

Identifying Cognitive Aspects to Improve Business Process Reengineering

Adriana Martín and Alejandra Cechich*

Departamento de Ciencias de la Computación, Universidad Nacional del Comahue,

Buenos Aires 1400, Neuquén, Argentina

Tel. (+54) 299 4490 312 – Fax: (+54) 299 4490 313

Email: {martinae@infovia.com.ar}; {acechich}@uncoma.edu.ar}

Abstract.

Knowledge-intensive processes are widely used to recover from errors, handle exceptional cases and complaints, and to improve or adapt a process itself. In this context, evolved Business-Process Reengineering (BPR) techniques are changing to give some answers to this reality. In this paper, we identify some cognitive aspects used by traditional and recent reengineering models. We provide a framework highlighting how cognitive aspects might improve reengineering through knowledge and perception modelling.

Keywords. Business Process Reengineering, Knowledge-Intensive Processes, Cognitive Science

1 Introduction

The concept of reengineering traces its origins back to management theories developed as early as the nineteenth century [13][15]. The term *reengineering* started to sound noisily in 1993, when Hammer and Champy presented the book “Reengineering the Corporation: A Manifesto for Business Revolution” [2]. But this book is not the only approach for reengineering. Table 1 transcripts some definitions on reengineering, highlighting their main focus, year of publication and references.

From Table 1, it becomes clear that in the first half of the nineties, reengineering was proposed as a radical approach to redesign business processes. Currently, reengineering methods and tools are usually applied in an incremental way as just one aspect in more extensive organizational change projects [6]. For example, at the beginning software reengineering was proposed as a radical transformation process with solid conceptual foundations [1][7]; then the advantages of object-oriented and information technology were incorporated to improve BPR [3]; and finally BPR methods and tools were applied in an incremental style considering other design aspects, such as software architectures [4] or pattern-based approaches [15].

But software systems have been in continuous evolution, truly deep in the last few years, were complex and *knowledge-intensive processes* have been commonly required in software systems. Knowledge-intensive processes are those in which knowledge is used to make decisions or create an output, like new product development or legal support and risk assessment. In general any process involves knowledge-intensiveness to some extent for recovering from errors, handling exceptional cases and complaints, and for improving or adapting the process itself [6].

In this paper, our main goal is to identify how knowledge and cognitive aspects are used by reengineering approaches. In section 2, we describe four conventional reengineering approaches. Then, in Section 3 we present two reengineering approaches which include concepts from cognitive science. In Section 4, we propose a framework for identifying cognitive aspects in conventional reengineering approaches and provide some discussion. Conclusions and future work are addressed in the final section of this paper.

* GIISCo (Grupo de Investigación en Ingeniería de Software del Comahue), UNComa.

Year	Main Focus	Reference
1990	Software reengineering (also known as both renovation and reclamation), suggests the examination and change of a software system to be rebuilt in a new software form, and the subsequent implementation of that form.	[7]
1992	The goal of software reengineering is to take an existing system and generate from it a new system, called the target system, which has the same properties as a system created by modern software development methods. These desired software properties include: maintainability, portability, reliability, reusability, usability, testability, and quality of documentation.	[1]
1991 - 1995	This reengineering approach is defined as the process of creating an abstract model of the system to be reengineered, reasoning about the changes in that abstract model, and then re-designing the system. The approach is known as a business process reengineering with object-technology.	[8] [3]
1993	Process Innovation is the major reduction in process cost or time, or major improvements in quality, flexibility, service levels or other business objectives.	[9]
1993	Reengineering is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service and speed.	[2]
2000	Reengineering is the fundamental rethinking and radical redesign of business processes by leveraging proven models of processes, organization, and technology. Reengineering aims at improving changes and reducing time to achieve measurable business results.	[10]
2002	The goal of reengineering is to reduce the complexity of a legacy system in such a way that it can be used and adapted at an acceptable cost. Reengineering Patterns codify and record knowledge about modifying legacy software: they help in diagnosing problems and identifying weaknesses which may binder further development of the system, and they aid in finding solutions which are more appropriate to the new requirements.	[15]
2002	A software architecture perspective on reengineering states that transformations should be done especially at the architectural level: architecture is recovered, reengineered and instantiated to become a desirable new architecture.	[4]
2003	Business process reengineering is the restructuring of tasks of an organization in such a way that information technology is optimally exploited. A BPR method should be decision-driven and performance-determined: the process is analyzed with the aim of deciding whether a particular design principle is applicable, and to assess consequences of alternatives on performance indicators.	[6]

Table 1: Main Focus and Definitions for Reengineering over the years

2 Conventional reengineering approaches

To illustrate conventional reengineering approaches, following we briefly introduce four of them, which are representative of a reengineering period as well as currently relevant.

Byrne's Model: In 1992, Erich Byrne presented a conceptual foundation for software re-engineering and established a general model of software re-engineering based on these foundations [1]. This conceptual foundation consists of a collection of *Properties* (Table 2), *Principles* (Table 3) and *Assumptions* (Table 4) that are often either loosely stated or implied when discussing re-engineering methods and problems.

1. **Separation of Concerns:** For each level of abstraction there exists a system representation that explicitly describes system's characteristics. Each level of abstraction defines a different set of characteristics.
2. **Information Inclusion:** (a) Information contained within a level of abstraction influences information contained at lower abstraction levels.
(b) Information contained within a level of abstraction has no influences on information contained at higher abstraction levels.
3. **Information Volume:** As the level of abstraction decreases, the amount of information describing a system increases.
4. **Creation of Characteristics:** Each characteristic of a system is created at a particular abstraction level, and the influence of each characteristic propagates downwards to lower abstraction levels.

Table 2: Properties of Byrne's Conceptual Foundation

1. **Refinement:** The gradual decrease in the abstraction level of a system representation is caused by the successive replacement of existing system information with more detailed information.
2. **Abstraction:** The gradual increase in the abstraction level of the system representation is created by the successive replacement of existing detailed information with information that is more abstract. Abstraction produces a representation that emphasizes certain system characteristics by suppressing information about others.
3. **Alteration:** Alteration is making of one or more changes to a system representation without changing the degree of abstraction. Alteration includes the addition, deletion, and modification of information.

Table 3: Principles of Byrne's Conceptual Foundation

1. Reengineering a software system produces a new form of the system that is better, in some way, than the original.
2. Software reengineering begins with an existing system representation expressed at some level of abstraction.
3. To alter a system characteristic, reengineering should be done at the level of abstraction in which information is explicitly expressed.
4. A system characteristic can be altered by working within a level of abstraction below the level at which information is explicitly expressed. However, the best re-engineering result might not be achieved.
5. A system characteristic cannot be altered by working within a level of abstraction above the level at which information is introduced.
6. For any existing software system, the only accurate and up-to-day system representation is source code. Any existing requirement or design documentation cannot be trusted to describe the current system accurately.
7. If a system representation at a particular abstraction level is missed or not up-to-date, it is possible to rebuild that representation, at least partially.
8. A system representation at a particular level of abstraction can be rebuilt, at least partially, from an existing representation expressed at a lower level of abstraction by using the principle of abstraction.

Table 4: Assumptions of Byrne's Conceptual Foundation

Figure 1 shows Byrne's Model where we can identify three steps: *Reverse Engineering*, *Re-Structuring* and *Forward Engineering*, respectively deliverables of an Abstraction, an Alteration and a Refinement process.

This model can be interpreted using the conceptual foundations summarised in Tables 2-- 4. For example: Property 1 and Property 2 establish the abstraction levels and their ordering; and the triangular shapes derive from Property 3 - see Table 2. Assumptions 7 and 8 justify the use of the Principle of Abstraction, which is the basis for Reverse Engineering; Assumption 3 and the Principle of Alteration show the level of abstraction at which certain types of change can be made; and the Principle of refinement is the basis for Forward Engineering, which creates a target system implementation - see Table 3 and Table 4.

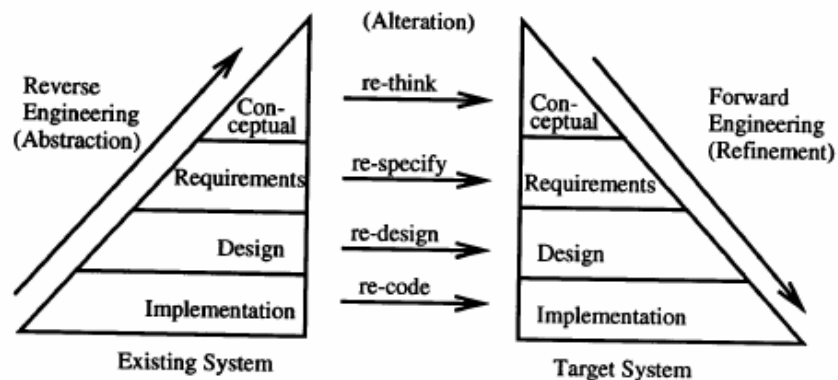


Figure 1: General Model for Software Re-engineering (from [1])

Jacobson's Method: This method [3] shows reengineering work being performed within the framework of business development, and it emphasizes that reengineering consists of mainly two steps: *Reverse Engineering* of the existing business and *Forward Engineering* of the new business. Intuitively and a little naively, Jacobson said that:

$$\text{Business Reengineering} = \text{Reverse Business Engineering} + \text{Forward Business Engineering}$$

(1) (2)

- (1) Understanding the existing company is a reverse-engineering activity; it implies that we make an abstract model of the business and the processes we wish to improve.
- (2) Designing the new company is a forward-engineering activity.

Figure 2 shows the main activities of a reengineering project, and is also a simplified model of how reengineering work is carried out. The arrows primarily show how the different activities convey information to one another and to the environment. The project starts in response to a reengineering directive, which explains why something must be done and specifies what the project should achieve. The reengineering directive triggers an activity called envisioning, which visualizes the new business enterprise or the new processes in the business, and the result of this activity is an objective specification – a vision of the future business. Thus, envisioning triggers the reversing of the existing business activity, which produces a model of the existing business. Engineering the new business implies creating one or more new processes, designing them, developing a supporting information system, perhaps simulating or prototyping them, and so on, to produce a model of the redesigned business. A reengineering project will normally include another main activity: *installing the redesigned business* in the real organization [3]. Finally, the project finishes with the reengineered corporation.

Summing up, this method presents an interesting technique for modelling business looking at company's usability, identifying customers as users of the business, and suggesting that the business model should be designed to offer a "usable" company. The reengineering technique is based on use cases.

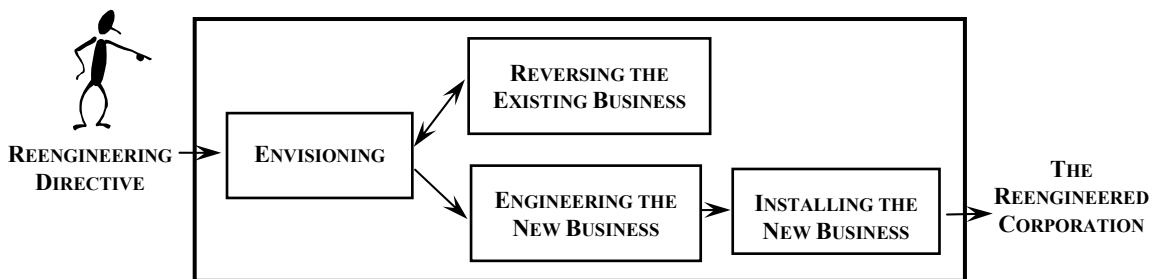


Figure 2: An overview of the main activities in a Reengineering project according to Jacobson's method (from [3])

Demeyer-Ducasse-Nierstrasz's Patterns: In [15] *Patterns*, as a way of communicating best practices, are particularly well-suited to presenting and discussing standard techniques that have emerged from reengineering processes. Reengineering patterns codify and record knowledge about modifying legacy software: they help in diagnosing problems and identifying weaknesses, which may hinder further development of the system. Patterns also aid in finding solutions which are more appropriate to the new requirements.

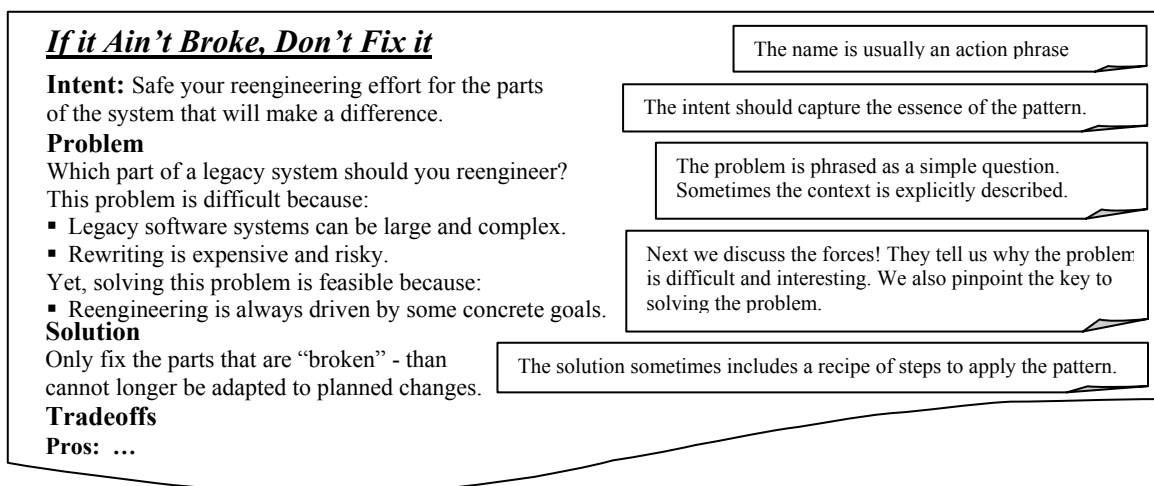


Figure 3: The format of a typical Demeyer-Ducasse-Nierstrasz's Pattern (from 15)

Whereas a design pattern presents a solution for a recurring design problem, a reengineering pattern presents a solution for a recurring reengineering problem. The artefacts produced by reengineering patterns are not necessarily designs. They may be as concrete as re-factored code, or in the case of reverse engineering patterns, they may be abstract as well as insights into how the system functions. The mark of a good reengineering pattern is (a) the clarity with which it exposes the advantages, the cost and the consequences of the target artefacts with respect to the existing system state, and (b) the description of the reengineering process - how to move from one state of the system into another [15]. Figure 3, shows part of a simple reengineering pattern that illustrates the format proposed by Demeyer, Ducasse and Nierstrasz. This format may vary slightly from pattern to pattern, since they deal with different kinds of issues.

The SEI Horseshoe Model: The SEI's Model is similar to Byrne's Model, and takes a Software Architecture perspective on reengineering [4]. This conceptual model distinguishes different levels

of reengineering analysis and provides a foundation for transformations at each level, especially at the architectural level.

As Figure 4 illustrates, there are three basic reengineering processes: *Analysis* of an existing system, *Logical Transformation* and *Development* of a new system. In the purest and most complete form of the model (represented by the large outlined arrows), the first process recovers the architecture by extracting artefacts from source code. The second process is *architectural transformation*; in which the “as-built” architecture is recovered and then reengineered to become the desirable new architecture. It is re-evaluated against the system’s quality goals and subjected to other organizational and economic constraints. The third process uses architectural-based development to instantiate the desired architecture. In this process, packaging issues are decided and interconnection strategies are chosen. Code-level artefacts from the legacy system are often wrapped or rewritten in order to fit into this new architecture [4].

The purpose of the visual metaphor of the “horseshoe” is used to integrate the code-level and the architectural reengineering views of the world, explicitly showing the different levels in which reengineering analysis and transformations occur, but especially highlighting the architectural level as the main level to initiate and propagate transformations.

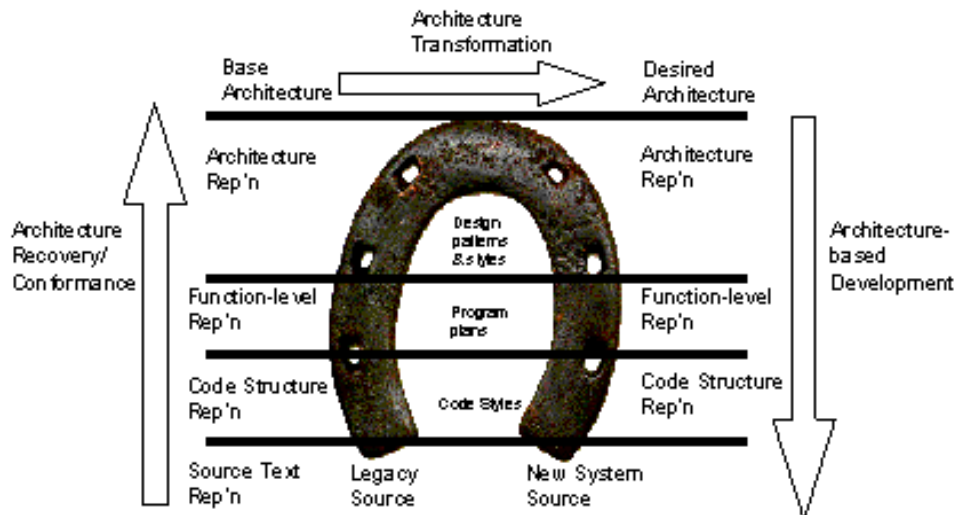


Figure 4: The SEI Horseshoe Model
(from [4])

3 Non-Conventional reengineering approaches: a Cognitive perspective

It is a fact, that BPR activities implies not only using good reengineering practices, but also gathering and manipulating process knowledge from machines and human minds. Within this vision, re-engineering business processes can be seen as a special case of knowledge management and a work domain in which human cognition plays a critical role [6]. Many software development tools and environments are considered useful because they beneficially reengineer developer cognition [5].

Consequently, we can say that in a cognitive work domain like software re-engineering, an approach can be considered useful if it improves cognitive reengineering potential and issues. Cognitive models and theories can contribute for a successful reengineering process, assisting reengineers in an intelligent use of large quantities of explicit and tacit knowledge. Research on this area is just starting. As examples, following, we briefly introduce two frameworks, which include concepts from cognitive science to improve reengineering analysis and reengineering of knowledge-intensive processes, respectively.

Walenstein’s Framework: In 2002, Andrew Walenstein proposed a Lightweight Framework for Cognitive Reengineering Analysis called HASTI [5] [12]. The intent of HASTI is to support “quick and dirty” cognitive reengineering analysis during the early stages of design. This work argues that designers indirectly reengineer cognition by engineering technological environments, so the main purpose of a cognitive model in this context is to say what should be re-engineer and how.

HASTI is a cognitive decomposition-and-integration framework that identifies five different “dimensions” or “aspects” of cognition. “HASTI” is an acronym created from the names of the model’s structures associated with the five dimensions or aspects: (1) a **H**ardware model, (2) an **A**gent model, (3) a **S**pecialization layering, (4) a **T**ask or problem partitioning, and (5) an **I**nteraction abstraction hierarchy [5].

The five modelling methods are summarized in Table 5. Aspects or dimensions are further described by using additional methods and techniques. For example, the *model technique* at *agent* level (A) is borrowed from related work on the blackboard and agent modelling literature, such as the Craig’s Agent-Based Architecture [16]; the *decomposition* at *specialization* level (S), is inspired on Rasmussen’s “SRK” taxonomy [17] that proposes three behavioural categories for human cognition – called “skill-based”, “rule-based” and “knowledge-based”; and so forth. We refer the reader to [5] [12] for further details on the way each aspect is described.

	Cognition Aspect / Dimension	Decomposition	Model Technique
H	invariant capabilities / constraints	memory, processors	hardware model
A	goal-related behaviour	agents, resources	agent model
S	specialization (adaptation)	SRK taxonomy	layering
T	task / problem	4-fold taxonomy	partitioning
I	interaction	virtual architectures	layering

Table 5: Walenstein’s five Dimensions or Aspects of Cognition and the way they are modelled
(from [5] [12])

HASTI is proposed as a lightweight modelling framework that highlights possibilities for reengineering cognition. Since HASTI is a computational model, cognitive reengineering corresponds precisely to *computational reengineering*. It might be defined as the reorganization of the structure of a computational system without changing its essential functionality. Thus the ways of improving cognition can be identified with different computational reengineerings of HASTI [5].

van Leijen-Baets’s Framework: In 2002, Hans van Leijen and Walter Baets developed a cognitive framework for improving knowledge-intensive aspects of administrative processes and gave some implication of this model for analyzing, designing, and implementing phases of reengineering knowledge-intensive processes efforts [6].

The authors’ view on executing a case in a business process corresponds to solving a problem, and analyzing this problem-solving behaviour in a set of generic phases. Choosing a cognitive model means speaking in terms of the mental processes and states of an agent. In the service organization, a goal is always an action that serves or protects the interest of a stakeholder, such as a customer, an employee or the principal agent of the organization. The agent is usually a human individual, but an information-processing machine can also be considered a cognitive agent [6].

This framework takes a cognitive perspective on business processes, in order to design the proper coordination, maintenance and use of operational knowledge in service organizations. The framework also presents a model of the knowledge and learning of agents in business processes inspired by related works in artificial intelligence.

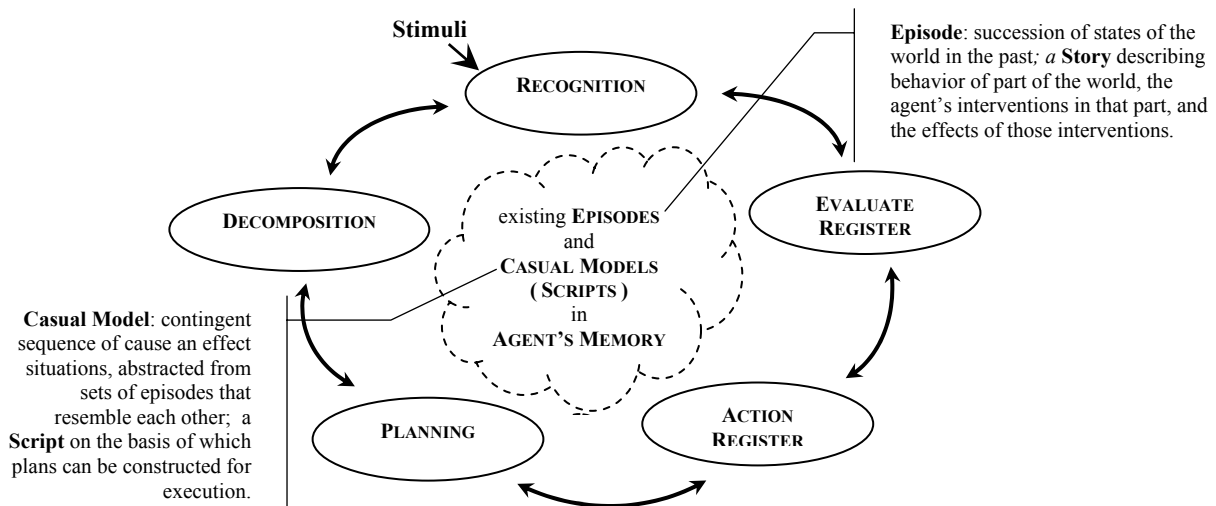


Figure 5: An illustration of Leijen-Baets's Cognitive Cycle

The framework's features allow the interpretation of organizations as goal-directed planners and the analysis of processes as a *Cognitive Cycle* composed of five phases: *recognition*, *decomposition*, *planning*, *action*, and *evaluation*. Figure 5 illustrates this cognitive cycle that can be recursively refined in all its five phases; that is, executing one phase may result in executing all five on a lower level of abstraction.

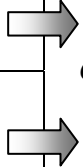
The process of recollection and construction of a casual model, is the first phase in solving a problem called *recognition* or *script selection*. During the second phase, named *decomposition*, an agent tries to construct a more accurate picture of the problem by collecting extra information; then *recognition* and *decomposition* alternate until the agent is sufficiently confident that the problem matches the model constructed. The next phase – *planning* – tries to find an adequate path of interventions by searching for previous intervention paths associated with similar situations. Remaining steps are *action register* and *evaluate register*, where the agent executes the plan and recursively finds sub-problems to solve. After executing the plan, the agent stores the new scenario along with its representation. An agent will usually not carry out the process sequentially but will interleave phases to increase efficiency.

This work declared that when defining and discussing cognitive process reengineering, it is necessary to account for a number of aspects: (1) cognitive process reengineering is the manipulation of scripts in cognitive agents; (2) reengineering serves the interest of stakeholders; and (3) reengineering always targets some performance criteria. So, the model proposed to add to a systematically search for opportunities to create new knowledge or exploit existing knowledge better.

4 A Framework for Identifying Cognitive Aspects in Reengineering Approaches

We have already said that software re-engineering is a work domain in which gathering and manipulating process knowledge from machines and human minds is a main issue and therefore, human cognition plays an essential role.

Motivated by the concepts in this area, Table 6 describes a framework for identifying cognitive aspects in reengineering approaches. Then, the framework is applied to the conventional reengineering approaches presented above.

<i>Framework Elements</i>	<i>Description</i>	
Goal, Focus & Basis/Background	What are the goal, focus and underlying basis/background of the approach?	 <i>Cognitive Aspects</i>
Reengineer, Other stakeholders & Reusability of existing knowledge	How helps the approach to the reengineer effort? Are other stakeholders involved? Is some kind of knowledge reusability considered by the approach?	
Table 6: Framework Elements for Identifying Cognitive Aspects in Reengineering Approaches		

Byrne’s Model

Goal, Focus & Basis/Background

Description: This approach provides a general model with its underlying foundation useful for examining reengineering issues such as the reengineering process and the reengineering strategies. A conceptual foundation composed of properties and principles that underlie reengineering method and assumptions about reengineering is used for: understanding re-engineering, proposing solutions to re-engineering problems and to explain the general model for software re-engineering. Byrne’s Model brings the individual foundation components together into a coherent form during a reengineering process.

Cognitive Aspects: From these elements it is possible to identify that knowledge recovery of useful ideas and insights that emerge from research and experience in software re-engineering have been taking into account to model a framework for understanding re-engineering.

Reengineer, Other stakeholders & Reusability of existing knowledge

Description: The conceptual foundation of the approach enables to clearly define, discuss, compare, and evaluate reengineering problems and methods. In this context, an experienced reengineer is required. In contrast, the involvement of many stakeholders is not specifically required, but it is important to say that the approach offers a good framework to all people involved.

Cognitive Aspects: The approach’s conceptual foundation guarantees that the stakeholders engaged in the reengineering process have a common discourse domain. It is evident that knowledge is a critical factor to take advantage of this framework, which can be used as an understanding platform for exploring reengineering concerns.

Jacobson’s Method

Goal, Focus & Basis/Background

Description: This approach provides a method to integrate the work of business reengineering, its processes and its information system. Background’s approach is from the 1993-1994 literature, when several important books on BPR were published, like [2] [9] [18], and from the OOSE Methodology with their Objectory Tool both describe in [19]. Jacobson’s reengineering approach obtains the Object Advantage in reengineering a business, by linking Object technology to the BPR model.

Cognitive Aspects: Evidences of a solid knowledge about business and its processes are inherent to the technique proposed by the approach. This work, with its logical framework and practical recommendation, details and gathers experiences on how to realize adaptive business enterprise.

Reengineer, Other stakeholders & Reusability of existing knowledge

Description: Jacobson's method provides reengineers a good and straightforward method with a useful object-oriented tool to develop a detailed and comprehensive model of the business. It also establishes that any stakeholder should have its own model of the company, which could be a segment or a simplified description of it: a view of the complete model according to the stakeholder category. A *Reuse Coordinator* involved in the re-engineering project and responsible for encouraging the reuse of already modelled infrastructure, is prescribed to evaluate the extent to which the project is undertaking reuse, as well as to investigate the potential reuse of the models.

Cognitive Aspects: Concepts taken into account by the approach focus the discussion on the company's adaptation from the stakeholders' business comprehension. Applicability is linked to customers' perception and, the model of the business should be designed to offer users the "usable" company. The approach forces reengineer to gain a complete understanding of the business under two visions: the "outside view" and the "inside view" developed by the Use-Case Model and the Object Model respectively. A reuse coordinator role is required because of his knowledge for applying reuse to the reengineering process.

Demeyer-Ducasse-Nierstrasz's Patterns

Goal, Focus & Basis/Background

Description: This approach provides a catalogue of patterns for reverse engineering and reengineering legacy systems. Main basis's approach is from traditional pattern literature, like Gamma Design Patterns [20], and from the European Industrial Research called FAMOOS (ESPRIT Project 21975 - Framework-based Approach for Mastering Object-Oriented Software Evolution) [21].

Cognitive Aspects: Pattern approaches founded their effectiveness in codifying and recording all available knowledge about historical experiences in developing software and, reengineering patterns are not the exception to this rule.

Reengineer, Other stakeholders & Reusability of existing knowledge

Description: Reengineering patterns help reengineers in diagnosing problems and identifying weakness which may hinder further development of the system, and aids in finding solutions which are more appropriate to the new requirements. Stakeholders' participation is required, for example, maintainers are involved in reverse engineering and users are involved in reengineering migration strategies. Reusability is a key concern in reverse engineering and reengineering legacy systems.

Cognitive Aspects: Reengineer cognition is improved by using recycled experiences in modifying legacy software. The catalogue pattern gives stakeholders valuable knowledge recovered from recurrent problems, optimal solutions to that problems and, experts' training and understanding in the reengineering field.

The SEI Horseshoe Model

Goal, Focus & Basis/Background

Description: This approach takes into account a software architecture perspective on reengineering, providing a foundation for transformations at different levels of reengineering analysis, especially for transformations at the architectural level. Byrne's Model [1] and, recent SEI's works in the field of Architectural Reconstruction and Architecture-Based Development (ABD) [4], compose the model's background.

Cognitive Aspects: The application of this conceptual model split human knowledge schema into three distinct levels: (1) the code level, (2) the function level and, (3) the conceptual level. The awareness is focused at this last concept level, which is proposed for initiating and propagating transformations during a system reengineering process.

Reengineer, Other stakeholders & Reusability of existing knowledge

Description: Because of their characteristics, the approach is basically oriented to reengineers. The SEI's model allows reengineers to distinguish different levels of reengineering analysis and describes the rich set of technical choices that reengineers make. Grounded in the technical underpinnings of the "horseshoe" model, a method called Option Analysis for Reengineering (OAR) [4] is also proposed for reengineering decision assistance.

Cognitive Aspects: It is easy to appreciate that the "horseshoe" model improves reengineer cognition by reasoning at different level of analysis, highlighting the architectural one. With this conceptual model as a starting point, the OAR method assists reengineers decisions during the reengineering work, as a structure and accessible vehicle for making knowledgeable choices in a variety of real world situations.

4.1 Discussion

Our purpose here is to establish a discussion on the inclusion of cognitive theories and models in re-engineering approaches based on the statement that human cognition has been always a decisive attribute when re-engineering work has to be done.

In first place, we propose to look at HASTI framework and their possible contributions to conventional reengineering approaches. Since HASTI can be used to support cognitive reengineering analysis, it is possible the application of this sort of analysis during BPR early stages of design. For example, using Byrne's model a reengineer can apply HASTI to assist their restructuring decisions at conceptual, requirement and design abstraction levels. He can also explore reengineering possibilities and, if it is necessary communicate these possibilities to other stakeholders involved, using the conceptual foundation common discourse domain.

Other example becomes clear form the SEI's "horseshoe" model, that is basically Byrne's model with a software architecture perspective. Because of this technical focus, the model does not assist reengineer in deciding on complex options regarding the future of their legacy systems, so the OAR method is provided. The model structure provided by HASTI can support this cognitive aspect too, assisting reengineers in inspecting and visualizing the results of potential cognitive arrangements, especially at the architectural level.

Now, let's make some considerations of what kind of support van Leijen-Baets's Framework can provide to BPR approaches. The work's purpose is, to use knowledge about how people think and act to design business processes support, which enhances the way people think and act. The framework defines a causal model, called "script", which describes the content of human experiential knowledge.

For example, in Jacobson's method, a reengineer is forced to develop a users' view of the business as a key concern for company's adaptation. Then, with the van Leijen-Baets's cognitive cycle might be possible to take a cognitive perspective on business processes turning knowledge from users' perception about the company into scripts for reengineering the business's usability. As another example, by using Demeyer-Ducasse-Nierstrasz's Patterns a reengineer is provided by recycled human experiences in modifying legacy software. Again, by storing this valuable catalogue of patterns into scripts, the cognitive cycle can be use as an instrument to manipulate all knowledge.

5 Conclusions and future work

In this paper, we have described traditional as well as cognitive-based approaches to business reengineering. We have also presented and applied a framework for identifying cognitive aspects in traditional BPR approaches. This application has resulted in a discussion about how cognitive science can support BPR approaches.

Our interest in studying and investigating cognitive models and their relation with BPR approaches, became from a recent work we have developed to carry out reengineering projects for personalizing existing web applications [11]. One of the current challenges in cognitive field is how to transform cognitive issues into a dynamical knowledge instrument for process improvement. As current Web based systems are a good example of software with knowledge-intensive process needs, our future work will apply cognitive aspects to improve web re-engineering processes.

References

1. Byrne, E. "A Conceptual Foundation for Software Re-engineering" In Proceedings of the International Conference on Software Maintenance, 1992
2. Hammer, M. and Champy, J. "Reengineering the Corporation: A Manifesto for Business Revolution" NY: HarperCollins, 1993
3. Jacobson, I. ; Ericsson, M. and Jacobson, A. "The Object Advantage: Business Process Reengineering with Object Technology" Addison-Wesley, 1994
4. SEI Technical Work Page, "The SEI Horseshoe Model" Carnegie Mellon University, Software Engineering Institute (SEI), 2003 <available at http://www.sei.cmu.edu/reengineering/horseshoe_model.html>
5. Walenstein, A. "HASTI: A Lightweight Framework for Cognitive Reengineering Analysis" In Proceedings of the 14th Psychology of Programmers Workshop, Brunel University, 2002
6. van Leijen, H. and Baets, W. "A Cognitive Framework for Reengineering Knowledge-Intensive Processes" In Proceedings of the 36th Hawaii International Conference on System Sciences, 2003
7. Chikofsky, E. and Cross II, J. "Reverse Engineering and Design Recovery: A Taxonomy" IEEE Software, Vol. 7, N°1, pp. 13-17, 1990
8. Jacobson, I. and Lindström, F. "Reengineering of Old Systems to an Object-Oriented Architecture" In Proceedings of OOPSLA'91, Phoenix, pp. 240-50, 1991
9. Davenport, T. "Process Innovation, Reengineering Work through Information Technology" Boston, MA: Harvard Business School Press, 1993
10. Business Architects Page "What is Reengineering?", 2000 <available at <http://www.businessarch.com/reengineering.htm>>
11. Martín, A. "Personalization of Web Applications: A Reengineering Approach" Master Thesis directed by Dr. Gustavo Rossi, Master in Software Engineering Post Grade, UNLP, La Plata, Argentina, June 2003 <available at <http://journal.info.unlp.edu.ar/postgrado/magister/ingsoft/Tesis.html/Martin.zip>>
12. Walenstein, A. "Cognitive Support in Software Engineering Tools: A Distributed Cognition Framework" Thesis for the Degree of Doctor of Philosophy in School of Computing Science, Simon Fraser University, May 2002.
13. Weicher, M. ; Chu, W. ; Ching Lin, W. ; Le, V. and Yu, D. "Business Process Reengineering: Analysis and Recommendations" Group of MBA and MS students at Baruch College, City University of New York, 1995 <available at <http://www.netlib.com/bpr1.htm>>
14. SEI Technical Work Page, "Reengineering" Carnegie Mellon University, Software Engineering Institute (SEI), 2003 <available at <http://www.sei.cmu.edu/reengineering/index.html>>
15. Demeyer, S. ; Ducasse, S. and Nierstrasz, O. "Object-Oriented Reengineering Patterns" (OORP), Editorial Morgan Kaufmann & Dpunkt, 2002
16. Craig, I. ; "From Blackboards To Agents" In Online Proceeding of the VIM Project Spring Workshop on Collaboration Between Human and Artificial Societies, 1997
17. Rassmussen, J.; "Skills, Rules, Knowledge: Signals, Signs, and Symbols and other distinctions in Human Performance Models" IEEE Transactions on Systems, man, and Cybernetics, 13(3):257-267, 1983
18. Martin, J.; "Enterprise Engineering -The Key of Corporate Survival" Vol I-V, UK: Savant Institute, 1994
19. Jacobson, I.; Christerson, P.; Jonsson, P. and Övergaard, G. "Object-Oriented Software Engineering - A Use Case Driven Approach" MA: Addison-Wesley; New York: ACM Press, 1992
20. Gamma, E.; Helm, R.; Jonson, R. and Vlissides, J. "Design Patterns: Elements of Reusable Object-Oriented Software" Editorial Addison-Wesley, 1995
21. FAMOOS European Industrial Research Project, 1996-2000 < available at iamwww.unibe.ch/~famoos >