required-to-complete-BP : QualityMeasure
        duration  : QualityMeasure
        cost : QualityMeasure
        efficiency : QualityMeasure
        effectiveness : QualityMeasure
        adaptability : QualityMeasure
        overall-quality-rating {
                rating : (1:5)
                description : TEXT }
        percentage-of-activities-handled-as-exceptions : (0:100)
        percentage-of-on-time-completions : (0:100)
        time-to-respond-to-customer : TIME
        #errors-per-BP-cycle : INTEGER
```

```
        overtime-increase : (1:100)
        #staff-increase : (1:100)
        optimization-level : (1:5)
        automation-level : (1:5) }
    (information-system-requirements : InformationSystem)*
end BusinessProcess
```

Explanations: Property ‘pre-condition’ describes requirements that must be satisfied before a *BusinessProcess* can start. Property ‘post-condition’ describes the state of business upon completion of a *BusinessProcess*. Property ‘cycles-per-time-unit’ is a structure that has three data elements as its components. Section ‘business process re-engineering-indicators’ contains process characteristics that are important for business process re-engineering. For completeness, we included references to relevant *BusinessProcesss* into this section. Property ‘BP-owner’ shows the Role of the process owner. (Notice that this property defines a relationship between the *BusinessProcess* and *Role* named ‘business process re-engineering-owner’.) The total value-added of a *BusinessProcess* is represented by a property ‘accumulated-value-added’. The last property describes requirements for information systems that are needed in order to support the *BusinessProcess*.

Frames for *BusinessRules* are defined as follows:

```
NextRule
    rule-name : TEXT ident
    rule-description : TEXT
    rule-specs : Rule-FACETS
    next-task {
        task-type : (Action, Rule)
        task-name : TEXT }
end NextRule
IfElseRule
    rule-name : TEXT ident
    rule-description : TEXT
    rule-specs : Rule-FACETS
    condition {
        cond-description : TEXT
        cond-specs : FORMAL }
    then-task {
        task-type : (Action, Rule)
        task-name : TEXT }
    else-task {
        task-type : (Action, Rule)
        task-name : TEXT }
end IfElseRule
```