Formal models of business process reengineering for design and design validation

by

Katayoun Atefi

TR-EIL-97-1

Enterprise Integration Laboratory
Department of Mechanical & Industrial Engineering
University of Toronto
4 Taddle Creek Road
Toronto, ON M5S 3G9

Tel: +1(416) 978-6823
Fax: +1(416) 971-2479
Internet: eil@ie.utoronto.ca

Acknowledgments

I would like to express my sincere gratitude and appreciation to my supervisor Professor Mark S. Fox for his encouragement, support, advice and for all I have learned from him, during my M.A.Sc. study at University of Toronto.

My special thanks to Michael Gruninger for his valuable time, great conversations on heuristics and ontologies and his valuable comments that have greatly improved this thesis.

I would like to thank Professor Michael Carter for his help. I thank Professor Beno Benhabib for his support. In addition, I thank the following staff members for all of their assistance: Norma Dotto, Alison Donald, Brenda Fung, and Louisa Kung. I wish to thank our system administrators Oscar del Rio and Evan Sidoriak for the help they have given me.

This research was supported, in part, by the Department of Mechanical and Industrial Engineering, University of Toronto, the Manufacturing Research Corporation of Ontario, and Natural Science and Engineering Research Council.

I would like to thank my husband, Jeff Sansome, for all of his encouragement and understanding. Finally, I wish to thank my father, mother, sisters and brothers in law for all of their support.

Abstract

Process design is an essential step in business process reengineering. Improving process design and searching for new process solutions are mostly based on success stories and heuristics. Since the underlying reasons of heuristics are often ambiguous, the results of their application are unpredictable. Formal models need to be developed to identify the foundation of the expertise. Towards this end, the thesis develops three formal models.

- The analytical model of agent setup time is used to demonstrate the effect of some process and agent assignment strategies on the agent setup time. The model allows us to describe the rationale of many heuristics which recommend assigning a process or some of its tasks to an agent or to a team to increase the process efficiency.
- The first order logic (FOL) model of agent/activity design strategies; i.e. a group of agent assignment strategies that can improve agent setup time. Given a process, a reasoning system can explore a variety of agent assignment designs and use this logical model to find the ones which lead to minimal agent setup time.
- The FOL model of design validation expertise defines a total of five principles with respect to information flow, case management and agent constraints. Given a process, the model can be employed to find the situations where these principles are violated.

The logical models make use of generic representation of activity, agent, information as well as the generic representation of what is the truth value of a property at different time points and how this value changes. Defining the meaning of each employed term, the logical models address the problem of ambiguity in BPR. They are also operational. We integrated them into a software tool; i.e. Process Integration advisor. By automatically generating alternative agent assignments that achieve minimal agent setup time and by informing the process designer of some problems in the process structure, the advisor supports analyzing a given process.

Chapter 1 Introduction 1

Chapter 2 - Part 1 Review of process design heuristics 6

2.1 Introduction 6

2.2 Reengineering the Corporation 11

2.2.1 Several jobs are combined into one 11

2.2.2 Hybrid centralized/decentralized operations are prevalent 11

2.2.3 Work is performed where it makes the most sense 12

2.2.4 The steps in the process are performed in a natural order 12

2.2.5 Reconciliation is minimized 12

2.2.6 Processes have multiple versions 13

2.2.7 A Case Manager provides a single point of contact 14

2.2.8 Workers make decisions 14

2.2.9 Checks and controls are reduced 14

2.3 Process Innovation 15

2.3.1 Order management processes 15

2.3.1.1 Case manager 15
2.3.1.2 Order segmentation 16
2.3.1.3 Customer participation 16
2.3.1.4 Real-time process execution 16
2.3.1.5 Parallel processes 16
2.3.1.6 Process partnerships 17

2.3.2 Other process types 17

2.3.2.1 Marketing processes 17
2.3.2.2 Service processes 17
2.3.2.3 Research processes 18
2.3.2.4 Engineering and design processes 19
2.3.2.5 Manufacturing processes 19
2.3.2.6 Logistical processes 20

2.4 Don’t Automate, Obliterate 21

2.5 Business Process Improvement 22

2.6 Other authors 25

2.6.1 Methods to Help Reengineer Your Company for Improved Agility [Ligus 93] 25

2.6.2 Business Re-engineering; a Strategy-driven Approach [Talwar 93] 26

2.6.3 Simple as ABC, What on Earth is Business Process Reengineering? [Booth 94] [Booth 95] 27

2.6.4 Useful hints [Miller 95] 28

2.6.5 Principles of Reengineering [Klein 95] 29

2.6.6 How to Make Reengineering Truly Effective? [Gilmore 95] 30

2.7 Conclusion 30

2.7.1 Heuristics; their positive aspects and limitations 30

2.7.2 Emerging themes from the reviewed heuristics 32

2.7.2.1 Agent assignments; the focus of chapters 3 and 4 of this thesis 32
2.7.2.2 Case manager, the focus of section 5.2 33
2.7.2.3 Concurrency in information intensive processes 34

Chapter 2- Part 2 Tools 36

2.8 Classification 36

2.9 Discontinuous Transformations (DT) 40

2.9.1 Review 41

2.10 Conclusion 42

Chapter 3 Analytical model of agent setup time 45

3.1 Agent setup time model 46

3.2 Manufacturing process strategies 48

3.2.1 Batch like orders 50

3.2.2 Transfer line 50

3.2.3 Common components 50

3.2.4 Standard interfaces 51

3.2.5 Computer controlled equipment 51

3.3 Agent/activity design strategies 52

3.3.1 Assign one agent to perform activities Aci and Acj 53

3.3.2 Assign an agent with the help of a computer program to perform activities Aci and Acj 54

3.3.3 Assign a team to perform activities Aci and Ac_j 55

3.3.3.1 Assign one agent to perform activities Acj-1 and Acj-2 57
3.3.3.2 Agent/activity design strategies and the issue of assigned agent 58

3.4 Conclusion and summary 60

Chapter 4 Formal model of agent/activity design strategies 63

4.1 Formalization methodology 64

4.2 TOVE project 67

4.3 Constructing the logical model of agent/activity design strategies 67

4.3.1 Motivating scenario 67

4.3.2 Informal competency question 69

4.3.2.1 Expressing the question, using FOL 70
4.3.2.2 Tailoring the question, with respect to TOVE's definition of activity 71
4.3.2.3 Consistent definitions at various levels of detail 72

4.3.3 Terminology 74

4.3.4 Axioms 76

4.3.5 Formal competency question 81

4.4 Extending the model 82

4.5 Generalization of the competency question 84

4.6 Summary 85

Chapter 5 Design validation model 87

5.1 Dangling information 88

5.1.1 Motivating scenario and informal competency question 88

5.1.2 Terminology and axioms 89

5.1.3 Formal competency question 90

5.2 Case management 90

5.2.1 Motivating scenario 90

5.2.2 Informal competency questions 92

5.2.2.1 Temporal projection 92
5.2.2.2 Agent constraints 93
5.2.2.3 Last version of informal competency questions 95

5.2.3 Terminology 96

5.2.4 Axioms 98

5.2.5 Formal competency questions 99

5.3 Changeable agent assignments 100

5.3.1 Motivating scenario, informal and formal competency question 100

5.4 Summary 101

Chapter 6 Incorporating FOL models into a software tool 103

6.1 Implementation of agent/activity design strategies 104

6.1.1 Implementation technique 104

6.1.2 Algorithm 107

6.1.3 Prolog program 108

6.2 Pre-order Management Process (PMP) 111

6.2.1 An overview of PMP 111

6.2.2 PMP subactivities 113

6.2.2.1 Identify potential order 113

6.2.2.2 Collect and evaluate customer data 113
6.2.2.3 Select and assign “transaction manager” 114
6.2.2.4 Evaluate, drop or select the pre-order 116

6.3 The Process Integration advisor 116

6.4 Analysis of PMP 117

6.4.1 Summary of results 118

6.4.2 Results 120

6.4.2.1 Dangling information 121
6.4.2.2 Case management 122
6.4.2.3 Changeable agent assignments 124
6.4.2.4 Agent/activity design strategies 125

6.5 Summary 127

Chapter 7 Summary and future work 129

7.1 Summary of the thesis 129

7.1.1 Demonstrate the heuristic nature of process design 130

7.1.2 Identify the dominant emerging theme from the heuristics 130

7.1.3 Create an analytical model of agent setup time 131

7.1.3.1 An overview of the analytical model of agent setup time 131
7.1.3.2 The application of our analytical model of agent setup time 132
7.1.3.3 The positive aspects of our agent setup time model 133
7.1.3.4 The limitation of our agent setup time model 133

7.1.4 Develop the logical model of agent/activity design strategies 135

7.1.4.1 The benefits of the logical model of agent/activity design strategies 135

7.1.5 Develop the design validation model 136

7.1.5.1 The benefits of our design validation model 136

7.1.6 Integrate the FOL models into the Process Integration advisor 138

7.1.6.1 The benefits of the Process Integration advisor 138

7.1.7 Demonstrate the application of our work 138

7.2 Future work 139

7.2.1 Analytical model of agent setup time 139

7.2.2 Ontologies 139
7.2.3 Implementation 139
7.2.4 Developing other formal models of BPR 140

References 141

Appendix A 151
A.1 Translating constraints from FOL into the PROLOG axioms 151

Appendix B 153
B.1 About Prolog 153
B.2 Files 153
B.2.1 Expertise 154
B.2.2 Temporal Projection 154
B.2.3 Pre-order Management Process (PMP) 154
B.2.4 Possible values for assigned agents 155
B.2.5 Demo 155
B.2.6 Other files 155
B.3 Files 157
B.3.1 all_thesis.log 157
B.3.2 thesis_ld_demo.pl 162
B.3.3 dng.pl 163
B.3.4 trc.pl 164
B.3.5 chg.pl 167
B.3.6 stp.pl 169

B.3.7 model.pl 175
B.3.8 main_def.pl 179
B.3.9 scenario.pl 185
B.3.10 driver1.pl 186
B.3.11 subactivity_def.pl 188
B.3.12 pmp_agents 189
B.3.13 thesis_Queries.txt 190

List of Tables

TABLE 1. Strategies and their effects on agent setup time 62

TABLE 2. Terminology for agent/activity design strategies 74

TABLE 3. Terminology for the “dangling information” 90

TABLE 4. Terminology for the “case management” 96

TABLE 5. Summary of the design validation model 102

TABLE 6. The possible outcomes and the activity enabled by each outcome 114

TABLE 7. Applying the Process Integration advisor to PMP; Summary of results 119

TABLE 8. The effects of process and agent assignment strategies on agent setup time 134

TABLE 9. Summary of the design validation model 137

List of Figures

FIGURE 1. The Components of BPR 10

FIGURE 2. Ag_i and Ag_j are the same. 68

FIGURE 3. Ag_i and Ag_j are a team. 69

FIGURE 4. Another subactivity such as Ac_k uses this information and Ag_k who performs Ac_k is the same as Ag_j. 69

FIGURE 5. An overview of PMP subactivities 112

Glossary

$\models$	*entails*
$\neg$	*not*
$\supset$	*implies*
$\equiv$	*if and only if*
$\forall$	*for all*
$\exists$	*there exists*
$\wedge$	*and*
$\vee$	*or*

Chapter 1

Introduction

The goal of this thesis is to develop formal models of business process reengineering (BPR) expertise; the ones that can demonstrate its underlying principles, extend the expertise to a larger group of users, and enable the consistent application of the practice across many enterprises. Towards this goal,

1. We develop an analytical model that highlights the various components of agent setup time. The model is used to describe the effects of different agent assignments and manufacturing process strategies on agent setup time.
2. We create a logical model that defines various “agent assignment” design strategies that improve agent setup time. A reasoning system can use the model to actually draw design alternatives that lead to minimal agent setup time.
3. We develop a logical model that can be employed to find the problems of an existing process design with respect to information flow, case management and agent constraints.
4. We incorporate the logical models into a software tool and use the tool to analyze a hypothetical process. The tool assists the designer in evaluating the process design.

Following, we describe the motivation and key aspects of the thesis.

The knowledge of business process reengineering (BPR) is informal and descriptive. BPR experts such as [Hammer et al. 93] and [Davenport 93] present a set of heuristics to help designers in the early stages of process design.

Reengineers like other human experts “are notoriously unreliable in explaining exactly what goes on in solving a problem” [Luger & Stubblefield 93]. Parts of their reasoning and analysis methodology might have become obvious or even automatic to them after awhile working in their field. For this reason, they often forget to mention the problem that their heuristic tries to solve, the technique that the heuristic uses to attack that problem, the conditions under which the employment of the technique would actually solve or improve the problem, and/or the meaning of the terminology used in describing the heuristic, (see the conclusion of chapter 2- part 1).

Heuristics with such characteristics are insufficient to make the BPR expertise available to a wider range of people. A software tool which is built based on these heuristics, would provide inconsistent solutions and thus would be incapable of supporting and expediting the process of enterprise design, (see the conclusion of chapter 2- part 2). Consequently the design of enterprises is still primarily dependent on the intuitive activity of consultants.

Our effort was motivated by these shortcomings in the expertise. Our goal is to transform process design knowledge into an engineering discipline where its principles can be applied in a consistent manner. Towards this goal:

1. We create an analytical model that highlights the components of agent setup time.

   The model allows us to explore the effect of some manufacturing process strategies and agent assignment strategies on the agent setup time, (see chapter 3). It reveals the underlying principles of a large group of heuristics. These heuristics recommend assigning the entire process or some of its activities to an agent or a team to improve the process efficiency through decreasing hand-offs, rework and error, (see the conclusion of chapter 2- part 1).

Agent setup exists if an agent requires some amount of preparation; e.g. understanding the information contained in the received transaction before performing the activity.

Manufacturing process strategies such as “Adam Smith’s division of labor principle”, “batch like orders”, and “producing different products that use the common components” structure the work so that an agent receives the transactions which are either identical or within a pre-defined range. In these situations the amount of context specific information (contained in the received transaction) is small and thus the agent setup time is trivial.

However, it might be the case that one agent receives different transactions. In this case, the agent needs to obtain a certain amount of context specific information, prior to performing the activity. For instance a designer needs to understand the product requirement before performing the design. In these situations, a group of agent assignment strategies can reduce the agent setup time. We refer to this group as “agent/activity design strategies”.

One of the agent/activity design strategies is “assigning one agent to the activity (Ac_i) which produces the information and the activity (Ac_j) which uses that information”. This strategy will decrease the agent setup time for the following reason. Performing Ac_i, the agent already assimilated the context specific information. S/he can perform Ac_j, without having to understand the information once again[1].

2. We develop a First Order Logic model of the “agent/activity design strategies”, (see chapter 4).

   The logical model has two roles. First, it is a means of expressing the strategies- i.e. through a set of logical axioms, the model defines the agent assignment strategies that can improve agent setup time. The second role comes from viewing the definition of strategies as a set of constraints. With respect to this view, the model actually defines a method of determining whether an arbitrary design satisfies these constraints. This means, a reasoning system can

1. Here, we assume that the agent does not forget what s/he has already obtained.

explore various design alternatives and use this logical model to find the answer(s) to the following question:

- Given a process, what is the redesigned process which satisfies the “agent/activity design strategies”, leading to minimal agent setup time?

The model is clear; i.e. the meaning of the each term, employed by the model, is formally defined. It is built upon the generic representation of activity, agent and information.

3. We develop a First Order Logic model that allows us to validate a process design with respect to information flow, case management and changing agents constraints, (see chapter 5). Expressing a total of five principles, the model enables a reasoning system to find the areas of design where these principles are violated. In specific, the model is capable of answering the following questions:

   - Given a set of activities, is there a piece of information which is produced by an activity and not used by any other activity?
   - Given a process, is there a time when no “case manager” exists for this process?
   - Given a process, is there a time when a “case manager” exists but s/he is unknown by the customer?
   - Given a process, is there a time when an agent should perform an activity in the process and the case manager of the process does not know about it?
   - Given a process, is there any activity which can change the assignment of an agent to a role or to an activity?

   The model is based on the generic representation of what is the truth value of a property at different time points and how this value changes, as well as the generic representation of agents, activities, roles, agent assignments and information.

4. We integrate all the logical models into a software tool, which is called Process Integration advisor (see chapter 6). The tool illustrates the practical use of our work in enterprise design.

Embedding the logical model of agent/activity design strategies, the advisor automatically generates alternative agent assignments that satisfy the “agent/activity design strategies”. Thus it can be employed to design or redesign processes, providing that the design perspective is improving the agent setup time.

Incorporating the design validation model, the advisor can be used to improve new designs and refine the design of existing processes.

Our effort in this line is one step towards automation of process design. We demonstrate this by applying the advisor to a hypothetical process. The Process Integration advisor which is an encapsulation of our logical models of expertise, enables us to identify the problems and strengths of this process and to provide a set of recommendations to improve its design.

In summary, we formalize some portions of BPR expertise. By formalization, we mean identification, formal representation and computer implementation of intuitions implicit in practice [FOX 94]. This promotes the growth and preservation of enterprise design expertise, provides an infrastructure for rapid and clear communication, facilitates the sharing and consistent use of the knowledge for various applications and users.

Chapter 2 - Part 1

Review of process design heuristics

In this part, we review the process design heuristics. We begin with an introduction to business process reengineering (BPR). Then we summarize the process design heuristics, presented by various authors. At last, we identify the common characteristics that prohibit these heuristics from serving as a foundation for a formal model of BPR.

2.1 Introduction

"Like a lot of fads, there's a good idea at the heart of it, but it's not capable of living up to all of the expectations created for it"[Davenport 96].

According to [Hammer et al. 93], two of the pioneers of the reengineering, the definition of business process reengineering is "the fundamental rethinking and radical redesign of business process to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service, and speed."

The concepts behind reengineering, are rooted in the other business improvement methods, such as sociotechnical approach, quality oriented methods, industrial engineering and competitive IT

[Davenport 93], [Earl 94], [Strassman 93]. Among these approaches, one of the most popular in the last decades was Quality movement. Like reengineering, it is a process centered approach[1] which recognizes the value of customer needs, employee empowerment and cultural change. Unlike reengineering, the focus of Quality efforts has always been on continuous and gradual enhancment of the existing processes. Only a few of Quality experts, e.g. Juran and Deming, encouraged “Radical process improvement” and in reality it was never practiced. Only recently, Quality experts, e.g. Harrington, spoke of the role of IT in process improvement [Davenport 93].

By the middle to late 1980s, some American companies recognized that under the intensified pressures of business environment, the gradual pace of improvement was insufficient. They understood the need for dramatic change in their business performance. These companies did not enhance their existing processes or automize them. They used IT to restructure some of their key processes. The improvement in those processes cost or execution time was dramatic. In the beginning of 1990s, the consultants who studied and worked with these companies, started to develop a new approach for business improvement. The approach has emerged under various names such as business process innovation [Davenport 93] and business process reengineering [Hammer et al. 93]. Because of the widespread acceptance of the word reengineering, it is used here.

There has been a proliferation of BPR literature in recent management and information technology literature. Various BPR methodologies are proposed, e.g. [Fitzgerald et al. 96], [Booth 94], [Ghani 96], [Klein 95], [Khoong 96]. These BPR methodologies vary in their components, the importance attached to the components, e.g. IT [Venkatraman 94], the relation and order among the components [Khoong 96], the type [Drew 94] and size [Guha 93], [Hale 96] of the industry, comprehensiveness and depth, etc. However there are some commonalities among them. Figure 1 on page 10 illustrates the common components of a BPR methodology and their relationship through an abstract influence diagram[2]. The diagram also shows the relationship among BPR

1. It looks at a set of activities that are designed to produce a specified output which is valuable for a customer.

heuristics[1] (indicated as highlighted ovals) with the other BPR components. These heuristics are rules of thumb that are based on the experiences of consultants and practitioners of BPR projects. These heuristics provide guidelines for designing and implementing enterprises. Some authors arrange them in groups [Davenport 93] [Hammer et al. 93]. The following groups can be identified:

1. **IT enabling roles heuristics.** These heuristics describe various ways in which information technology can enable or constrain new process designs. Understanding the various deployments of IT to improve processes, designers can come out with better designs. For instance, [Davenport 93] recognizes a number of key applications of IT for each type of process. He introduces automated design, simulation systems, tracking systems, decision analysis systems and interorganizational communication systems as the key applications of IT for product development processes.
2. **Process and scope selection heuristics.** The heuristics of this group describe various criteria to choose a process and its scope for reengineering. Some of these criteria are strategic relevancy, process health and manageability [Davenport 93].
3. **Organizational structure, systems and behavior heuristics.** These heuristics describe the employees educational strategies, management responsibilities and behavior, performance measures and compensation, organizational structure and culture that enable new process designs. For instance, it is recommended that compensation systems must be based on results, managers should have a coach like behavior, organizational structure should be flat, organiza-

2. An influence Diagram consists of three node types:

   Decision nodes. These nodes represent the decisions under the control of the decision maker; i.e. the one(s) who decide about how new processes and enterprise should be. Arrows entering such decision nodes show the information that is available at the time of the decision.

   Chance nodes. The quantities in these nodes are considered uncertain. Arrows entering chance nodes means the node is probablistically dependent on whatever is at the other end of arrow.

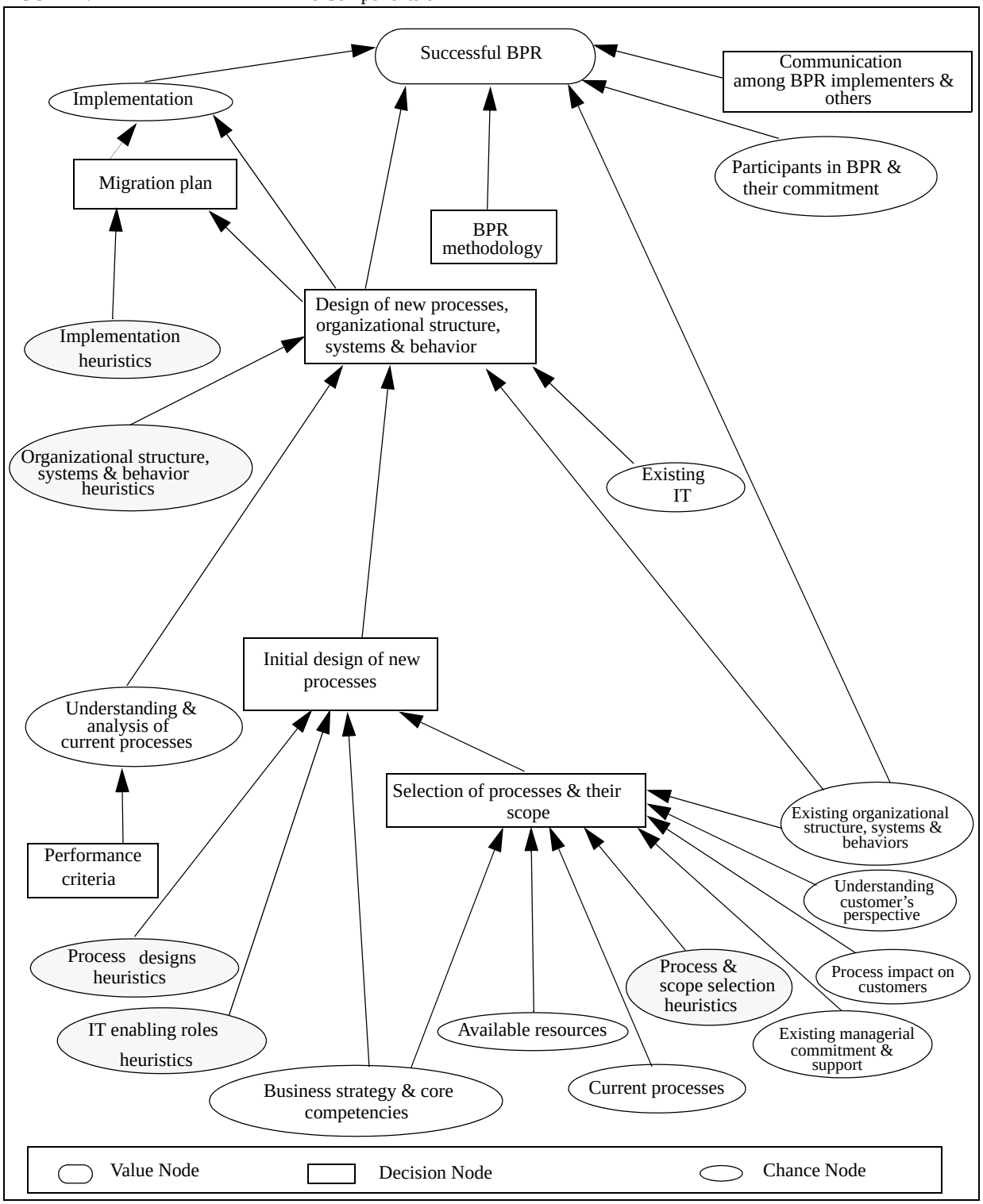
   Value node. This node represents the variable whose value must be determined or the question that must be answered.

1. A heuristic is “a rule of thumb, strategy, or trick used to improve the efficiency of a system which tries to discover the solutions of complex problems” [Slagle 71].

tions should be structured around the process teams, and workers should be multi dimensional and empowered [Hammer et al. 93].

4. **Implementation heuristics.** The intent of these heuristics is to prevent implementation pitfalls and to manage the required changes from as-is to the future situation. The examples of pitfalls are assigning someone who does not understand reengineering to lead the effort, applying bottom-up approach to reengineering, neglecting people's values and beliefs, emphasizing only process design and ignoring the changes that are triggered by the new process design [Hammer et al. 93].
5. **Process design heuristics.** There is no "one best way" [Mintzeburg 79] to design a process. Various factors such as business strategies, objectives(s), technology and the degree of its deployment, customers demands, the intensity of competition, policies and available resources affect the design. BPR experts more or less recognize the importance of these factors in designing new processes. For instance all of them agree that process design should be conducted in light of business strategies [Drew 94]. However in order to assist designers in developing process design ideas, they package the characteristics of typical reengineered processes. These packages are presented under different names, e.g. reengineering principles [Hammer et al. 93], innovation strategies for typical process types [Davenport 93]. We refer to them as "process design heuristics". In the next sections, we organize these heuristics based on various authors and summarize them. In conclusion, we describe why these heuristics can not provide a foundation for a reliable and consistent model of business process reengineering.

FIGURE 1. The Components of BPR



2.2 Reengineering the Corporation

Having more emphasis on design heuristics, Hammer and Champy [Hammer et al. 93], two of the reengineering pioneers, write:

"We have noticed striking similarities among the various re-engineered processes, similarities that transcend industry type and even the identity of the particular process."

The following sections review their heuristics.

2.2.1 Several jobs are combined into one

All the steps of a process should be compressed into one integrated job, performed by a single person. If this is not possible under some situations (e.g. various steps must be performed in different locations or one person can not learn all skills) then the process should be assigned to a team. The benefits of this heuristic include: eliminating hand-offs and their associated errors, delays and rework and reduced process administration overheads.

2.2.2 Hybrid centralized/decentralized operations are prevalent

Decentralization provides more flexibility and a better service for the customer, but at the price of redundancy, bureaucracy, and missed economies of scale. On the other hand, coordination and economies of scale are the benefits of centralization. Providing instant access to expertise and information, IT enables companies to operate as if their employees are self governed, while the companies still get the benefits of centralization, (i.e. coordination and economies of scale). For instance, the sales representatives of a company work independently. However, at the same time, they can have instant access to the information, maintained in the central office, and as important

use a software that prevent them from quoting prices or specifying delivery conditions that their company can not meet.

2.2.3 Work is performed where it makes the most sense

It is sometimes appropriate to shift a part of a process to the process customer or supplier. For instance, a company which is manufacturing and maintaining electronic equipment, assigns some of its repair activities which were previously performed by its technicians, to its customer. The company also stores some of the spare parts at the customer's site and manages the inventory there.

2.2.4 The steps in the process are performed in a natural order

The process steps should be delinearized. In a traditional process, the steps are performed in a linear order; i.e. one task does not start until the previous one is completely finished. The linearity among the tasks slows the work down. After reengineering, the process is delinearized and work is ordered in terms of *"what needs to follow what"*. The benefits are: 1) jobs are performed simultaneously and 2) since the elapsed time between the earlier and later process steps is reduced, the amount of rework due to a change in the later steps is decreased.

2.2.5 Reconciliation[1] is minimized

Reconciliation is another form of nonvalue-adding work. Reengineering reduces reconciliation by minimizing the number of external contact points in a process. Reconciliation is needed when inconsistent data is received. Cutting back the number of external contact points in a process, reduces the possibility of receiving inconsistent data which requires checking and matching.

1. Reconciliation means resolving the inconsistencies.

The example is the accounts payable process at Ford. Before reengineering, an employee received the purchased goods, wrote their description in a form (i.e. receiving document) and sent it to Ford's account payable department. The accounts payable department compared the original order, the invoice and the receiving document. If the items in these documents were the same then the department would issue the payment. After reengineering, as soon as the order is issued it is registered in a data base to which the employee who will receive the goods has access. When this employee receives the goods, s/he compares them with the registered order. If the goods match the order then s/he issues a check. This strategy eliminates the invoice which is one of the external contact points, and hence removes the necessity of matching the order with its invoice.

2.2.6 Processes have multiple versions

In traditional organizations, a process is designed to meet the requirements of its most difficult transaction. All the transaction types, regardless of their need or degree of complexity, are processed in the same manner. After reengineering, each process has multiple versions. Each version is designed to satisfy the requirements of a different transaction. A triage step is set at the beginning of a multi-version process. The role of this step is to determine to which version a received transaction should be assigned. For instance, IBM Credit has established three versions of the credit insurance process. One version deals with routine transactions and is performed by the customer with the help of a computer program, the other is for medium hard cases and is performed by one of IBM Credit's employee (called a deal structurer) and the last version which handles difficult transactions is performed by a deal structurer with the help from a team of specialists.

2.2.7 A Case Manager provides a single point of contact

Sometimes the process is too complex to be performed by an individual or a small team. In these cases, it is suggested to assign an individual as the "Case Manager" for the entire process. The "Case Manager" is responsible for answering the customer's questions and solving his/her prob-

lems. In order to do so, the “Case Manager” should have access to all of the information systems which the employees who perform the process use. Also s/he should have the ability to contact these employees for further assistance. In the customer’s eyes, it seems that the “Case Manager” performs the whole process, but this is not the case.

2.2.8 Workers make decisions

In a traditional process, at the point where a decision needs to be made, employees who perform the process are not permitted to make that decision on their own. Instead, they need to go up the managerial hierarchy for the decision. In a reengineered process, an individual or a team who is assigned to perform the process is also authorized to make decisions. In this way the process will benefit from *“fewer delays, lower overhead costs, better customer response, and greater empowerment for workers.”*

2.2.9 Checks and controls are reduced

Preventing employees from abusing processes, traditional organizations fill each process with checking and control activities. Establishing these non-value added control points along a process increases the process duration and cost. Reengineering separates checking and control steps from each individual process. Checks then are performed in an aggregate, random and deferred manner. For instance, in a typical purchasing process, the purchasing department checks the signature of the person requesting an item to ensure that person is authorized to acquire the goods in the dollar amount specified and to verify that the department’s budget is sufficient for the bill. After reengineering, this control activity is removed from the purchasing process. Instead, several instances of the purchasing process is controlled randomly at the end of the month. This strategy, however, *“more than compensates for any possible increase in abuse by dramatically lowering the costs and other encumbrances associated with the control itself.”*

2.3 Process Innovation

[Davenport 93] recognizes the importance of organizational and technological constraints in the heuristics implementation. He characterizes his heuristics based on process types, e.g. order management processes, marketing processes and so on. In this section, we review his heuristics for each process type.

2.3.1 Order management processes

2.3.1.1 Case manager

An individual or a team is assigned to perform the entire process. The level of empowerment of the case manager and the information s/he should have access to (e.g. production scheduling and pricing policies) must be decided.

The heuristic is very similar to 2.2.1. However, to some extent, it encompasses the concept discussed by heuristic 2.2.7.

Davenport attempts to provide some guidelines for the successful application of this heuristic.

> The concept of case management was practiced in manufacturing many years before the rise of reengineering, but few firms had positive results. “Assembly-line” model (i.e. breaking the work into operations and assigning each operation to an employee) seems to be more effective for routine tasks in manufacturing.
>
> On the contrary, service industries employed case management successfully. In services, “assembly-line” model leads to buffers (inbox and outbox), communication interface and consequently longer process duration.

2.3.1.2 Order segmentation

Companies categorize their orders by complexity. Straightforward orders can be processed by computer and the rest are processed by case managers. Similar to 2.2.6.

2.3.1.3 Customer participation

Parts of order management process such as entering, tracking, configuring, and/or scheduling the order are performed by the customers. Similar to 2.2.3.

2.3.1.4 Real-time process execution

At the highest level of customer service, order management demands real-time performance. For instance, price and ship commitments should be made when the customer places an order. This is usually possible only with computers that provide access to inventory databases and pricing algorithms. Similar to 2.2.2.

2.3.1.5 Parallel processes

In order management process, credit checking and financing are separated from the rest of the process and performed simultaneously. Similar to 2.2.4.

2.3.1.6 Process partnerships

Eliminating unnecessary transactions, exchanging work better suited to one partner than the other, or changing restocking or payments triggers. This strategy allows firms to concentrate on the processes of critical importance to their success. Similar to 2.2.3 and 2.2.5.

2.3.2 Other process types

For other types of processes, he did not itemize the heuristics. Instead, he described the common approaches that were taken to reengineer each process type in a text format. In this section, we highlight these approaches.

2.3.2.1 Marketing processes

1. Rapid evaluation of how advertising and promotion impact sales. This is possible by using point-of-sale information gathering technologies.
2. Understanding and taking advantages of buyer behavior, e.g. individualized magazine publishing.
3. Identification of exceptions to normal patterns in data by employing expert logic.
4. Close partnerships with advertising agencies, data collection and database marketing firms. This promotes faster flows of more useful marketing data to decision makers through joint participation in activities such as design of data collection methods, development of analysis tools, and even marketing of new tools and techniques.

2.3.2.2 Service processes

1. Providing fast service by using computer programs; e.g. insurance agents with laptop computer can deliver real-time quotes and hotel customers can checkin and checkout without visiting the registration desk. Similar to 2.2.2 and 2.3.1.3.
2. Individualization treatment of customers by having rapid access to the customer and order information before or just after the customer calls to place an order, asks for information, and so forth.
3. Controlling or at least monitoring of the factors that affect the service quality, e.g. Federal Express predicts incoming package volumes on the basis of a criteria such as the day of week and weather conditions to minimize unexpected factors that might delay package deliveries.

4. Separating the company's performance from its monitoring and control. For instance, service requests are received at central offices and then assigned to the company location that can best fulfil the request. An example is Pizza-Hot where the central office takes the customer order and assigns it to the geographically suited franchise for preparation and delivery. A centralized service enables companies to centrally monitor service quality and provides data that can help them better plan new locations.
5. Performing part of the process during the move towards the customer's site.

2.3.2.3 Research processes

1. Clear and measurable project objectives.
2. Rigorous communication throughout the organization and using a common vocabulary about research projects and their status.
3. Close ties with firm's strategic planning process.
4. Project management approach to manage the timing and duration of activities for which specialized resources will be needed.
5. Formal cross functional meetings.
6. Using computers in the field, scientists conduct research design and analysis in the field to reduce the number of failed experiments linked to local conditions.

2.3.2.4 Engineering and design processes

1. Concurrent engineering or parallel process flow to reduce cycle time. Similar to 2.2.4.
2. Facilitating design for manufacturability and cost by: 1) use of computer programs, 2) communication through cross functional teaming among designers and manufacturers, 3) formal design standards and data models that specify preferred design and component choices.
3. Relevant process interfaces between engineering, sales and manufacturing. The examples are:
    - Developing computer programs to help sales people understand how a change in the customer order affects the product cost and delivery date. Similar to 2.2.2.

- Pre-engineering of a set of component designs that can be combined into many different versions of products.

4. Reducing the number of changes in product development cycle.

2.3.2.5 Manufacturing processes

Process thinking have been applied to manufacturing processes for a long time. In this term, manufacturing processes are, on average, probably one decade ahead of service or customer facing processes. Most of the credit is due to the quality movements. Following is a number of common strategies that are employed by companies to restructure their manufacturing processes.

1. Switching from batch processes to a cell-based work flow.
2. Saving more time and money in manufacturing by use of equipment maintenance expert systems which diagnose a complex equipment malfunction and recommend corrective action.
3. Flexible production tools.
4. Greater functional integration between manufacturing, sales, marketing, engineering, and logistics.
5. Involve the provision of a higher level of service such as consulting, real-time commitments and arranging optimum financing for the customer.
6. Use of MRP and MRP II for production control and material management and structuring work so that teams build entire products rather than simple components. These strategies did not turn out to be effective in practice.
7. Better interfaces between manufacturing and engineering, manufacturing and logistics, manufacturing and sales. For instance, using information from sales to drive manufacturing.

2.3.2.6 Logistical processes

1. Rich flow of communication and clear understanding among the supply chain agents.
2. JIT practices.

3. Having fewer suppliers enables easier communication and management of supply chain processes.
4. Elimination of warehousing and finished goods inventory management by creating finished goods to fill customer orders and shipping completed goods to customers.
5. Parallel processing of ancillary activities, e.g. site preparation and credit checking, in the supply chain process. Similar to 2.3.1.5.
6. Close relationships with third parties. The examples are:
    - Shifting reordering and shelf management to suppliers. Similar to 2.3.1.6.
    - Shifting the incoming inspection and testing of components to their suppliers. Similar to 2.3.1.6.
    - Providing information about the most effective use of their products for customers.
    - Consolidation of purchased components from other third parties, kited and packed to suit customer requirements and JIT delivery.

2.4 Don’t Automate, Obliterate

These heuristics, presented by [Hammer 91], are similar to the ones that we reviewed in section 2.2.

1. Organize around outcomes, not tasks.

   One person performs all the steps in a process and his/her job is designed around an objective instead of a single task. Similar to 2.2.1.
2. Have those who use the output of process perform the process.

   The person or department who uses a product or service, can perform the process (or part of the process) that actually provides that product or service. For instance, by using expert systems, departments such as accounting can make their own purchases. Similar to 2.2.3.
3. Subsume information-processing work into the real work that produces the information.

A person or department who produces the information can as well processes it. For instance, the Ford's receiving department which produces the information about the goods received, processes this information instead of sending it to accounts payable. Similar to 2.2.3.

4. Treat geographically dispersed resources as though they were centralized.

   IT enables coordination among separate divisions or employees. For instance, in order to coordinate among its several purchasing units, the corporate office of Hewlett-Packard has established and maintained a data base on vendors and their performance. All purchasing units use this shared data base to issue their purchase orders. Similar to 2.2.2.

5. Link parallel activities instead of integrating their results.

   Rather than integrating the results of the activities when they are completely finished, use shared data bases and communication networks to coordinate these activities while they are in process.

6. Put the decision point where the work is performed, and build control into the process.

   Those who carry out the work should also monitor it and make decisions about it. Similar to 2.2.8.

7. Capture information once and at the source.

   A piece of information should not be collected repeatedly. It should be once collected and stored for all who need it. This will reduce delays, entry errors and overheads. Similar to 2.2.5.

2.5 Business Process Improvement

[Harrington 91] defined a detailed methodology for business process improvement with foundations in "total quality management". He described 12 heuristics to streamline a process. [Davenport 93] believes Harrington's approach is different than the reengineering's approach because:

- Harrington concentrates on step-by-step improvement of the existing process rather than challenging the initial process structure.
- Harrington considers the role of IT after a process has been improved.

Nevertheless, as we will see, some of the concepts behind Harrington's heuristics are very close to Davenport's and other reengineering experts'.

1. **Bureaucracy elimination.** Minimizing delays, red tape, documentation, reviews and approval.
2. **Value added assessment.** The recommendations to eliminate non-value added are:
   - Eliminating rework by removing the causes of the errors.
   - Eliminating movement of documents and information by combining operations, (similar to 2.2.1), moving people closer together, or automation.
   - Minimizing waiting times by combining operations, balancing work loads, or automation.
3. **Simplification; i.e. less tasks, stages and interdependencies.** Simplification can be achieved by:
   - Combining tasks, to remove duplication and/or fragmentation. Similar to 2.2.1.
   - Changing the orders of tasks, combining, or separating tasks, and even balancing the workload of different individuals to manage complex flows and bottlenecks.
   - Preparing more understandable materials for presentation, establishment of meeting protocols, and fewer meetings with less duration.
   - Combining similar or consecutive activities.
   - Reducing amount of handling by combining responsibilities or by substituting a call for mail.
   - Eliminating unused data.
   - Eliminating useless copies of reports and letters.
   - Refining standard reports.

4. **Process cycle time reduction.**
    - Serial versus parallel activities. Similar to 2.2.4.

      For instance instead of performing sequential reviews by design, manufacturing and purchasing departments, the documents can simultaneously be sent to the reviewers or reviewers can have mutual meetings.
    - Changing activity sequence to decrease the involved physical moving of documents.
    - Reducing interruption. The location of the agents who performed the critical activities should be in a quiet area. Someone else must answer their phones.
    - Improved timing of activities. For instance if the mail pickup is at 10:00 a.m., all outgoing mail should be processed before 9:45 a.m.
    - Location analysis. Where the activity is performed physically can have a strong effect on cycle time, labor cost, etc. "*As a general rule, the closer the process is located to the customer, the better.*" The benefits are economies of scale, stocking costs, equipment costs, and utilization considerations.
    - Providing working cells organized to fit a process in which a lot size of one is the production plan. Similar to 2.3.2.5- step 1.
    - Order the activities based on their priorities, communicate the result with the employees and follow-up if the priorities are met.
5. **Error proofing.** Make it difficult to commit an error. For instance, use a computer program that checks spelling.
6. **Upgrading.** Upgrade the process equipment and office layout and people skills.
7. **Simple language.**
    - Preparing the documents with respect to the comprehension level of the audience.
    - Using clear words and specifying the meaning if necessary.
    - Using flowchart to show the procedures that take more than 4 pages of description.

- Using acronyms when it is frequently used in the document. Defining the abbreviation that is used in a document.

8. **Forms.** Self explanatory forms, non-redundant information and well defined abbreviations.
9. **Standardization.** Adequate documentation is required to standardize the process.
10. **Supplier partnerships.**

    "All processes are highly dependent on people outside the process who provide input in the form of materials, information, and/or ideas." In this respect, the following questions should be asked:

    - Does the process really need the input or get more than its need?
    - Is the timing and entering point of the input correct?
    - Is the input received in the best possible format and required quality?

11. **Big picture improvement.** So far the focus was on gradual change. In order to bring a substantial change, the process regardless of existing organizational constraint should be redefined.
12. **Automation and/or mechanization.** Using information technology to automate the process.

2.6 Other authors

In this section, we list the heuristics proposed by other authors.

2.6.1 Methods to Help Reengineer Your Company for Improved Agility [Ligus 93]

1. Reduce the physical distance between supply points, production, assembly and the customer for the core products. Similar to 2.5- step 4- bullet 5.

2. Integrate processes and reduce setups using a zero based goal. Streamline the physical flow within the factory. Physically couple successive operations in the chain of work, remove non-value-adding functions, and induce velocity. Similar to 2.5- step 2.
3. Implement physical changes to place facilities close to sources of supply.
4. Form partnerships with fewer suppliers such that components can be delivered to satisfy real demand. Similar to 2.3.2.6- step 3 and step 6.
5. Create short, direct lines of distribution to make it very easy for customers to place an order and receive fast delivery.
6. Streamline and electronically link the information chain so that flow is direct-without interruptions and delays. Reduce business cycle times to the time it actually takes to efficiently process information.
7. Induce fast communications and decisions throughout the organization by physically clustering functions needed to complete business cycles quickly. Tear down physical walls that stand in the way of communications.
8. Recompose operational organizations with cells that address logical separations of business cycles, containing multi skilled members, trained to do everything in the cell. Allow cell leaders to be periodically chosen by cell members; give the members the responsibility for making 90 percent of the decisions. Employ effective use of automation, technology and techniques.

2.6.2 Business Re-engineering; a Strategy-driven Approach [Talwar 93]

1. Eliminating unnecessary activities and reducing the number of delays, e.g reviews, authorizations, inspections and hand-offs between departments. Similar to 2.5.
2. Minimizing the delays between processing stages by automating workflows.
3. Increasing flexibility by creating a multi-skilled workforce. Similar to 2.6.1- step 8.

4. Reducing duplication of effort and investment by forming stronger partnerships with customers and suppliers, sharing more key information and undertaking joint development activities. Similar to 2.3.1.6 and 2.3.2.6- step 6.
5. Improving internal communications by bringing different organizational functions together to speedup product and service development. Similar to 2.6.1- step 7.
6. Outsourcing activities which add no value but divert management time and energy.

2.6.3 Simple as ABC, What on Earth is Business Process Reengineering? [Booth 94] [Booth 95]

1. Integrate to achieve lead time compression by:
   - initially linking order entry with manufacturing and eventually linking design and operations.
   - extending the links into customers and suppliers. For instance, a customer could directly enter a design on the company's systems, and the production schedules (both internally and among suppliers) would be updated. Similar to 2.6.1- step 5.
2. Plan for concurrent marketing, manufacturing process and product design process so that a product which meets the needs of a market segment can be designed and produced quickly. This is achievable by decreasing the fragmentation on functional lines in the organization. Similar to 2.6.2- step 5.
3. Remove the fragmentation in the production process. Then remove the managerial hierarchy which was in place to manage the fragmented process. Similar to 2.2.1 and 2.2.8.
4. Make information accessible to staff so that they can perform their work independent of referral upwards to middle management. Similar to 2.2.2 and 2.2.8.
5. Plan for designed-in quality rather than inspected-in quality. This is achievable by having concurrency between product design and manufacturing process.

6. Reorganize so that one department or individual is responsible for the whole process to minimize departmental handovers and to ensure a clear accountability. Similar to 2.2.1.
7. Arrange concurrent teams or cells to provide quality and timeliness. Basic scheduling and quality control is handled within the team. Similar to 2.6.1- step 8.
8. Have a modular and reusable product design to allow customized features to be contained in a single part of the design.
9. Have a product structure that allows variety to be introduced at the end of the manufacturing process as opposed to the beginning.
10. Have non fragmented staff roles. Similar to 2.2.1.

2.6.4 Useful hints [Miller 95]

1. Eliminate bottle necks by speeding up the slowest activity in the process.
2. Reduce the number of steps, complexity levels, and people. Similar to 2.5.
3. Reduce defects to prevent rework.
4. Increase flexibility by envisioning the parameters of possible change when designing the process, using adaptable people and designing processes to accommodate future change.
5. Eliminate non-value added activities, assets, and costs. Similar to 2.5.
6. Decentralize unless there are compelling reasons such as economies of scale and critical resources to do otherwise. If you centralize make sure this does not comprise service, quality or flexibility.
7. Streamline, simplify, automate and integrate. Similar to 2.5.
8. Use cellular and self-directed work teams to handle an entire process. Similar to 2.2.1.

2.6.5 Principles of Reengineering [Klein 95]

1. Rethink the boundaries between your processes and those of your suppliers and customers and integrate them with their processes. Similar to 2.3.1.3 and 2.3.2.6- step 6.
2. Consider outsourcing a process if your costs are higher than that of an outsource vendor and if you add no more value than the outsource vendor would add to that process. Similar to 2.6.3-step 6.
3. Give more responsibility to the front line people and increase flexibility. This approach often leads to decentralization. However, providing shared databases, expert systems and so on, information technology makes it possible to decide on centralization or decentralization on the basis of what makes the most sense for the business. Similar to 2.2.2.
4. Consider segmenting process inputs and creating parallel process flows to simplify the process (similar to 2.2.6), or create entirely new products or services.
5. Resequence activities where possible to eliminate the need for separate subprocesses. For instance, Disney provided automated kiosks at which customers could prepay. This reduces the time spent in queues. Similar to 2.2.2.
6. Simplify interfaces and information flows. For instance, Loews Co. allows its customers to find out what movies are playing and their show times, order a ticket by phone, pay with a credit card, and then pickup their tickets at an ATM or special line in the theatre lobby. For $1 more they can reserve a seat. These innovations improve customer service and enable Loews to measure the true demand for various films, so they can better schedule their theatres and better select films for specific audiences. Similar to 2.3.1.3.

2.6.6 How to Make Reengineering Truly Effective? [Gilmore 95]

1. Design for Flexibility. Rather than designing a process in considerable detail, build it in away that it can change to meet the customer needs over time.

2.7 Conclusion

2.7.1 Heuristics; their positive aspects and limitations

Heuristics are useful at the starting point of process design. They identify various attributes of successful processes and enable companies to look for alternative ways of design. For instance, a company attempting to redesign its order management system, for example, might look at "A Case Manager provides a single point of contact" Hammer et al. 93, "Several jobs are combined into one" [Hammer et al. 93], or "customer participation" [Davenport 93].

However, in general, heuristics are ambiguous and unreliable. In the following, we explain these two characteristics.

1. **Heuristics are ambiguous.**
   - The benefits of a heuristic (the problems that it solves or improves) are not clearly stated. This is an impediment to understanding the applicability of the heuristic. For instance the benefits of having a multi-version process (2.2.6), customer participation (2.3.1.3) or combining similar or consecutive activities (2.5- step 3) are either not stated or muddled.
   - The solution technique that the heuristic recommends is vague.

     For instance, 2.4 states that work should be ordered in terms of *"what needs to follow what"* but does not explain how one can determine which activity should succeed the other one.

     Another example is 2.2.3 which suggests to shift the responsibility of performing an activity from one agent to another who is more "appropriate". The issue is: how can we decide that an agent is more "appropriate" than the other? In chapter 3 (section 3.3.3.2), we discuss the issue of appropriateness in more detail.
2. **Heuristics are unreliable.**

Using a scenario, the author describes some perspectives (such as the process cost or duration) that are improved by the use of the heuristic. The heuristic is able to improve these perspectives, but only under some conditions, pertaining to that specific scenario. These conditions are mostly not expressed. This approach will lead to an unreliable heuristic, in a sense that the heuristic might not provide the same benefits under a different scenario.

For instance, heuristic 2.2.1 suggests to assign the entire process to a person or a team to improve hand-offs, delays, reworks and administrative cost. It does not discuss under what conditions this suggestion would improve these aspects.

Regarding the same concept, [Davenport 93] (as mentioned in 2.3.1.1) specifies some abstract conditions:

> Whereas, assigning one person or a team to routine and structured tasks in manufacturing industry has not proven to be efficient, the idea has worked well for service industries in which fragmented roles lead to buffers, communication interfaces and longer process duration.

However, further elaboration is yet required to explain why the idea is successful for service industries and why it is inefficient for routine jobs. Chapter 3 elaborates this issue.

The above characteristics, ambiguity and unreliability, prohibit the heuristics to be consistently applied across various scenarios. The goal of this thesis is to transform process design expertise into an engineering discipline where its principles can be repeatedly applied in a consistent manner. We achieve this goal by taking the following approach.

We discover the foundation of a portion of design knowledge and develop a formal model of this foundation. The formal model will be composed of a terminology and the definitions for each of the terms- that every user can understand. Such a model will ensure the consistent and reliable application of the design principles across various enterprises. The approach which is called "*ontological*[1] *engineering*" [Fox 94], will be explained in chapter 4.

2.7.2 Emerging themes from the reviewed heuristics

The reviewed heuristics directly suggest or imply some general classes of change. Following, we identify three of them and specify the ones which are the focus of the thesis.

2.7.2.1 Agent assignments; the focus of chapters 3 and 4 of this thesis

A large number of heuristics propose different ways of assigning agents to perform activities. Their objective is to improve efficiency. This group includes the heuristics which suggest:

1. assigning an individual (with the help of computer program) or a team to perform a set of activities, or
2. shifting the responsibility of performing an activity from an individual or a group to another.

The examples are heuristics 2.2.1, 2.2.8, 2.3.1.1, 2.3.1.3, 2.3.1.6, 2.3.2.5- step 6, 2.3.2.6- step 6, 2.4- step 1, 2.4- step 2, 2.4- step 3, 2.4- step 6, 2.5- step 2, 2.5- step 3, 2.6.2- step 1.

Apart from being directly recommend by many heuristics, suggestions 1 and 2 seem to be a key feature of some of the other heuristics. For instance:

- **"Processes have multiple versions"** has two components, (see section 2.2.6):
  1. Breaking the process into different versions so that each version can be performed either by an individual or by a team.
  2. At the beginning of a multi-version process, there should be a step that assigns each received transaction to one of the process versions.

  As we can see, "assigning each version of the process to a team or an individual" has the key role in the first component.

1. An ontology is a formal description of entities, properties of entities, and relations among entities; it forms a shared terminology for the objects of interest in the domain, along with definitions for the meaning of each of the terms [Fox 94].

- **“Hybrid centralized/decentralized operations are prevalent”** describes a case in which a software system prevents the sales representatives from quoting prices that their company can not meet, (see section 2.2.2).

  In this example, the software actually enables one agent to perform the activity of stating a price and the activity of reviewing the price.

The general idea offered by this class of heuristics is the most dominant one. For this reason, in chapters 3 and 4 of this thesis, we direct our efforts towards it; we will identify the underlying principles of this class of heuristics and develop a formal model of them.

2.7.2.2 Case manager, the focus of section 5.2

In order to improve customer service, these heuristics are recommended:

- For each transaction, there should be a single agent to answer the customer queries. Such an agent is referred to as a “case manager” [Hammer et al. 93].
- Case managers should have instant access to all the information systems used by those agents who process the transaction.

The examples are heuristics 2.2.7 and 2.3.1.1. In chapter 5 (section 5.2), we develop a formal model that can evaluate the effectiveness of the “case manager” role in an existing design.

2.7.2.3 Concurrency in information intensive processes

The heuristics in this group deal with ordering activities to increase concurrency. They illustrate some processes that before re-engineering their activities were performed sequentially and after re-engineering, some of their activities are performed simultaneously. The examples are heuristics 2.2.4, 2.2.9, 2.3.1.4, 2.5- step 4.

The heuristics do not explain the criteria based on which they determine whether or not two activities can be performed concurrently. However, the supporting examples of these heuristics are

mostly information intensive processes. By looking through these examples, we infer that one of the determinant factors is “information dependency between the activities”; i.e. it might be possible to perform two activities concurrently, if one does not produce a transaction which contains some information used by the other. In this thesis, we will not target this group of heuristics.

Chapter 2- Part 2

Tools

Many tools have emerged to support enterprise design. In this part, we classify these tools, and review one of them that automates some process design heuristics.

2.8 Classification

Through implementation of an enterprise design many difficulties can occur. In order to ascertain the major difficulties and identify the key areas of research that can attack these problems, a survey in the UK was conducted [Weston 96]. Some important areas of research needs that were identified include:

- improved conceptualisation and analysis methods, coupled with improved business metrics
- improved support for ongoing business analysis and system development

To support these needs, the models and tools which assist enterprise design analysis are very important.

[Weston et al. 95] classifies tools with respect to the following dimensions:

- **Life-phase.** Tools are used in the strategic planning of an enterprise, conceptual design, detailed design, implementation and execution.
- **View (or perspective).** The focus of the tool is on the structure and behavior of an enterprise from a particular view point such as information.
- **Genericity.** Genericity can be evaluated on a continuum. At one end of the continuum, the tool is tailored to a single enterprise, while at the other end it can be generally applied to business processes.
- **Consistency.** Consistency must be maintained between different views and within each view with respect to the life-phase dimension.

[Gruninger 95a] characterizes computer-based BPR tools. His characterization has two roles: 1) it can be used to evaluate the existing tools for enterprise design, 2) it provides a set of requirements for BPR software. In general, the tools can be characterized by their:

- **Problem solving capability.** The problem solving capability of a tool can be identified by specifying the problem that the tool solves and the solution to the problem. This includes the definition of what is the appropriate input to each tool and what is the correct output.

  For a given tool, different reasoning tasks fall on different points in the spectrum of automation.

  At one end of the spectrum, there are tools which provide visualizations of the enterprise models that enable easier communication and provide comprehension of the enterprise and its problems. Enterprise knowledge is gathered, usually by structured interviewing from process owners and then represented graphically. These tools can support the analysis by facilitating understanding and communication for their users. Examples are Apache, Business Improvement Facility and RADitor, all briefly discussed in [Spurr et al. 94], Workflow Factory Product Information, Cosmo, Extend+BPR, and Optima! Express, all briefly discussed in [Business Process Reengineering Tool Repository 96], BSSM [Clegg et al. 96].

As we move along the spectrum, there are BPR tools that analyze a given enterprise model. The tool might evaluate models from a particular perspective. Examples are ISO 9000 Quality Advisor [Kim et al. 94] which deduces whether or not an enterprise is ISO 9000 compliant and Activity-based Costing Advisor [Tham et al. 94] which evaluates the cost associated with some set of activities. The tool might determine the value of some proposition at a point in the future, e.g. the quantity of a resource after performing a set of activities. It might provide guidance for the user; e.g. the reasons when a particular enterprise model fails to satisfy some property and what should be changed in the enterprise model so that it does satisfy the property.

Current simulation tools evaluate alternative models with respect to a particular behavior. Simulation allows the users to experiment and observe the effects of making parameter or structural changes on the process behavior. It helps in the selection of target process for the redesign stage, experimenting with different process alternatives before the final choice, testing the functionality and impact of the newly designed process before implementation, communication and employee training. However it is the users' responsibility to produce the set of design solutions.

Some current simulation tools are Business Design Facility, Caddie, Ithink, Object Management Workbench, Processwise Workbench, SES/WORKBENCH and Vensim, all discussed in [Spurr et al. 94], Dynamic modelling for reengineering organizations [Vredde et al. 96], Bonapart, Process Charter, Clear Process, Gensym's ReThink and DPA, all discussed in [Business Process Reengineering Tool Repository 96], Design/IDEF (Meta software Corp., a Cambridge, Mass. Company) [Arend 93], and GAMEVIEW software [Laakso et al. 95].

Some automated tools generate alternative solutions. However the evaluation is by the users. An example is Discontinuous Transformations [Wagner et al. 94].

In the most automated form of analysis, the tools perform automated design with particular properties.

- **Supporting enterprise models.** An enterprise model is a computational representation of the structure, processes, information, resources, goals, and constraints of an organizational system [Gruninger et al. 95b].

  In general, there are several views in an enterprise. [Hirshheim 86] contrasts several alternative views (office activities, office functions, office semantics, decision making, work roles, transactional, and language action) through which an enterprise can be conceived. Some views are more analytical while others are more interpretive.

  Ideally a rich enterprise model enables us to analyze the enterprise from the various views. However, not all of the views need to be modeled to enable an individual type of analysis. According to [Fox 93], the competence of an enterprise model can be evaluated by its degree of problem solving support; i.e. what questions can the representation answer or what tasks can it support?

  In addition to competency, [Fox 93] recognizes the following important issues concerning an enterprise model:

  Perspicuity: is the representation easily understood by the users?

  Granularity: does the representation support reasoning at various levels of abstraction and detail?

  Consistency: given the set of possible applications of the model, can the model's contents be precisely and rigorously defined so that its use is consistent across the enterprise?

  Extensibility: can new data items be added to it and the properties of existing data items be extended, without having to redesign the entire representation method?

- **Software functionality.** The issues such as the tool's ability to capture the model (i.e. formulate the model) and validate it (i.e. perform completeness and consistency checking), work with other tools, translate the utilized terminology to different users, integrate partial models into an integrated model of enterprise, and work with incomplete data are considered under

software functionality. These are the capabilities of the tools that are independent of the reasoning tasks, required for problem solving.

- **Visualization.** Visualization environment should be adequate for communicating the essential information provided by the tools.
- **The intended users.** Each class of users- external consultant, internal consultant, manager, employee- specifies a different set of requirements on tools' different characteristics such as modelling, analyzing, visualization, software functionality and implementation.

2.9 Discontinuous Transformations (DT)

In this section, we review a BPR tool to which their developers, [Wagner et al. 94], refer as DT.

There is a lack of tools that automatically identify design problems of a given process and/or generate solutions. In this thesis, we will develop a tool to which we refer as the Process Integration advisor, (see chapter 6). The Process Integration advisor encapsulates a portion of design expertise; given a process model, the advisor generates agent assignment alternatives which will improve the process agent setup time and finds the design problems with respect to information flow, case management and agent constraints.

Among the existing tools, we found only one tool, DT [Wagner et al. 94], that- like our advisor- proposes solutions to improve a given process design. For this reason, DT makes an interesting candidate for the review.

2.9.1 Review

DT's main objective has been defined as *"formalization and developing a rationale to the traditionally ad hoc process of performing BPR"* [Wagner et al. 94].

The input to DT is a process model. In this model, the process subactivities, work objects (defined as inputs to and outputs of the subactivities) and the current agents of these subactivities are specified. Subactivities are classified as tasks or decisions and the agents are classified as initiators, internal operators, recipients and external customers.

The output of DT is new agent assignments and/or new temporal relationships for the subactivities, on the basis of the following heuristics.

1. A subactivity which is performed by internal operator should be assigned to its initiator and if this is not possible to its recipient.
2. All the successive subactivities of the same type (i.e. task or decision) should be aggregated into one activity. If one of the current performers is an external customer then it is preferable to assign this aggregate activity to that external customer.
3. If two successive task and decision subactivities act on the same set of work objects then they should be aggregated.
4. A master coordinator should be assigned to the process.
5. The subactivities that have no or a small number of work objects in common, should be performed concurrently.

2.10 Conclusion

Formal models and tools that capture the process design expertise are scarce. Processes should often be redesigned to respond to their continuously changing environment. Design of new processes is complex and requires many skills. Certainly, one tool can not satisfy all the requirements of this iterative, ongoing and complex process. A variety of tools are required to support the design. The majority of existing BPR tools are based on simulation. The current simulation tools are useful; they allow users to observe the effects of making parameter changes on the process behavior. Nevertheless, they do not embed the process design expertise and thus are not capable

of automatically identifying the process design problems and generating alternative solutions. In order to expedite the process design, we need definitive tools that encapsulate the process design knowledge. It is important to note that the tools that simply implement ambiguous heuristics do not respond to this need. For instance, consider DT (see section 2.9.1). Even though, it contains five process design heuristics and generates alternative solutions, typically, its solutions are not understood or consistently interpreted by different users. This problem is due to the fact that DT lacks semantics; i.e. it employs some terms for which no meaning is stated. For instance, what is the meaning of the term “aggregation” in its second heuristic? Does the term “successive”, employed by the second and third heuristic, imply a causal relationship between the subactivities or a temporal relationship (i.e. before/after relationship)? On the other hand, the underlying reasons for DT’s heuristics are not clear. For instance, what problem will be solved by moving a subactivity to the initiator or to the recipient and why is it more desirable to move it to the initiator rather than to the recipient?

Our tool, the Process Integration advisor (see chapter 6), is a definitive tool which embeds a portion of process design expertise. Given a process, it automatically proposes alternative agent assignments that improve agent setup time and finds the design problems with respect to information flow, case management and agent constraints. This advisor is definitive because it is based on logical models with two important characteristics. First, these models provide the meaning for each of the employed terms. Second, they precisely state the problems that they can solve. These characteristics assure the consistent usage of the terms across different users (even the ones who are not used to that terminology), and give an opportunity to users to evaluate the tool’s appropriateness for their needs.

Chapter 3

Analytical model of agent setup time

The goal of this chapter is to develop an analytical model of agent setup time. The model highlights different components of agent setup time and allows us to explore various strategies which eliminate or improve some of these components.

The process design heuristics, reviewed in chapter 2- part 1, were often unreliable and/or ambiguous. Such heuristics cannot form the basis of a robust model of business process reengineering. In order to create a precise and reliable model of BPR, the underlying principles of these heuristics should be identified. To achieve this goal, we focus on a group of heuristics which suggest the assignment of a number of activities to an employee (with the help of a computer program) or to a team, combine some activities, or shift some of the responsibilities of one employee or department to another. We develop an analytical model which enables us to examine the effect of different strategies on agent setup time. The model and strategies (which will be described in the following pages) clarify the intent of this group of heuristics.

3.1 Agent setup time model

Let us consider two activities Ac_i and Ac_j which are respectively performed by two agents, Ag_i and Ag_j:

$\{Ac_i, Ag_i\}$ --- T_{ij} ---> $\{Ac_j, Ag_j\}$

where T_{ij} is the transaction from Ag_i to Ag_j that causes Ag_j to perform activity, Ac_j.

Agent setup exists if an agent requires some amount of preparation before the activity Ac_j can be performed.

One type of preparation is understanding the transaction, T_{ij}, on which the activity, Ac_j, is to be performed. For example the transaction is the product's requirement and the activity might be design. Understanding the transaction entails understanding the information provided in the product's requirement. It is often the case that a transaction contains information that is unnecessary (i.e. redundant and/or irrelevant) for the task to be performed. Nevertheless, the receiving agent does not know that the information is unnecessary until s/he assimilates it. Therefore we define our agent setup time model as follows:

$$PT_j(T_{ij}) = PT_j(Necessary(T_{ij})) + PT_j(Unnecessary(T_{ij})) \quad \textbf{(EQ 1)}$$

where $PT_j(T_{ij})$ is the agent setup time and $Necessary(T_{ij})/Unnecessary(T_{ij})$ is the subset of information in T_{ij} that is necessary/unnecessary to the activity from the performing agent's perspective. Obviously, one way of reducing setup time is to remove all $Unnecessary(T_{ij})$ from T_{ij}.

It is also the case, that a transaction may not provide all the information necessary for the agent to perform the activity. Therefore, there will be some amount of information gathering (e.g. identifying, locating and completing) that has to be performed prior to performing the activity. For instance, the transaction is the product's design and the activity is reviewing the design from a

manufacturability perspective, performed by a manufacturing engineer. Features of a product considered insignificant to the product's designer but crucial to manufacturing may not appear in the design. After receiving the design, the manufacturing engineer discovers that s/he needs more information to review the design. On the other hand, the designer might use a term that is unfamiliar or ambiguous for the manufacturing engineer who reviews the design. The manufacturing engineer needs to understand the term's definition before proceeding with the review.

We extend our model to include information gathering as follows:

$$PT_j(T_{ij}) = PT_j(Necessary(T_{ij})) + PT_j(Unnecessary(T_{ij})) + PT_j(Missing(T_{ij})) \quad \textbf{(EQ 2)}$$

This model is still incomplete; the amount of time that it takes an agent to prepare to perform a task may also include the time it takes to learn how to perform the task. For instance, the product design might include some electronic features. The manufacturing engineer who reviews the design from a manufacturability perspective may not be familiar with the electronic requirements contained in the transaction and will have to learn some standards before s/he can process the transaction's information correctly.

Let $SK(Ag_j, Ac_j)$ be the skill required by Agent *j* to perform Activity *j*. $SK(Ag_j, Ac_j)$ consists of the skill that already exists in Agent *j*, *SKe*, and the "new" skill that Agent *j* must acquire in order to perform the task, $SKn(Ag_j, Ac_j)$.

$$SK(Ag_j, Ac_j) = SKe(Ag_j, Ac_j) + SKn(Ag_j, Ac_j) \quad \textbf{(EQ 3)}$$

We can now refine the model to include this skill acquisition:

$$PT_j(T_{ij}) = PT_j(Necessary(T_{ij})) + PT_j(Unnecessary(T_{ij})) + PT_j(Missing(T_{ij})) + PT_j(SKn(Ag_j, Ac_j)) \quad \textbf{(EQ 4)}$$

EQ 4 presents our agent setup time model. The model allows us to explore different strategies that can reduce the agent setup time.

3.2 Manufacturing process strategies

In this section, we use the agent setup time model (as given below):

$$PT_j(T_{ij}) = PT_j(Necessary(T_{ij})) + PT_j(Unnecessary(T_{ij})) + PT_j(Missing(T_{ij})) + PT_j(SKn(Ag_j, Ac_j))$$

to explain the success of various manufacturing process methods. Let us use the following notation for the rest of this chapter:

Notation:

Agent setup time $PT_j(T_{ij})$ for strategy *stg* is denoted as $PT_j^{stg}(T_{ij})$. For instance, the agent setup time for the strategy "*Adam Smith's division of labor principle*" and the strategy *"Transfer line"* are represented by $PT_j^{Smith}(T_{ij})$ and $PT_j^{Transfer\ Line}(T_{ij})$ respectively.

Consider the Adam Smith's "division of labor principle" [Smith 1850] and the prototypical pin factory that he described:

> *One man draws out the wire, another straightens it, a third cuts it, a forth points it,..., and the important business of making a pin is, in this manner, divided into about eighteen distinct operations, which in some factories, are all performed by distinct hands, though in the others the same man will sometime perform two or three of them.*

Consistency and repeatability were the keys to the division of labor success. In the pin factory, since all the transactions (i.e. the first, second, third,..., nth pins) were the same, the worker did

not need to understand the information in every received transaction, separate the unnecessary, gather the missing information, or learn a new skill before he performed the task. Using the above notation we have:

$$PT_j^{Smith}(Necessary(T_{ij})) = 0 \quad \textbf{(EQ 5)}$$

$$PT_j^{Smith}(Unnecessary(T_{ij})) = 0 \quad \textbf{(EQ 6)}$$

$$PT_j^{Smith}(Missing(T_{ij})) = 0 \quad \textbf{(EQ 7)}$$

$$PT_j^{Smith}(SK_n(Ag_j,Ac_j)) = 0 \quad \textbf{(EQ 8)}$$

Thus we have:

$$PT_j^{Smith}(T_{ij}) = 0 \quad \textbf{(EQ 9)}$$

In summary, receiving identical transactions leads to a zero value for agent setup time.

In the following, we identify five strategies that minimize the agent setup time, $PT_j(T_{ij})$. All of them except the fifth one minimizes $PT_j(T_{ij})$ through minimizing part specific information. The fifth way allows a certain amount of variety in the received transactions. However since the scope of variety is already known, $PT_j(T_{ij})$ is very small. These strategies are:

1. Batch like orders
2. Transfer line
3. Common components
4. Standard interfaces
5. Computer controlled equipment

The following sections describe these strategies.

3.2.1 Batch like orders

Although agent setup time for changing from one batch to a different one might be greater than zero, the value of agent setup time for individual items within a batch is zero. The reason is, within each batch all the items are the same and consequently there is no part specific information that needs to be understood. Using the previous notation (presented in section 3.2), we will have:

$$PT_j^{Batch}(T_{ij}) = 0 \quad \textbf{(EQ 10)}$$

Equation 10 states that the value of agent setup time within a batch is zero.

3.2.2 Transfer line

Creating a transfer line that works on one part type only and setting up all the machines to work on that part type. For such a transfer line, there is no part specific information. Thus the value of agent setup time is zero. Using the previous notation (presented in section 3.2), we will get:

$$PT_j^{TransferLine}(T_{ij}) = 0 \quad \textbf{(EQ 11)}$$

3.2.3 Common components

Designing different products that use the common components and are differentiated at the end of the production process. Until the point that the products are differentiated, the received transactions are the same and therefore the agent setup time is zero. Using the previous notation (presented in section 3.2), we have:

$$PT_j^{Common}(T_{ij}) = 0 \quad \textbf{(EQ 12)}$$

3.2.4 Standard interfaces

Having standard interfaces between the components of assembly typed products. Standard interfaces greatly reduce the difference from one received transaction to another, minimize the part specific information required to join various components and thus lead to a value for setup time which is more than zero but small. Using the previous notation (presented in section 3.2), we will have:

$$PT_j^{StandardInterfaces}(T_{ij}) \cong 0 \quad \textbf{(EQ 13)}$$

3.2.5 Computer controlled equipment

Computer Controlled equipment is capable of producing various parts within a pre-defined set. For all work pieces inside the set, computer programs are already written, tested and stored in a microprocessor-based controller. In this situation, receiving incomplete and unnecessary part specific information is inconsequential. Therefore, using the previous notation (presented in section 3.2), we have:

$$PT_j^{ComputerControlledEquip}(Unnecessary(T_{ij})) = 0 \quad \textbf{(EQ 14)}$$

$$PT_j^{ComputerControlledEquip}(Missing(T_{ij})) = 0 \quad \textbf{(EQ 15)}$$

Since each part is different than the previous one, some time is required to process the part specific information, prior to the manufacturing operation. The part specific information processing time is greater than zero but not significant. Thus:

$$PT_j^{ComputerControlledEquip}(Necessary(T_{ij})) \cong 0 \quad \textbf{(EQ 16)}$$

If we assume that agents do not forget what they have learned then a multi tasking worker who has already learned the required skills for a specific set does not need to obtain new skills to produce the products inside the fixed limits of this set. Hence, we have:

$$PT_j^{ComputerControlledEquip}(SK_n(Ag_j,Ac_j)) \cong 0 \quad \textbf{(EQ 17)}$$

As a result, the value for agent setup time is more than zero but very small:

$$PT_j^{ComputerControlledEquip}(T_{ij}) \cong 0 \quad \textbf{(EQ 18)}$$

3.3 Agent/activity design strategies

Agents can be assigned to activities in many different ways. A subset of these ways can actually improve the agent setup time. We refer to this subset as “agent/activity design strategies”.

Within service processes, a certain amount of context specific information such as customer information, product requirement and part design is associated with each transaction, T_{ij}. The context specific information makes each transaction different than the other one. Since in these cases there is no repetition, the above strategies that rely on consistency can not decrease the agents’ setup time. Agents need to understand information, prior to performing their activities. In fact, the context specific information might lead to large agent setups; i.e. the ratio of preparation or setup time for each activity is much greater than a given percentage of processing time.

Given the agent setup time model,

$\{Ac_i, Ag_i\}$ --- T_{ij} ---> $\{Ac_j, Ag_j\}$

$PT_j(T_{ij}) = PT_j(Necessary(T_{ij})) + PT_j(Unnecessary(T_{ij})) + PT_j(Missing(T_{ij})) + PT_j(SKn(Ag_j, Ac_j))$

how should we modify agents assignments to reduce agents setup time?

In the following sections, we answer the above question.

3.3.1 Assign one agent to perform activities Ac_i and Ac_j

Let us use $\boxed{Ag_i = Ag_j}$ to denote the strategy stating that one agent performs both Ac_i and Ac_j and use $\boxed{Ag_i \neq Ag_j}$ to denote the strategy stating that two different agents perform Ac_i and Ac_j.

At first, consider a general case when the agents Ag_i and Ag_j are different. Since Ag_i (e.g. the agent who provides product requirement) might not understand the data requirements of Ag_j (e.g. the designer), the completeness and relevancy of the information in the transaction (produced by Ac_i) cannot be assured. Using the above symbols and the notation from section 3.2, we have:

$$PT_j^{Ag_i \neq Ag_j}(Necessary(T_{ij})) > 0 \quad \textbf{(EQ 19)}$$

$$PT_j^{Ag_i \neq Ag_j}(Unnecessary(T_{ij})) > 0 \quad \textbf{(EQ 20)}$$

$$PT_j^{Ag_i \neq Ag_j}(Missing(T_{ij})) > 0 \quad \textbf{(EQ 21)}$$

Now consider the case that one agent performs activities Ac_i and Ac_j. In this case, the agent knows what information is necessary or unnecessary to perform Ac_j and therefore s/he rarely provides unnecessary information or misses the necessary one.

Performing Ac_i, the agent already assimilates the relevant context specific information. Under the assumption that agents do not forget what they already absorbed, there is no need that the agent understands the information once again to perform Ac_j.

Thus we have:

$$PT_j^{Ag_i = Ag_j}(Necessary(T_{ij})) = 0 \quad \textbf{(EQ 22)}$$

$$PT_j^{Ag_i = Ag_j}(Unnecessary(T_{ij})) = 0 \quad \textbf{(EQ 23)}$$

$$PT_j^{Ag_i = Ag_j}(Missing(T_{ij})) = 0 \quad \textbf{(EQ 24)}$$

We assume the time required to learn new skills is the same, no matter that Ag_i and Ag_j are different agents or the same.

From the above discussion, we can deduce:

$$PT_j^{Ag_i = Ag_j} < PT_j^{Ag_i \neq Ag_j} \quad \textbf{(EQ 25)}$$

Equation 25 states that $PT_j^{Ag_i = Ag_j}(T_{ij})$ (i.e. the agent setup time when the agents of Ac_i and Ac_j are the same) is less than $PT_j^{Ag_i \neq Ag_j}(T_{ij})$ (i.e. the agent setup time when these agents are not the same).

3.3.2 Assign an agent with the help of a computer program to perform activities Ac_i and Ac_j

We use $\boxed{AgComp}$ to denote the strategy stating that one agent with the help of computer program performs both Ac_i and Ac_j.

One agent might not have all the required skills to perform both Ac_i and Ac_j. In this case, it might be possible that some or all of the skills required to perform Ac_i and/or Ac_j to be encapsulated in a computer program. A computer program can rapidly direct the performing agent to enter the complete and necessary information required to perform Ac_j and assist the agent to perform Ac_i/Ac_j. Therefore, using the above symbols and the notations from section 3.2 and section 3.3.1, we have:

$$(PT_j^{Ag_i = Ag_j}(Necessary(T_{ij})) = 0) < PT_j^{AgComp}(Necessary(T_{ij})) < PT_j^{Ag_i \neq Ag_j}(Necessary(T_{ij})) \quad \textbf{(EQ 26)}$$

$$(PT_j^{Ag_i = Ag_j}(Unnecessary(T_{ij})) = 0) < PT_j^{AgComp}(Unnecessary(T_{ij})) < PT_j^{Ag_i \neq Ag_j}(Unnecessary(T_{ij})) \quad \textbf{(EQ 27)}$$

$$(PT_j^{Ag_i = Ag_j}(Missing(T_{ij})) = 0) < PT_j^{AgComp}(Missing(T_{ij})) < PT_j^{Ag_i \neq Ag_j}(Missing(T_{ij})) \quad \textbf{(EQ 28)}$$

For instance, a human agent might not have the design skills and at the same time know all the requirements of a manufacturable design. However, the knowledge specific to design review (e.g. the requirements of various manufacturing facilities) can be incorporated into a computer program. Employing this knowledge, the program can direct the agent to enter the necessary design

information and notify him/her of the issues that might negatively impact the product's manufacturing.

With respect to the above discussion, and assuming that the time a person needs to learn a new skill is equal to the time which is required to extend the computer program to incorporate that skill, we can deduce:

$$PT_j^{Ag_i = Ag_j} < PT_j^{AgComp} < PT_j^{Ag_i \neq Ag_j} \quad \textbf{(EQ 29)}$$

Equation 29 states that:

- the agent setup time when one agent with the help of a computer program performs Ac_i and Ac_j is more than its value when one agent performs them.
- However, the agent setup time when one agent with the help of a computer program performs Ac_i and Ac_j is less than its value when two different agents perform them.

3.3.3 Assign a team to perform activities Ac_i and Acj

Let $\boxed{team(Ag_i, Ag_j)}$ denote the strategy stating that a team performs Ac_i and Ac_j, $\boxed{Ag_i \neq Ag_j}$ denote the strategy stating that two different agents who are not team members perform them and $\boxed{Ag_i = Ag_j}$ represent the strategy stating that the same agent performs both activities.

In general, Ag_i and Ag_j might have divergent goals. This divergence might discourage Ag_i to produce the complete and relevant information within the transaction, even if Ag_i knows what information is considered relevant and complete, from Ag_j's view.

Now, let's assume that Ag_i and Ag_j are team members; i.e. they have mutual goals. Under the assumption that team members are rationale, it is more likely that Ag_i tries to understand the effect of his/her transaction on the Ag_j's activity and produce the information to satisfy Ag_j's requirements. Therefore establishing a team relationship between Ag_i and Ag_j leads to providing less

unnecessary and incomplete information by Ag_i. Using the above symbols and the notation from section 3.2, we will have:

$$PT_j^{Ag_i = Ag_j}(Unnecessary(T_{ij})) < PT_j^{team(Ag_i, Ag_j)}(Unnecessary(T_{ij})) < PT_j^{Ag_i \neq Ag_j}(Unnecessary(T_{ij})) \quad \textbf{(EQ 30)}$$

$$PT_j^{Ag_i = Ag_j}(Missing(T_{ij})) < PT_j^{team(Ag_i, Ag_j)}(Missing(T_{ij})) < PT_j^{Ag_i \neq Ag_j}(Missing(T_{ij})) \quad \textbf{(EQ 31)}$$

For instance, when the manufacturing engineer and designer develop and share the mutual goal(s), it is more likely that the designer tries to understand the producibility aspects of the product in advance and comes out with the design that satisfies the requirements for the product's form, fit and function as well as the product's manufacturability characteristics.

On the other hand, a team can gather a variety of skills. Arranging a team so that a member who requires a new skill to perform the task can acquire it from the other member who already has that skill, leads to less learning time. This will lead to:

$$PT_j^{team(Ag_i, Ag_j)}(SK_n(Ag_j, Ac_j)) < PT_j^{Ag_i \neq Ag_j}(SK_n(Ag_j, Ac_j)) \quad \textbf{(EQ 32)}$$

From the above discussion, we obtain:

$$PT_j^{Ag_i = Ag_j} < PT_j^{team(Ag_i, Ag_j)} < PT_j^{Ag_i \neq Ag_j} \quad \textbf{(EQ 33)}$$

Equation 33 states that:

- The value of agent setup time when the performing agents of Ac_i and Ac_j are team members is more than its value when these agents are the same.
- However, the value of agent setup time when the performing agents of Ac_i and Ac_j are team members is still less than its value when they are not team members.

3.3.3.1 Assign one agent to perform activities Ac_{j-1} and Ac_{j-2}

Consider a situation where two different activities *(Ac_{j-1}* and *Ac_{j-2})* are caused by the same transaction, as shown below.

$\{Ac_i, Ag_j\}$ --- T_{ij-1} ---> $\{Ac_{j-1}, Ag_{j-1}\}$

$\{Ac_i, Ag_i\}$ --- T_{ij-2} ---> $\{Ac_{j-2}, Ag_{j-2}\}$

$T_{ij-1} = T_{ij-2}$

Since the activities Ac_{j-1} and Ac_{j-2} are different, their sets of necessary information, missing information and so on, which are produced by Ac_i and lead to their agent setup time, are not necessarily identical. However, due to the fact that Ag_{j-1} and Ag_{j-2} (the agents of the activities) need to know the information in the same transaction, these sets are definitely overlapping.

At first consider the case where Ag_{j-1} and Ag_{j-2} are different. The total agents' setup time for Ac_{j-1} and Ac_{j-2} is the sum of Ag_{j-1} setup time and Ag_{j-2} setup time.

Now consider the case that one agent is assigned to perform both Ac_{j-1} and Ac_{j-2}. Once this agent is prepared to perform Ac_{j-1}, s/he is also partly prepared to perform Ac_{j-2} and therefore the total agent setup time for Ac_{j-1} and Ac_{j-2} will decrease.

Hence, if $\boxed{TPT}$ represents the total agents setup time for activities Ac_{j-1} and Ac_{j-2}, if $\boxed{TPT^{Ag_{j-1} \neq Ag_{j-2}}}$ denotes *TPT* when the agents Ag_{j-1} and Ag_{j-2} are different, and if $\boxed{TPT^{Ag_{j-1} = Ag_{j-2}}}$ denotes *TPT* when Ag_{j-1} and Ag_{j-2} are identical, we have:

$$TPT^{Ag_{j-1} = Ag_{j-2}} < TPT^{Ag_{j-1} \neq Ag_{j-2}} \quad \textbf{(EQ 34)}$$

Equation 34 states that total agents' setup time when one agent performs Ac_{j-1} and Ac_{j-2} (two activities which use the information contained in the same transaction) is less than its value when different agents perform them.

For instance, each of the product supplier and the product customer separately designs a quality test to inspect the product's functionality. Let's assume the transaction that causes both activities is the product specification (i.e. prior to each of these test designs, the performing agent needs to know the information in product specification). The total value of agent setup time when one agent designs the supplier's test and the customer's test is considerably less than its value when two different agents perform these activities.

3.3.3.2 Agent/activity design strategies and the issue of assigned agent

Given the agent setup time model (EQ 4), agent/activity design strategies (sections 3.3.1, 3.3.2, 3.3.3, and 3.3.3.1) and a set of candidate agents such as CAg_i (i.e. the agent who currently performs Ac_i), CAg_j (i.e. the agent who currently performs Ac_j), and *John Smith* (i.e. an arbitrary agent),

Who is more appropriate to be assigned to perform both Ac_i and Ac_j?

The agent/activity design strategies do not answer this question; i.e. they are incapable of prioritizing one agent ahead of the other. Authors tried to provide rules of thumb to answer this question. For instance, consider the following heuristics:

- "Subsume information-processing work into the real work that produces the information" [Hammer 91] which implies to assign CAg_i to perform Ac_i and Ac_j.
- "Have those who use the output of the process perform the process" [Hammer 91] and "Customer participation" [Davenport 93] which state to assign the process customer to perform Ac_i and Ac_j.
- "Perform the work where it makes the most sense" [Hammer et al. 93] which suggests to assign Ac_i and Ac_j to CAg_i or the process customer, whoever is more appropriate.

However, as can be seen above, their attempts to provide an answer that always works were unsuccessful because there is no unique answer to the above question. It really all depends on the existing constraints in a scenario. Examples of constraints are: the agents' capability, resource

availability, agents' availability and policies. These constraints which differ from one scenario to another lead to various answers. See the following examples.

- Ac_i is providing the requirements that a new product should satisfy. Ac_j is processing the requirements; i.e. finding the product suppliers, choosing among them and issuing the order to purchase the product from the selected supplier.

  The non-relaxable constraint is: "the only agent who is capable of providing the product's requirements (i.e. perform Ac_i) is the one who will actually use that product.

  Based on the constraint, we can infer that the performing agent for both of these activities is the user of the product. Such a scenario might have been a driver for the heuristic "Subsume information-processing work into the real work that produces the information". As we will see in the next example, a different scenario might lead to a different suggestion.

- Consider the two activities of issuing an order according to the current inventory level, Ac_i, and manufacturing the goods to fill that order, Ac_j. Assume that there is a constraint stating that due to financial constraints and available manufacturing capabilities, the agent CAg_i who currently issues the order can not produce the product.

  This constraint implies that CAg_i can not be assigned to perform both activities. We should look for a candidate who is capable of issuing and manufacturing the order. If CAg_j (the agent who currently manufactures the order) has the required skills and resources to issue the order, then CAg_j can be a considered as a potential answer[1] to the question[2].

- One of our previous examples consists of two activities: design and reviewing the design from manufacturability perspectives. The strategy we want to apply is that one agent with help of computer performs both of the activities. The computer program mostly incorporates the required skills to perform the review (as opposed the skills to perform the design). The agent

1. It is a potential answer, i.e. it might be rejected due to the other constraints such as "from a controlling point of view, the one who issues the order and the one who manufactures the goods to fill that order can not be the same".

2. An example is Wal-Mart and its supplier Procter & Gamble where Procter & Gamble checks the Wal-Mart's inventory level for its products, uses this data to schedule the manufacturing of those products, and even issues the order [Davenport 93].

and computer program together should be capable to perform both of the activities. This entails that the assigned agent should definitely have the design expertise.

3.4 Conclusion and summary

Through the agent setup time model, we pointed out the important components of agent setup time. We discussed different strategies that can reduce or remove some of these components. As we saw in the previous chapter, several authors presented these strategies in the form of various heuristics. However, none of them describe the conditions under which the strategies are applicable. Although [Davenport 93] mentioned that the agent/activity design strategies seemed to be more effective for service than manufacturing industry, he did not discuss the rationale. Many of [Hammer et al. 93] heuristics[1] are based on these strategies but muddled within the context of scenarios.

Table 1 on page 61 summarizes our discussion. The strategies are listed in the first column. The agent setup time and its components are presented in the first row, based on the same notations we employed in the model. The definition of each notation is given in the table footnote. Each cell presents the effect of a strategy on the agent setup time or one of its components; the strategy can either “eliminate”, “reduce” the agent setup time (or one of its components) or has “no effect” on it. In the next chapter, we develop a First Order Logic model of agent/activity design strategies.

1. The heuristics such as “Several jobs are combined into one”, “Work is performed where it makes the most sense”, and “Workers make decisions”.

TABLE 1. Strategies and their effects on agent setup time

Strategy	**$PT_j(T_{ij})$ [a]**	**$PT_j(Necessary(T_{ij})$ [b]**	**$PT_j(Unnecessary(T_{ij}))$ [c]**	**$PT_j(Missing(T_{ij}))$ [d]**	**$PT_j(SKn(Ag_j,Ac_j))$ [e]**
Within a batch order	eliminate	eliminate	eliminate	eliminate	eliminate
Transfer line	eliminate	eliminate	eliminate	eliminate	eliminate
Common components	reduce	reduce	reduce	reduce	reduce
Standard interfaces	reduce	reduce	reduce	reduce	reduce
Computer controlled equipment	reduce	reduce	reduce	reduce	reduce
Assign one agent to perform Ac_i and Ac_j [f]	reduce	eliminate	eliminate	eliminate	no effect
Assign an agent with the help of a computer program to perform Ac_i and Ac_j [f]	reduce	reduce	reduce	reduce	no effect
Assign a team to perform activities Ac_i and Ac_j [f]	reduce	no effect	reduce	reduce	reduce
Assign an agent to perform Ac_{j-1} and Ac_{j-2} [g]	reduce	reduce	reduce	reduce	reduce

a. Agent setup time

b. The required time to understand necessary information.

c. The required time to separate unnecessary information.

d. The required time to gather missing information.

e. The required time to learn a new skill.

f. Ac_j is the activity which uses the information contained in the transaction, caused by the activity Ac_i.

g. Ac_{j-1} and Ac_{j-2} are two activities that use the information contained in the same transaction.

Chapter 4

Formal model of agent/activity design strategies

In this chapter, we develop a First Order Logic (FOL) model of agent/activity design strategies. It is important to note that this FOL model of agent/activity design strategies and the analytical model of the agent setup time (presented in the previous chapter) serve different purposes.

We used the latter to convey the effect of the agent/activity design strategies on agent setup time. Through this process, although we described the strategies, we did not formally represent them.

The First Order Logic (FOL) model allows us to formally express these strategies through a set of axioms. The axioms are important for two reasons. First, they provide a precise definition of the strategies. Second, they can be viewed as a set of constraints. With regard to this view, a reasoning system can use the axioms to generate alternative process designs that solve the following problem:

> Given a process, what is the redesigned process which satisfies the “agent/activity design strategies”, leading to minimal agent setup time?

In chapter 6, we employ the Prolog’s reasoning system[1] to demonstrate how this can be done and in which ways it can support the process design. On account of the important attributes of the axi-

1. For more information about Prolog, see appendix B (section B.1).

oms, the logical approach enables us to accomplish the final goal of the thesis; i.e. to develop a formal, precise and operational model of process design expertise.

In the following sections, at first, we specify the methodology that we employ to develop our logical model, and we introduce the TOVE project which has provided a set of representations for enterprise generic knowledge (e.g. activity and agent). Then, based on the methodology and TOVE representations, we construct our First Order Logic (FOL) model.

4.1 Formalization methodology

The identification and formalization of generic knowledge has come to be called "Ontological Engineering" [Fox 94].

An ontology is a formal description of entities, properties of entities, and relationships among entities; it forms a shared terminology for the objects of interest in the domain, along with definitions for the meaning of each of the terms [Fox 94]. Ontological Engineering promotes communication and provides an infrastructure to facilitate the sharing and reuse of knowledge across various applications.

One of the important characteristics of Ontological Engineering is its emphasis on providing definitions. It is common that experts use different names to refer to the same concept. For instance "transaction manager" and "case manager" might be different names for the same set of responsibilities for an employee. On the other hand, experts might use the same word for distinct concepts. For instance one might use "case manager" to refer to a person or a team who handles the order from the beginning to the end [Davenport 93] and the other one to a person who is responsible to answer customers' queries [Hammer et al. 93]. By providing adequate definitions for each term, a model plays a key role in providing consistent interpretations and uses of that term.

In order to formalize the agent/activity design strategies, we employ a methodology [Gruninger 94] which is a guiding mechanism to design the ontologies and also provides a framework to evaluate the adequacy and competency of a proposed ontology with respect to the set of questions that arise from applications.

The methodology comprises the following steps:

1. Motivating scenario

   The development of a formal model is motivated by scenarios that arise in different applications. A motivating scenario introduces a problem(s) and its solutions. Documentation of motivating scenarios is important because it provides a rationale for the competency questions (discussed in the next step) and the answers to these questions.

2. Informal competency questions

   The problem which a model of expertise tries to solve is stated as a query or informal question. These questions, to which we refer as "competency questions", provide a criteria to evaluate the competency of the problem solving and reasoning capability of the model. As important, they justify the existence and properties of the entities within the ontology.

   The competency questions are "*ideally defined in a stratified manner, with higher level questions requiring the solution of lower level questions*" [Gruninger 94]. They also assist us to make an initial evaluation whether the questions can be solved by existing ontologies or whether an extension or a new ontology is required.

3. Terminology

   The required terms to ask and answer the competency questions should be identified and represented in first order logic. The terms are specified by the objects with specific properties and relationships. Objects are structured into taxonomies. Constants or variables represent objects,

unary predicates represent properties and n-ary predicates represent relationships among objects.

4. Axioms

Specification of terminology in first order logic is insufficient to construct a formal model. In order to have a precise formalization, the terms must be defined. Axioms are first order logic sentences that provide the definitions for and the constraints on the terms' interpretations. The axioms must be necessary and sufficient to express the competency question and its solutions.

5. Formal competency questions

The informal competency questions should also be defined in first order logic. They are expressed as an entailment, or consistency problem with respect to the defined terminology and axioms. They have one of the following formats where $T_{ontology}$ is the set of axioms in ontology, and Q is a first order sentence using only predicates in $T_{ontology}$.

Determine $T_{ontology} \models Q$

Determine whether $T_{ontology} \not\models \neg Q$; that is, determine if Q is consistent with $T_{ontology}$.

4.2 TOVE project

A goal of the TOVE project is to create a set of enterprise ontologies which have the ability to deduce answers to queries that require relatively shallow knowledge of the domain [Fox et al 93].

Towards this goal, TOVE 1) provides a shared terminology for the enterprise that every application can jointly understand and use, 2) defines the meaning of each term (semantics) in a precise and as unambiguous manner as possible, 3) implements the semantics in a set of axioms that will enable TOVE to automatically deduce the answer to many “common sense” questions about the enterprise and 4) defines a symbology for depicting a term or the concept constructed thereof in a graphical context [Fox 92].

TOVE ontology currently provides representations for activity, time, causality, resources, organization, quality and cost.

4.3 Constructing the logical model of agent/activity design strategies

We develop the FOL model, following the steps of the methodology (as described in section 4.1), and using the TOVE ontology.

4.3.1 Motivating scenario

- Given a process, how can we assign agents to this process to improve its agents’ setup time?

We want to solve the above problem. As we recall from the previous chapter, the agent/activity design strategies (listed below) improve the agent setup time:

1. assigning one agent (with the help of computer) to perform the activity which produces the information and the one which uses that information.

2. assigning one team of agents to perform the activity which produces the information and the one which uses that information.
3. assigning one agent to perform two activities which use the same information.

With respect to these strategies and their effects on agent setup time, we can substitute the above problem with the following one:

- **Given a process, how can we assign agents to this process to satisfy the agent/activity design strategies?**

The problem is solved when the agent assignment of the process satisfies the agent/activity design strategies. This exactly means:

- for all the process subactivities, if a subactivity such as Ac_i produces the information, if a subactivity such as Ac_j uses this information, if agent Ag_i performs Ac_i, and if agent Ag_j performs Ac_j, then:

  1) Ag_i and Ag_j are the same (see Figure 2 on page 68), or

  2) Ag_i and Ag_j are a team (see Figure 3 on page 69), or

  3) Another subactivity such as Ac_k uses this information and Ag_k who performs Ac_k is the same as Ag_j (see Figure 4 on page 69).

FIGURE 2. Ag_i and Ag_j are the same.

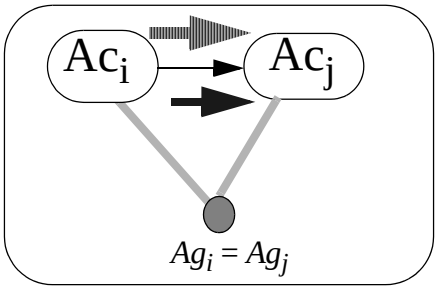


FIGURE 3. Ag_i and Ag_j are a team.

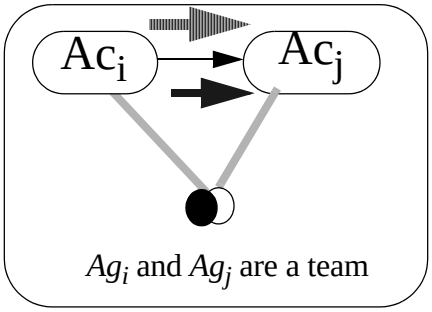
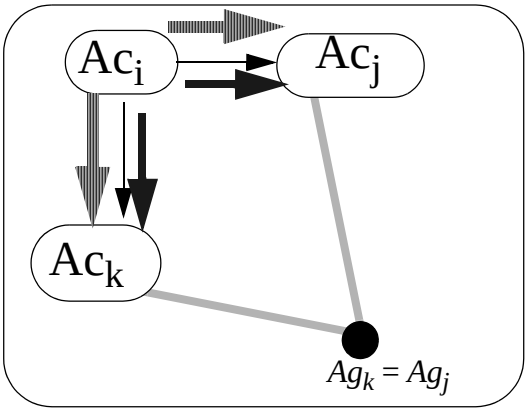


FIGURE 4. Another subactivity such as Ac_k uses this information and Ag_k who performs Ac_k is the same as Ag_j.



4.3.2 Informal competency question

The first version of informal competency question is the problem presented in section 4.3.1:

- **Given a process, how can we assign agents to this process to satisfy the agent/activity design strategies?**

We revise it to meet the following requirements:

- the competency question should be expressed, using FOL, as will be discussed in 4.3.2.1.
- the competency question should be tailored with respect to TOVE's definition of activity, as will be discussed in 4.3.2.2.

- the concepts that are used by the competency question should have consistent definitions at various levels of detail. This will be discussed in 4.3.2.3.

Basically, this section prepares the reader for the final representation of the competency question and justifies the employed terms and their definitions.

4.3.2.1 Expressing the question, using FOL

- **Given a process, how can we assign agents to this process to satisfy the agent/activity design strategies?**

In FOL, the above sentence can be expressed as:

- Given a process, does there exist any agent(s) who is assigned to this process such that the agent/activity design strategies are satisfied?

In FOL, agent/activity design strategies are represented as a **sentence**. We adopt the following convention to refer to two classes of sentences:

1. Any sentence which expresses how agents should be assigned to activities is an **"agent assignment constraint"**. In section 4.5, we will show the use of this class of sentence. Each of the following sentences is an example of "agent assignment constraint":

   There exists one activity which is performed by John Doe.

   All the activities which require management skills are performed by the agents who have managerial skills.

2. Obviously, our agent/activity design strategies are also a subclass of "agent assignment constraint", for they describe a specific way of assigning agents to activities. We label these strategies as **"agent/activity constraint"** and with respect to this label, we modify the competency question as:

- Given a process, does there exist any agent(s) who is assigned to this process such that the "agent/activity constraint" is satisfied?

4.3.2.2 Tailoring the question, with respect to TOVE's definition of activity

- Given a process, does there exist any agent(s) who is assigned to this process such that the "agent/activity constraint" is satisfied?

In TOVE's ontology a process is represented as an (aggregate or complex) activity. Using TOVE's terminology, we ask:

- Given an activity, does there exist any agent(s) who is assigned to this activity such that the "agent/activity constraint" is satisfied?

On the other hand, in TOVE an activity is defined by specifying its agents, subactivities and constraints over the occurrence of these subactivities. Based on this definition, given an activity, the modifications such as eliminating one of its subactivities or assigning agents to its subactivities will lead to an activity which is different from the given one. Therefore, in the competency question, instead of looking for agents, we should search for new activities that satisfy the "agent/activity constraint".

With respect to the above discussion, the competency question is refined into the following form:

- Given an activity, does there exist any "new activity" that satisfies the "agent/activity constraint"?

where the specifications of the "new activity" are:

- "new activity" is, of course, originated from the given activity,
- its agent assignment is different than the given activity,
- in some cases, the subactivities of "new activity" and the given one are identical and in other cases they are not. We elaborate this issue in section 4.3.2.3.

4.3.2.3 Consistent definitions at various levels of detail

- Given an activity, does there exist any “new activity” that satisfies the "agent/activity constraint"?

Suppose the given activity has the following subactivities:

1. preliminary design
2. sending and receiving the preliminary design
3. studying the preliminary design
4. detailed design

Assigning one agent to the first and the fourth subactivity eliminates the need for the existence of the second and third subactivity in the “new activity”. In this case, the subactivities of “new activity” and the given one are not identical.

Under the following two conditions, the above activity can be presented in a way that it only consists of the first and the fourth subactivity; i.e. preliminary design and detailed design. First, the modeler does not see any necessity to go to the detailed level of abstraction to represent send, receive and study activities. Second, since this is an activity which is not currently performed in the enterprise, its agent assignment is not designed yet. In these cases, the “new activity” will have the same subactivities as the given one.

The above discussion indicates that the relationship between the given activity and “new activity” might vary from one level of detail to another. However, we want to define a relationship which is independent of the model’s level of abstraction and is consistent across various scenarios. For that, we introduce the concept of “base activity” and define the “new activity” with respect to this concept.

In simple words, a “base activity” is defined as if one agent performs all of its subactivities. More precisely, its definition is:

Given an activity, the “base activity” has all the subactivities of the given one except the subactivities which are from the class of agent set up activity. Examples of agent setup activities are send, receive and study.

At this point, we are able to give a consistent definition for “new activity” and that is:

“New activity” has exactly the same subactivities as its “base activity”. The performing agents of the new activity are specified.

Using the above concepts, we elaborate the competency question to its final form which is:

- **Given a “base activity”, does there exist any “new activity” that satisfies the "agent/ activity constraint"?**

where (as described in 4.3.1) the "agent/activity constraint" is:

- for all the “new activity” subactivities, if a subactivity such as Ac_i produces the information, if a subactivity such as Ac_j uses this information, if agent Ag_i performs Ac_i, and if agent Ag_j performs Ac_j, then:

  1) Ag_i and Ag_j are the same, or

  2) Ag_i and Ag_j are a team, or

  3) Another subactivity such as Ac_k uses this information and Ag_k who performs Ac_k is the same as Ag_j.

4.3.3 Terminology

The terms employed by the competency question are listed in the first column of Table 2. The FOL representation of each term and a short description of the FOL representation are listed in the second and third columns of Table 2.

TABLE 2. Terminology for agent/activity design strategies

Term	FOL representation	Description
activity	*Do(a,s1,s2)*	An activity whose performing agent is not specified is represented by *Do.* *Do* is a relationship among an activity *(a*), the situation[a] *(s1*) when it starts, and the situation[a] (*s2*) when it ends.
activity and its performing agent	$Do^{\alpha}(a,s1,s2,ag)$	Do^{α} represents an activity whose performing agent is specified. Do^{α} is a relationship among an activity *(a*), the situation[a] *(s1*) when it starts, the situation[a] (*s2*) when it ends, and its performing agent (*ag*).
subactivity	*subactivity(sub-a,a)*	*subactivity* specifies a relationship between an activity *(a*), and its subactivity *(sub-a*). *sub-a* and *a* are both activities.
agent setup activity	*agent-setup(a)*	*agent-setup* specifies a class of activity; e.g. send and receive. *a* is an activity.
base activity	*base-activity(a)*	*base-activity* specifies a class of activity. *a* is an activity.
new activity	*new-activity(na, a)*	*new-activity* is a relationship between a base activity (*a*) and the new activity (*na*) which is generated from the base activity (*a*). *a* and *na* are activities.

TABLE 2. Terminology for agent/activity design strategies

Term	FOL representation	Description
agent/activity constraint	*AAC*(*a*,*ag*)	*AAC* is a relationship which represents the agent/activity constraint. *a* is an activity. *ag* is an agent.
produces information	*produces-information(a,inf,ag)*	*produces-information* is a relationship among an activity (*a)*, its performing agent *(ag*) and the information *(inf)* produced by the activity (*a)*.
uses information	*uses-information(a,inf,ag)*	*uses-information* is a relationship among an activity (*a)*, its performing agent *(ag*) and the information *(inf)* used by the activity (*a)*.
team	*team(ag1, ag2,g)*	*team* is a relationship between two agents (*ag1* and *ag2)* and their common goal, *g*.

a. Given that time is represented as a continuous line, a situation corresponds to a point on this line. For more information about situations, see [Gruninger et al. 94].

4.3.4 Axioms

In this section, the definition of each term is specified, in English and in FOL. The superscript on each term, in the form of "T^{si}", indicates that the definition of term *"T"* is found in step number "*si*" of this section.

1. **What is the "*base activity*"?**

   base-activity(bs-a) specifies that activity *bs-a* is a base activity. If an activity such as *bs-a* is a base activity then its subactivities are not from the class of agent setup. The performing agent(s) of the base activity, *bs-a*, is (are) not specified. See (EQ 35).

   $(\forall$ *bs-a) base-activity(bs-a)* $\supset$ *(*$\forall$*s1,s2)* $(\neg\exists$ *sub-a) Do(bs-a,s1,s2)*^{s9} $\wedge$ *subactivity(sub-a, bs-a)* $\wedge$ *agent-setup -activity*^{s3}*(sub-a).* **(EQ 35)**

2. **What is the "*new activity*"?**

   new-activity(new-a, bs-a) is a relationship between two activities (*new-a* and *bs-a*) where *bs-a* is a base activity and *new-a* is a new activity which is generated from the base activity *(bs-a)*. This relationship is specified as follows:

   > All the subactivities of the base activity *(bs-a)* are included in the new activity *(new-a*) and all the subactivities of the new activity *(new-a*) are included in its base activity *(bs-a)*. See (EQ 36).
   >
   > The performing agents of the base activity *(bs-a)* are not specified, but the performing agents of new activity *(new-a*) are specified. See (EQ 36).

   ($\forall$*bs-a, new-a) new-activity(new-a, bs-a)* $\equiv$ **(**$\forall$*s1,s2,s3,s4, sub-a,sub-aa) base-activity(bs-a)* $\wedge$ *(subactivity(sub-a, bs-a)* $\supset$ *subactivity(sub-a, new-a))* $\wedge$ *(subactivity(sub-aa, new-a)* $\supset$ *subactivity(sub-aa, bs-a))* $\wedge$ $(\exists$*ag) Do*$^{\alpha}$*(new-a,s3,s4,ag)*^{s9} **(EQ 36)**

3. What is the "*agent setup activity*"?

"Agent setup activity" is a class of activity. Prior to performing an activity (*act*), an agent *(ag*) should know some information *(inf*). In order to obtain that information *(inf)* the agent *(ag)* might need to perform an activity *(set-a*). And, *set-a* might include activities such as send, receive and/or study. We classify *(set-a*) as an agent setup activity and define it as:

set-a is an agent setup activity, iff:

inf is the knowledge precondition for an activity *act* (i.e. *ag*, the performing agent of *act*, can not initiate *act* until *ag* knows *inf*), and

ag will know *inf*, if the agent setup activity *(set-a)* is performed (i.e. if *inf* is not known and *set-a* is performed then the agent *ag* will obtain *inf*).

One of the terms that we used in the above definition is "agent knows". The FOL representation of this term is *Ks(ag,inf,s)* where *ag* is an agent, *inf* is the information known by this agent in situation *s*. For more information about *Ks*, see step 7 on page 80.

Equations 37, 38 and 39, together, represent the definition for the agent setup activity *(set-a)*.

$$(\forall \textit{set-a})\ \textit{agent-setup-activity(set-a)} \equiv (\forall \textit{act, inf})\ \textit{knowledge-precondition(act,inf)} \wedge \textit{achieve(set-a,act,inf)}. \quad \textbf{(EQ 37)}$$

$$(\forall \textit{act,inf})\ \textit{knowledge-precondition(act,inf)} \equiv (\forall\ \textit{s1, s2,ag})\ Do^{\alpha}(act,s1,s2,ag)^{s9} \supset Ks(ag,inf,s1)^{s7} \quad \textbf{(EQ 38)}$$

$$(\forall\ \textit{set-a,act,inf})\ \textit{achieve(set-a,act,inf)} \equiv (\forall s3,s4,ag) \neg Ks(ag,inf,s3)^{s7} \wedge Do(set\text{-}a,s3,s4)^{s9} \supset Ks(ag,inf,s4)^{s7}. \quad \textbf{(EQ 39)}$$

4. What is *"AAC*"?

AAC(a, ag) is a relationship between an activity and its performing agents. It represents the "agent/activity constraint" and is defined as:

Given that *a* is an activity, for all subactivities of *a*:

if *a2* is a subactivity which uses the information, *inf*, and the performing agent of *a2* is *ag2*,

if *a1* is a subactivity which produces the information, *inf*, and the performing agent of *a1* is *ag1*,

then:

AAC1(ag1,ag2) is true; meaning that *ag1* and *ag2* are the same, or

AAC2(ag1,ag2) is true; meaning that *ag1* and *ag2* are a team, or

AAC3(a,a2, ag2,a3,ag3,inf) is true; meaning that there is another subactivity such as *a3* (which is different than *a2*), *a3* uses the information, *inf*, and *ag3* who is the performing agent of *a3* is the same as *ag2*.

(EQ 40), (EQ 41), (EQ 42), and (EQ 43) represent the above definition.

$(\forall a,ag)\ AAC(a,ag) \equiv (\forall a1,a2,ag1,ag2,inf,s1,s2,s11,s12,s21,s22,s31,s32)\ Do^{\alpha}(a,s1,s2,ag) \wedge subactivity(a1, a) \wedge subactivity(a2,a) \wedge Do^{\alpha}(a1,s11,s12,ag1) \wedge Do^{\alpha}(a2,s21,s22,ag2) \wedge uses\text{-}information^{s5}(a2,inf,ag2) \wedge produces\text{-}information^{s6}(a1,inf,ag1) \supset AAC1(ag1,ag2) \vee AAC2(ag1,ag2) \vee \exists a3,ag3 AAC3(a,a2,ag2,a3,ag3,inf).$ **(EQ 40)**

$(\forall ag1,ag2)\ AAC1(ag1,ag2) \equiv (ag1 = ag2).$ **(EQ 41)**

$(\forall ag1,ag2)\ AAC2(ag1,ag2) \equiv team(ag1, ag2,g).$ **(EQ 42)**

$(\forall a,a2, ag2,inf,a3,ag3)\ AAC3(a,a2, ag2,a3,ag3,inf) \equiv subactivity(a3,a) \wedge \neg(a3 = a2) \wedge (\forall s31,s32)\ Do^{\alpha}(a3,s31,s32,ag3) \supset uses\text{-}information^{s5}(a3,inf,ag3) \wedge (ag2 = ag3).$ **(EQ 43)**

- **Assumption**

The above definition is based on the following assumption that specifies the relationship between the performing agent of an activity and the performing agent(s) of its subactivities.

If *a* is an activity, if *ag* is the performing agent of *a*, if *a1* is a subactivity of *a*, and if *ag1* is the performing agent of the subactivity *a1*, then *ag* and *ag1* are the same or *ag* is a group agent and one of its members is *ag1*.

$(\forall a,ag,ag1,a1,s1,s2,s3,s4)\ Do^{\alpha}(a,s1,s2,ag) \wedge subactivity(a1,a) \wedge Do^{\alpha}(a1,s3,s4,ag1) \supset (ag1 = ag) \vee (member(ag1,ag))$. **(EQ 44)**

5. What is the definition of *"uses-information"*?

uses-information(a,inf,ag) is a relationship among an activity (*a*), an agent *(ag*) and information *(inf),* as follows:

Activity *a* uses information *inf* iff *a* can not be initiated until its performing agent *ag* has the information *inf.*

$(\forall a,inf,ag)\ uses\text{-}information(a,inf,ag) \equiv (\forall\ s1,s2)\ Do^{\alpha}(a,s1,s2,ag)^{s9} \supset Ks(ag,inf,s1)^{s7}$ **(EQ 45)**

An example of an activity which uses information is "preliminary design" which uses the information "product specification".

6. What is the definition of *"produces information"*?

produces-information(a,inf,ag) is a relationship among an activity (*a*), an agent *(ag*), and information *(inf),* as follows:

Activity *a* produces information *inf* iff before the start of *a,* its performing agent, *ag*, does not know *inf* and at the end of *a* the agent, *ag*, knows it.

$(\forall a,\ inf,ag)\ produces\text{-}information(a,\ inf,ag) \equiv (\forall s1,\ s2) \neg Ks(ag,inf,s1)^{s7} \wedge Do^{\alpha}(a,s1,s2,ag)^{s9} \supset Ks^{s7}(ag,inf,s2)$. **(EQ 46)**

An example of an activity which produces information is "providing product requirements" which produces "product requirement".

7. What is *"Ks"*?

Ks(ag,inf,s) is a relationship among an agent *(ag*), information[1] *(inf)*, and a situation *(s)*. *Ks(ag,inf,s)* specifies that agent *ag* knows information *inf* in situation *s*.

An example is: after reading the database, John knows Joe's credit status.

Following is the FOL representation of this sentence:

$Ks(John,credit\text{-}status(Joe), s1) \wedge (s1 = do(read(dbase), s))$.

where *dbase* represents the data base and *do(read(dbase), s)* represents the situation (*s1*) when activity "reading the data base" is terminated.

8. What is a *"team"*?

There are many different types of teams. In this work, the term *"team"* refers to a group of agents who has a common goal.

There exists a team relationship between two agents if the agents have an identical goal. See (EQ 47).

$$(\forall ag1, ag2, g)\ team(ag1,ag2,g) \equiv (\forall s)\ holds(agent\text{-}constraint(ag1, goal(g,ag1)),s) \wedge holds(agent\text{-}constraint(ag2, goal(g,ag2)),s). \quad \textbf{(EQ 47)}$$

where *holds(agent-constraint(ag1, goal(g,ag1)),s)* is the representation of an agent's goal in TOVE and in this representation, *ag1* is an agent, *g* is a goal[2], and *s* is the situation[3] when the agent has that goal.

1. Information is represented as a fluent where a fluent is a predicate or function whose value may change with time.
2. Goal is a fluent; i.e. a predicate or function whose value changes over time.
3. For more information about situations, see [Gruninger et al. 94].

9. What is the definition of *"Do"* and *"Do$^{\alpha}$"*?

In TOVE, an activity is represented by one of the following predicates:

- *Do(a,s1,s2)* in which *a* is an activity, *a* is initiated in situation *s1* and terminated in situation *s2*.

  The representation *Do* is used when the performing agent of the activity is not specified.

- *Do$^{\alpha}$(a,s1,s2,ag)* in which *a* is an activity, *a* is initiated in situation *s1*, terminated in situation *s2*, and *ag* is the agent who performs *a*.

  The representation *Do$^{\alpha}$* is used when the performing agent of the activity is specified.

Situations are defined as distinguished intervals on the time line. A more complete specification of situations can be found in [Gruninger et al. 94].

4.3.5 Formal competency question

Using the above terms and axioms, we present the formal competency question as[1]:

Given a base activity^{s1}, *A*, does there exist a new activity^{s2}, *new-a* that satisfies the "agent/activity constraint"^{s4}, *AAC*?

$$Theories \models (\exists\ new\text{-}a,ag)\ base\text{-}activity(A) \wedge new\text{-}activity(new\text{-}a, A) \wedge AAC(new\text{-}a,ag). \quad \textbf{(EQ 48)}$$

where *Theories* are TOVE theories and the axioms which defined the objects and predicates used by our competency question. These axioms were presented in section 4.3.4.

4.4 Extending the model

[Fox 93] proposed a set of criteria based on which enterprise ontologies can be evaluated. One of these criteria is extensibility; i.e. can the new concepts be easily added to the representation? In

1. Note the superscript on each term, indicates that the term is defined in step number "*si*" of section 4.3.4.

this section, we demonstrate how our model can be extended to encompass a new class of constraint.

- From section 4.3.5, we know the problem that our FOL model solves is: Given a base activity, *A*, does there exist a new activity, *new-a* that satisfies the "agent/activity constraint", *AAC*?

The solutions to this problem are alternative new activities which satisfy the "agent/activity constraint". The alternatives are generated without respect to any other constraint, such as “agents capability”, “agents availability” and so on.

Suppose we want to augment the model so that its proposed alternatives also satisfy the **“agents’ capability constraint”**. This means, if the model assigns an agent to any subactivity of the new activity, then that agent should have the skills required to perform that subactivity. We add this capability to the model, through the following steps:

1. We identify the problem that our extended model should solve as:

   Given a base activity, *A*, does there exist a new activity, *new-a*, which satisfies the "agent/activity constraint", *AAC*, and satisfies the **“agents’ capability constraint”**?

2. We identify the required terms to state the competency question.

   The new terms we need are: “agents’ capability constraint”, “the skills which are required to perform an activity” and “the agent’s skills”.

   ACPC(a,ag) is a relationship which represents the “agents’ capability constraint”, in which *a* is an activity and *ag* is an agent.

   required-skill(a,f) is a TOVE predicate which specifies that skill *f*[1] is required to perform activity *a*.

1. Skill, *f*, is a fluent; i.e. a predicate or function whose value changes over time.

has-skill(ag,f) is a TOVE predicate which specifies that agent *ag* has the skill f^1.

3. Using the above terms and also employing the terms Do^{α} and *subactivity* (see Table 2 on page 74), we define the "agents' capability constraint", as:

   Given that *a* is an activity, for all of its subactivities such as *a1*, if the required skill to perform *a1* is *f1*, and if the performing agent of *a1* is *ag1*, then *ag1* should have the skill *f1*.

   $(\forall a,ag)\ ACPC(a,ag) \equiv (\forall a1,ag1,f1,s1,s2,s11,s12)\ (Do^{\alpha}(a,s1,s2,ag) \wedge subactivity(a1, a) \wedge Do^{\alpha}(a1,s11,s12,ag1) \wedge required\text{-}skill(a1,f1)) \supset has\text{-}skill(ag1,f1).$ **(EQ 49)**

4. And finally, the formal competency question of the extended model will be:

   $Theories \models (\exists\ new\text{-}a,ag)\ base\text{-}activity(A) \wedge new\text{-}activity(new\text{-}a, A) \wedge ACC(new\text{-}a,ag) \wedge ACPC(new\text{-}a,ag).$ **(EQ 50)**

The above competency question (stating that given a base activity, *A*, does there exist a new activity, *new-a*, which satisfies the "agent/activity constraint", *AAC*, and satisfies the "agents' capability constraint", *ACPC*?) indicates the problem solving and reasoning capability of our extended model.

4.5 Generalization of the competency question

The "agent/activity constraint" and "agents' capability constraint" are sentences with a commonality and that is, they express how agents should be assigned to activities. Another example of such a sentence is:

- Jane Doe is the only agent who should perform the activity of evaluating the transcripts.

  $(\forall s1,s2,ag,t)\ (Do^{\alpha}(evaluate(t),s1,s2,ag) \wedge transcript(t)) \supset ag = Jane\text{-}Doe.$ **(EQ 51)**

As mentioned in section 4.3.2.1, we classify these sentences as "agent assignment constraint". We can generalize our competency question with respect to this class as:

> Given a base activity, *A*, does there exist a new activity, *new-a*, which satisfies a given "agent assignment constraint"?
>
> *Theories* |= ($\exists$ *new-a,ag) base-activity(A)* $\wedge$ *new-activity(new-a, A)* $\wedge$ φ(*new-a,ag*). **(EQ 52)**
>
> where "agent assignment constraint" is any sentence in the form of φ*(new-a,ag)* which expresses how agent *ag* is assigned to activity *new-a.*

The previous competency questions (presented in 4.3.5 and 4.4) are subclasses of this general class.

4.6 Summary

In this chapter, we created a FOL model of agent/activity design strategies which can solve the following problem:

> Given a base activity, *A*, does there exist a new activity, *new-a* which satisfies the "agent/activity constraint", *AAC*?

The solutions to the above problem are alternative new activities which satisfy the "agent/activity constraint".

We showed how to extend the reasoning capability of the logical model of agent/activity design strategies so that its proposed alternatives also satisfy the **“agents’ capability constraint”**. This means, if the model assigns an agent to any subactivity of the new activity, then that agent should have the skills required to perform that subactivity.

We also introduced a general class of problem to which the competency question that our model solves belongs.

Since the model of agent/activity design strategies is in FOL, it can be implemented in a logic programming language such as Prolog. In chapter 6, we discuss the implementation of the model.

Chapter 5

Design validation model

In this chapter, we develop a First Order Logic model of three categories of BPR expertise. They are:

- Dangling information
- Case management
- Changeable agent assignments

The logical model can be employed to validate an existing process design. For this reason, we refer to it as design validation model. Specifically, the design validation model is capable of making process designers aware of missing elements in the process structure.

- The focus of “dangling information” is on information flow. It finds situations where an activity which should use a specific piece of information is missing.
- With respect to the “case management” expertise, the model identifies the situations where an agent with a certain role does not exist, a piece of information which is required by this agent and/or by the process customer are left out from the process definition.

- In regard to the “changeable agent assignments” expertise, the model finds an agent assignment for which a cancellation is not defined.

A missing element is not necessarily a sign of a problem. A problem develops when the omission is due to a design oversight rather than a design decision. A process designer can use the design validation model to identify these missing parts. With respect to each omission the designer has two choices, either altering the process design to eliminate the omission or to accept the design as it is. In chapter 6, through an example, we demonstrate the application of the design validation model.

As before, we build the above models upon the TOVE ontologies. For “dangling information”, there is no need for new ontologies; it uses the TOVE ontologies, already presented for agent/activity design strategies.

However, for “case management”, we need to introduce the TOVE representation of what is the truth value of a property at different time points and how this value changes, as well as the TOVE representation of agent constraints. These representational constructs will also be employed by the “changeable agent assignments”.

In order to create the design validation model, we follow the methodology [Gruninger 94] (which was described in the previous chapter). However, we will be less rigorous than we were in developing the logical model of agent/activity design strategies.

5.1 Dangling information

5.1.1 Motivating scenario and informal competency question

Often, the information produced by an activity is needed to enable another activity such as reporting, monitoring and/or decision making. The activity which produces the information and the one

which uses it can be the subactivities of one process, as well a the subactivities of two separate processes. Unless the use of information is not included as a part of the processes, the relationship between the information produced and the decisions or actions taken can never be understood or improved.

The term “dangling information” refers to the information which is produced by an activity but not used by any other activity. The existence of “dangling information” might be an indicator of incompleteness of the process definition.

Answering the following competency question, the “dangling information” expertise identifies the situations where the contribution of the obtained information to the continuation of process(es) is not recognized.

- Given a set of activities, does there exist a dangling piece of information?

which precisely means:

- **Given a set of activities, does there exist a piece of information that is produced by an activity and not used by any other activity?**

5.1.2 Terminology and axioms

In the previous chapter, the required terms to ask and answer the above competency question were introduced. Table 3 reiterates these terms and their FOL representation to refresh the reader’s memory. The axioms which define these terms were presented in chapter 4, section 4.3.4, steps 5 and 6 on page 76.

TABLE 3. Terminology for the "dangling information"

Term	FOL representation	Description
produces information	*produces-information(a,inf,ag)*	*produces-information* is a relationship among an activity (*a)*, its performing agent *(ag*) and the information *(inf)* produced by the activity (*a)*.
uses information	*uses-information(a,inf,ag)*	*uses-information* is a relationship among an activity (*a)*, its performing agent *(ag*) and the information *(inf)* used by the activity (*a)*.

5.1.3 Formal competency question

Using the above terms, we present the formal competency question as:

$$Theories \models (\exists\ a1,inf,ag1)\ produces\text{-}information(a1,\ inf,ag1) \wedge \neg\ (\exists\ a2,ag2)\ uses\text{-}information(a2,inf,ag2). \quad \textbf{(EQ 53)}$$

stating that:

- Given a set of activities, does there exist a piece of information that is produced by an activity and not used by any other activity?

5.2 Case management

5.2.1 Motivating scenario

It is important that for each transaction moving through a process, one agent is assigned to manage it. For example, if the process focuses on customer service, the agent should be known by the customer of the process. In order to answer the customer's queries about the transaction status, s/he should be able to trace the transaction. We call this agent the "case manager". The existence of

such an agent is an indicator that the customer's demand can be captured through all steps of the process.

The idea was discussed in the reviewed heuristics, e.g. "A Case Manager provides a single point of contact", [Hammer et al. 93].

Basically, "case manager" is an organizational role which is defined in terms of the authorities and responsibilities of the agent who fills this role. The requirements of enterprises/processes vary and so do their definitions for the "case manager" role. For instance, some case managers might have full access to pricing and credit policies, be empowered to assign agents to perform the subactivities of the process and/or even change the schedule of those subactivities.

In this thesis, we assume that the minimum responsibility of the agent who fills this role is to answer the customer's queries; in particular the queries about the transaction status. We consider a case that a process is given, it has a customer, and there is a "case manager" role associated with this process. In this case, any of the following situations might be an indication of the flaw in the effectiveness of the "case manager" role.

1. John who was the "case manger" of this process, resigned, and a new case manager is not assigned yet. This means Jill, the customer, has lost her contact point and consequently she might experience some delay, when asking a query to which an answer is required.
2. Consider in the above scenario, Joe is assigned as the new case manager but Jill the customer has not been informed yet. From the customer's point of view, having an unknown case manager is as useless as not having a case manager at all; in both cases, Jill can not smoothly transfer her needs to the enterprise.
3. Jill finally understood Joe is the current "case manager" for her transaction. She calls him to know the current status of her transaction. Currently Jack is processing Jill's transaction but Joe, the case manager, has not been informed yet. If he was informed as soon as Jack was assigned, he could answer Jill much faster.

Given a process, our objective is to look for the situations- similar to the above scenarios- where the role of case manger might lose its effectiveness. To achieve this objective, we are interested to know:

1. Is there a time when no agent fills the “case manager” role?
2. Is there a time when an agent fills the “case manager” role but this agent is not known by the customer?
3. Is there is a time when an agent should perform a subactivity of the process and the “case manager” of this process does not know it?

5.2.2 Informal competency questions

Above, we presented the first version of our competency questions. In this section, we discuss the representational requirements to ask and answer them, introduce TOVE’s ontologies that fulfil these requirements, and finally use the discussion to restate our competency questions. This section justifies our terminology and formal competency questions which will be presented later.

5.2.2.1 Temporal projection

In the above questions, we validate the effectiveness of the case manager’s role by examining the truth value of a property of a given process at various time points. For instance, we check to see whether there is a time when no agent fills the “case manager” role. This entails that we should be able to represent what is the truth value of a property at different time points and, as important, represent how this value changes; e.g. an agent who filled this role before, does not fill it any more.

What changes the truth value of a property is the occurrence of actions, for instance, as the result of the assignment, an agent would fill the “case manager” role. Evaluation of the truth value of a proposition at some point in time- on the basis of a set of actions that occur at different points- is also called “Temporal Projection” [Gruninger et al. 95b].

TOVE adopted the situation calculus to provide the ontologies for Temporal Projection. Here, we only use what TOVE has provided, without going through the details. However, more information about situation calculus can be found in [Gruninger et al. 94]. We make use of the following TOVE predicates to represent what is the truth value of a property at different time points, and how the occurrence of an action changes or does not change this value:

- *holds(f, s)* is used to represent that a property *(f)* of the world is true in situation *s*.

  There is an initial situation, and the world changes from one situation to another when actions are performed [Gruninger et al. 95b].
- *holdsT(f, t)*, representing that a property *(f)* of the world is true at time *t*.

  Given that time is represented as a continuous line, a situation corresponds to a point on this line.
- *holds(f, do(a,s))* is used to represent what holds or does not hold in the world after performing some action *(a)* in situation *s*.
- *occursT(a, t)*, represents the occurrence of an action *a* at a time point *t*.

5.2.2.2 Agent constraints

Recall the problems presented in the motivating scenario. In those problems, we specifically need to represent the following terms: organizational role, “an agent who should perform a subactivity”, and a customer. Basically, TOVE groups these terms under the category of “agent constraints”. Since all agent constraints (e.g. an organizational role) may change between situations, TOVE represents them, using *holds/2* (as presented in the previous section).

Following, we describe TOVE’s ontologies for the above terms. We also present TOVE’s general approach that allows us to define different agent constraints and tailor them, according to the requirements of various organizations.

1. **Organizational role**

In TOVE, any organizational role is viewed as an agent constraint; a role is a group of constraints that any agent filling that role should satisfy. TOVE represents an organizational role, as follows:

holds(agent-constraint(ag, role(ag,r)),s), specifying in situation *s*, agent *ag* has the constraint to fill role *r.*

2. **An agent should perform a subactivity**

   This is also another type of agent constraint which again is represented by *holds/2* (presented in the previous section), as follows:

   holds(agent-constraint(ag, process-obligation(ag,ac),s)), specifying in situation *s*, agent *ag* has the constraint to perform activity *ac.*

3. **Customer**

   Similarly, TOVE's customer is represented as:

   holds(agent-constraint(ag,process-customer(c,a)),s), specifying that in situation *s*, agent *c* is the customer of activity *a*.

4. **Agent constraint**

   TOVE provides the following form to allow enterprise modelers to represent and define any type of agent constraint.

   $(\forall ag,x,s)$ *holds(agent-constraint(ag,c(x)),s)* $\equiv \phi(ag,s)$

   Intuitively, *agent-constraint(ag,c(x))* specifies agent *ag* must satisfy the sentence associated with the constraint name term *c*. For more information about agent constraints, see [Gruninger 95c].

5.2.2.3 Last version of informal competency questions

Using TOVE's terminology, we present the informal competency questions as:

- Assuming that an activity[1], *A*, its customer, the occurrences of those subactivities which modify the assignment of an agent to a role or to a subactivity of *A*, and also those subactivities which produce information are given,
  1. Is there a time point (during the performance of *A*) when the customer exists but no agent is assigned to the “case manager” role?
  2. Is there a time point (during the performance of *A*) when an agent has the “case manager” role but this agent is not known by the customer?
  3. Is there a time point when an agent has the obligation to perform one of the *A*’s subactivities but the “case manager” does not know it?

Above, the underlined sentence identifies the appropriate input for the competency questions; i.e. what should be given so that the questions can be answered.

1. From the previous chapter, recall that in TOVE’s ontology, a process is represented as an (aggregate or complex) activity.

5.2.3 Terminology

The terms employed by the competency questions are listed in the first column of Table 4. The FOL representation of each term and a short description of the FOL representation are listed in the second and third columns of Table 4.

TABLE 4. Terminology for the "case management"

Term	FOL representation	Description
subactivity	*subactivity(sub-a,a)*	*subactivity* specifies a relationship between an activity *(a)*, and its sub-activity *(sub-a)*. *sub-a* and *a* are both activities.
Agent knows	*Ks(ag,inf,s)*	*Ks* is a predicate which specifies agent *ag* knows information *inf* in situation *s*.
occurrence of an action	*occursT(a, t)*	action *a* occurs at time *t*.
The truth value of a property in a situation	*holds(f, s)*	In general, *holds(f, s)* represents the fact that fluent *f* is true in situation *s*, where fluent is a predicate or function whose value may change between situations.
The truth value of a property at a time point	*holdsT(f, t)*	In general, *holdsT(f, t)* represents the fact that fluent *f* is true at time point *t*, where fluent is a predicate or function whose value may change.

TABLE 4. Terminology for the "case management"

Term	FOL representation	Description
The truth value of a property after termination of an activity	*holds(f, do(terminate(a),s)).*	The property *f* is true when activity *a* is terminated. *f* is a fluent (i.e. a predicate or function whose value may change between situations). The function *do(a,s)* is a name of a situation that results from performing the action *terminate(a)* in situation *s*.
In our competency questions, we specifically use "holds/2" to represent:		
organization-al role	*holds(agent-constraint(ag, role(ag,r)), s)*	This predicate specifies that agent *ag* has role *r* in situation *s.*
an agent is assigned as the "case manager".	*holds(agent-constraint(ag, case-manager(agc,a)),s)*	This predicate specifies that in situation s, the agent *agc* is the case manager of process *a*.
an agent is assigned to perform an activity	*holds(agent-constraint(ag, process-obligation(ag,ac)),s)*	This predicate specifies that in situation s, the agent *ag* has the constraint to perform activity *ac*.
customer	*holds(agent-constraint(ag, process-customer(ag, a)),s)*	This predicate specifies that in situation *s*, agent *ag* is the customer of activity *a*.
any agent constraint	*holds(agent-constraint(ag,c(x)),s)*	*agent-constraint(ag,c(x))* specifies agent *ag* must satisfy the sentence associated with the constraint name term c.

5.2.4 Axioms

1. **What is the definition of the *"case manager"*?**

   As mentioned above, the definition of the "case manager" varies from one enterprise to another. However, in this thesis we assume that any agent who fills this role should satisfy the following constraint:

   If an agent *(agc*) is assigned as a case manager of an activity *(a)* with customer *(cus)*, then this agent should answer the customer's queries.

   $(\forall agc,a,cus,f,t1)\ holdsT(agent\text{-}constraint(agc,case\text{-}manager(agc,a)),t1) \equiv ((holdsT(agent\text{-}constraint(cus,process\text{-}customer(cus,a)),t1) \wedge occursT(ask(f,agc,cus),t1)) \supset (\exists t)\ occursT(answer(f,cus,agc),t) \wedge t => t1).$ **(EQ 54)**

2. **What are the definitions of a *"role"*, and a *"process customer"*?**

   For formal and informal definitions of organizational roles, customer and agent constraint, see [Gruninger 95c].

3. **What are the assumptions?**

   Following are the assumptions under which the model will provide the right answers to the competency questions.

   - Assumption *HC* is the following sentence:

     $(\forall a)\ (\exists cust,t)\ holdsT(agent\text{-}constraint(cust,process\text{-}customer(cust,a)),t).$ **(EQ 55)**

     stating that an activity should have a customer.

   - Assumption *CM* is the following sentence:

     $(\forall a)\ (\exists agc,t)\ holdsT(agent\text{-}constraint(agc,case\text{-}manager(agc,a)),t).$ **(EQ 56)**

     stating that the activity should have a case manager.

- Assumption *SO* states that the termination times of subactivities which change (impose or cancel) the agent's assignments are specified.

  $(\forall sub\text{-}a,a,ag,t)\ subactivity(sub\text{-}a,a) \wedge (change\text{-}assignment\text{-}activity(a,ag) \vee inf\text{-}producing\text{-}activity(a,ag)) \supset occursT(teminate(sub\text{-}a),t) \equiv (sub\text{-}a = A_1 \wedge t = T_1) \vee ... \vee (sub\text{-}a = A_n \wedge t = T_n).$ **(EQ 57)**

  $(\forall a,ag)\ change\text{-}assignment\text{-}activity(a,ag) \equiv (\forall ac,c,r,s)\ (\neg holds(agent\text{-}constraint(ag,c), s) \wedge holds(agent\text{-}constraint(ag,c), do(terminate(a),s)) \vee (holds(agent\text{-}constraint(ag,c), s) \wedge \neg holds(agent\text{-}constraint(ag,c), do(terminate(a),s))) \wedge (c = process\text{-}obligation(ag,ac) \vee c = role(ag,r)).$ **(EQ 58)**

  $(\forall a,ag,s)\ inf\text{-}producing\text{-}activity(a,ag) \equiv Ks(ag,inf,do(terminate(a),s)).$ **(EQ 59)**

5.2.5 Formal competency questions

Assuming that an activity, A, its customer, the occurrences of those subactivities which modify the assignment of an agent to a role or to a subactivity of A, and also those subactivities which produce information are given,

1. **Is there a time point (during the performance of A) when the customer exists but no agent is assigned to the "case manager" role?**

   $Theories \wedge CM \wedge HC \wedge SO \models (\exists\ sub\text{-}a,agc,cus,t)\ subactivity(sub\text{-}a,A) \wedge occursT(terminate(sub\text{-}a), t) \wedge holdsT(agent\text{-}constraint(cus,process\text{-}customer(cus,A)),t) \wedge \neg holdsT(agent\text{-}constraint(agc,case\text{-}manager(agc, A)), t).$ **(EQ 60)**

   Please note that in the above FOL statement, $[subactivity(sub\text{-}a,A) \wedge occursT(terminate(sub\text{-}a), t)]$ ensures that the model only looks for those time points that are during the performance of *A*.

2. **Is there a time point (during the performance of A) when an agent has the "case manager" role but this agent is not known by the customer?**

   Theories $\wedge$ *CM* $\wedge$ *HC* $\wedge$ *SO* $\models$ ($\exists$ *sub-a,agc,cus,t) subactivity(sub-a,A)* $\wedge$ *occursT(terminate(sub-a), t)* $\wedge$ *holdsT(agent-constraint(agc,case-manager(agc, A)), t)* $\wedge$ *holdsT(agent-constraint(cus,process-customer(cus,A)),t)* $\wedge$ $\neg$*ks(cus, agent-constraint(agc,case-manager(agc, A)), t).* **(EQ 61)**

   Please note that in the above FOL statement, *[subactivity(sub-a,A)* $\wedge$ *occursT(terminate(sub-a), t)]* ensures that the model only looks for those time points that are during the performance of *A.*

3. **Is there a time point when an agent has the obligation to perform one of the A's subactivities but the "case manager" does not know it?**

   Theories $\wedge$ *CM* $\wedge$ *HC* $\wedge$ *SO* $\models$ ($\exists$ *sub-a,ag,agc,cus,t) subactivity(sub-a,A)* $\wedge$ *holdsT(agent-constraint(ag,process-obligation(ag,sub-a)), t)* $\wedge$ *holdsT(agent-constraint(agc,case-manager(agc, A)),t)* $\wedge$ $\neg$*ks(agc,(agent-constraint(ag,process-obligation(ag,sub-a)), t).* **(EQ 62)**

5.3 Changeable agent assignments

5.3.1 Motivating scenario, informal and formal competency question

Except for activities such as "drying paint"[1] which are performed independent of any agent, other activities have to be performed by agents. The process definition should incorporate the possibility of change in agent assignments. We determine if a given process design satisfies this criterion, through asking the following question:

1. TOVE categorizes the activities which can be performed independent of agents under "natural activities" [Gruninger 95c].

- Once an agent is assigned to perform a role or an activity, does there exist any activity which can change this assignment?

Organizational roles and assignments are all considered as agent constraints. We already described the concept in section 5.2.2.2 on page 93 and in Table 4 on page 96. With respect to this concept, the above question is modified into the following informal competency question:

- Given an activity[1] *A*, its subactivities and a set of agent constraints in the form of roles and process obligations, once a constraint is imposed on an agent, does there exist any subactivity which can remove this constraint?

This is translated into the following formal competency question:

$$Theories \models (\exists\ ag,c,r,ac,s)\ holds(agent\text{-}constraint(ag,c),s) \wedge (\exists\ sub\text{-}1,s\text{-}1)\ subactivity(sub\text{-}1,A) \wedge \neg holds(agent\text{-}constraint(ag,c), do(terminate(sub\text{-}1), s\text{-}1) \wedge (c = role(ag,r) \vee c = process\text{-}obligaion(ag,ac)). \quad \textbf{(EQ 63)}$$

5.4 Summary

In this chapter we developed the logical model of three categories of BPR expertise that can be employed to validate a design process. Table 5 presents the summary of our work. In each row, the first cell presents the name of the BPR expertise and a brief description. With respect to each category, the model is able to answer a question(s). The second cell specifies the question(s).

1. From the previous chapter, recall that in TOVE's ontology, a process is represented as an (aggregate or complex) activity.

TABLE 5. Summary of the design validation model

BPR expertise	The question(s) that the model answers
Dangling information The contribution of the output of an activity to the continuation of a process(es) should be recognizable. Existence of Dangling Information might be an indicator of an incomplete process definition.	Given a set of activities, does there exist a piece of information that is produced by an activity and not used by any other activity?
Case management For each transaction moving through a process, one agent should be assigned as the contact point for the customer and the customer should know this agent. We recognize this agent as the "case manager". This is an indicator that the enterprise is capable of capturing the customer's demands through all the steps of the transaction. The "case manager "should be able to trace the transaction. Traceability indicates that the "case manager" can respond to the customer's questions; specifically, the questions about the transaction's status.	Given an activity, is there a time point when no agent is assigned to the "case manager" role? is there a time point when an agent has the "case manager" role but this agent is not known by the customer? is there a time point when an agent has the obligation to perform one of the subactivities but the "case rmanager" does not know it?
Changeable agent assignments The process design should allow the modification or cancellation of "agent assignments".	Given an activity A, its subactivities and a set of agent constraints in the form of roles and process obligations, once a constraint is imposed on an agent, does there exist any subactivity which can remove this constraint?

Chapter 6

Incorporating FOL models into a software tool

We have implemented the logical models of agent/activity design strategies and design validation expertise in Prolog. We incorporated the implemented models into a software tool which we call the “Process Integration advisor”, as it assists in integrating the agents and activities within a process. The goals of this chapter are to describe the implementation of the logical models in Prolog, and to demonstrate the usability of the Process Integration advisor in process design and analysis.

To accomplish the first goal, in section 6.1, we focus on the logical model of agent/activity design strategies and explain its implementation in Prolog[1]. Since the Prolog axioms for the design validation model are very similar to their corresponding FOL axioms that we presented in the previous chapter, we do not discuss the implementation of this model in detail. However, the Prolog programs of the model can be found in appendix B (sections B.3.3, B.3.4, and B.3.5).

To accomplish the second goal, in sections 6.2, 6.3, and 6.4, we describe a hypothetical process and apply the Process Integration advisor to it. This allows us to analyze the process design from the aspects of “dangling information”, “case management”, “changeable agent assignments”, and “agent/activity design strategies”, and to provide a set of recommendations to improve the robustness of the process in these areas. Table 7 on page 118 summarizes the results of this analysis.

1. Precisely, Quintus Prolog which is one of the implementations of Prolog.

6.1 Implementation of agent/activity design strategies

In this section, we discuss the implementation of agent/activity design strategies in Prolog.

6.1.1 Implementation technique

The competency question that the FOL model of agent/activity design strategies (developed in chapter 4) attempted to solve was:

- Given a “base activity”, does there exist any “new activity” that satisfies the "agent/activity constraint"?

This is a constraint satisfaction problem (CSP). The most straight forward technique to search for a solution(s) for CSPs is generate-and-test [Shoham 94].

The generate-and-test assigns values to the variables in a way that is consistent with all the constraints, or determines whether or not such assignment exists. In general, this technique is a conjunction of two routines:

- generator (the first routine) enumerates a possible combination of variable values.
- tester (the second routine) examines the values of variables one by one to see whether they satisfy all the constraints.

  If not, then execution backtracks to the generator which generates another possible combination. This continues iteratively until the tester finds a solution (an instantiation of variables which satisfies the constraints), or until the generator has exhausted all the possible combinations.

In order to employ the generate-and-test technique to answer the competency question:

Given a “base activity”, does there exist any “new activity” that satisfies the "agent/activity constraint"?

we have to address the following issues:

1. What is the constraint that should be satisfied?
2. What is the variable set (i.e. the set of variables to which the generator should assign value) that characterizes any new activity?
3. What is the domain (possible values) of each variable?

Following we discuss these issues.

1. **What is the constraint that should be satisfied?**

   In the competency question, we want to find a new activity that satisfies the "agent/activity constraint". From chapter 4, we know the "agent/activity constraint" is:

   Given that *p* is an activity, for all subactivities of *p*:

   if *a2* is a subactivity which uses the information, *inf*, and the performing agent of *a2* is *ag2*,

   if *a1* is a subactivity which produces the information, *inf*, and the performing agent of *a1* is *ag1*,

   then:

   ag1 and *ag2* are the same, or

   ag1 and *ag2* are a team, or

   there is another subactivity such as *a3* which uses the information, *inf*, and *ag3* who is the performing agent of *a3* is the same as *ag2*.

2. **What is the variable set (i.e. the set of variables to which the generator should assign value) that characterizes any new activity?**

In the competency question, we intend to find a “new activity” which satisfies the "agent/activity constraint". As we remember from chapter 4, a “new activity” is specified by its subactivities and performing agents. This requires that the variable set that characterizes the new activity should contain the new activity’s subactivities and their performing agents. However, for the following reasons, we can prune the variable set to a smaller set that only contains the performing agents of those subactivities which either produce or use information.

First, since the subactivities of “new activity” and its “base activity” are exactly the same[1] and the subactivities of the “base activity” are given, the variable set can only consist of the new activity’s performing agents.

Second, since the performing agents of subactivities which neither use nor produce information do not participate in the "agent/activity constraint" (see step 1), we can further refine the variable set so that it only contains the agents of those subactivities which either produce or use information.

3. **What is the domain (possible values) of each variable?**

We assume that the domain of all the variables are finite, explicitly enumerated and the same. The user specifies this domain. An example for this domain set is *{Mike, Mark, John}* which means the performing agent of each subactivity which either produces or uses information can be *Mike*, *Mark* or *John*.

1. We know this from chapter 4 (page 69 and page 73).

6.1.2 Algorithm

Using the above variable set, domain of variables, and constraint, the following algorithm finds answer(s) for the competency question.

1. Given a base activity *P*, establish the set *LAC* with the following properties:

   Each member of *LAC* is a subactivity of *P* which either produces the information used by another subactivity or uses the information produced by another subactivity.

   In the Prolog program, this step is performed by *make_inf_dep_set(P, LAC).*

   Here a generate and test cycle begins.

2. Let *LAG* denote the domain set; i.e. the set which contains all the possible values for the agents of those subactivities which either use or produce information. Generate an instantiated list *(G)* in the following way:
   - *G* is a vector whose elements are members of *LAG. G* can contain repeated elements.
   - The *i-th* element of *G* represents the performing agent of the *i-th* element of *LAC*. For this reason, *G* has the same length as *LAC*.

   For instance, if *LAG = {Mike, Mark, John}* and *LAC = {Producing product specification, Design}*, then one possible *G* is *{Mike, Mike}*.

   In the Prolog program, the combination of *same_length(G, LAC)* and *generate(G, LAG)* performs this step.

3. Test if the instantiated *G* satisfies the constraint. If yes, *G* is a solution; otherwise backtrack to generate another *G.*

   In the Prolog program, the test step is performed by *not(violate(LAC,G))* and backtracking is automatically performed by Prolog.

4. Check to see if all the possible combinations have been generated. If yes, terminate. If no, backtrack to step 2 to generate another combination. Prolog performs this step automatically.

   It should be mentioned that *P, G* and *LAC* together specify the structure of "new activity". The subactivities of "new activity" are the same as *P.* Its performing agents are specified by *G* and *LAC*; i.e. the *i-th* element of *G* is the performing agent of that subactivity which is the *i-th* element of *LAC.*

6.1.3 Prolog program

Using the above algorithm and notation, the top level of our Prolog program (implemented in Quintus Prolog) is:

agent_assignment(P,LAC,G):-

make_inf_dep_set(P, LAC), make_agent_set(LAG),
same_length(G, LAC), generate(G, LAG), not(violate(LAC,G)).

Abstractly the program works as follows:

- The program is invoked by calling *agent_assignment(P,LAC,G).*

  P should be given.

  The program returns "no" if Prolog fails to find any solution. This can be because of one of the following reasons:

  - No subactivity of *P* produces or uses information, (or the information used or produced by the *P*'s subactivities is not defined in the model).
  - The domain set, *LAG*, (that contains the possible values of agents) is empty.

The program returns an instantiated *G*, if a solution (i.e. an agent assignment which satisfies the constraint) is found.

The *i-th* member of the instantiated *G* is the assigned value for the performing agent of the *i-th* subactivity listed in *LAC*. *LAC* is established by the program, as described below.

- Given *P, make_inf_dep_set(P, LAC)* builds a set *(LAC)* from the subactivities of *P* which produce or use information. *LAC* does not contain repeated elements.
- The possible values of agents (the members of the domain set) are specified by the user, one by one. For instance:

  agent(Mark).
  ...
  agent(John).

  where *Mark,...,John* are constants.

  make_agent_set(LAG) is a utility program whose only role is to make a set *(LAG)* from these given individual constants. Thus, for our example *LAG* will be *{Mark,..., John}*.

- The combination of *same_length/2* and *generate/2* is the generator of all possible *Gs*, including those *Gs* that are solutions (i.e. will cause the tester to become true) and those *Gs* that are not solutions (i.e. will cause the tester to fail).

  same_length(G, LAC) is one of the built-in Quintus Prolog library predicates. It constructs a list *(G)* which has the same length as *LAC*.

  generate(G, LAG) instantiates the members of this list *(G)*, using the possible values of agents (listed in *LAG)*.

- *(violate/2)* or more precisely the negation of *(violate/2)* is the tester of the program. Overall, it checks whether the generated *G* satisfies the constraint. The format of the tester (i.e. negation

of *(violate/2))* is the result of transforming the constraint from FOL into Prolog. More details about how to translate the FOL constraint into Prolog can be found in appendix A.

The commented sub-programs are listed in appendix B (section B.3.6).

6.2 Pre-order Management Process (PMP)

One of the goals of this chapter is to illustrate the application of Process Integration advisor in the area of process design. For that, we focus on a hypothetical process, “Pre-order Management Process” (PMP). In this section, we introduce PMP and describe its subactivities. In the next section, we perform an analysis of PMP, using the Process Integration advisor.

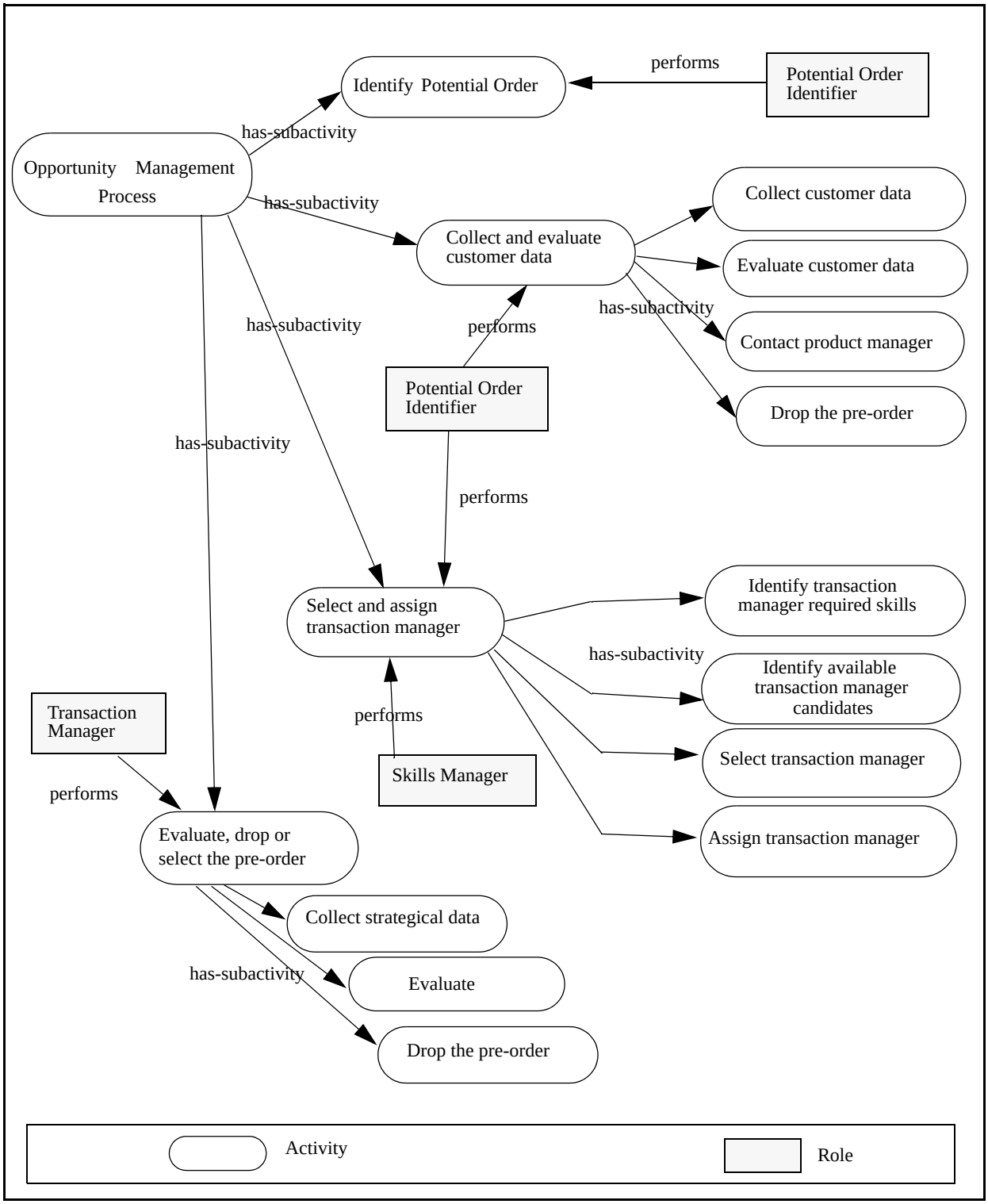
6.2.1 An overview of PMP

HC[1] Ltd. designs, implements and customizes a variety of telecommunication products and services. Pre-order Management Process (PMP) is one of the HC’s processes. The purpose of PMP is to identify whether or not there is a reasonable chance to make an acceptable profit from a specific potential order. In summary, PMP identifies a potential order, validates the customer’s business needs, selects and assigns a “transaction manager”, evaluates the pre-order from the HC’s viewpoint, and finally determines whether or not HC should pursue the pre-order. The pre-order, if selected by PMP, is then passed to another process; i.e. Order Management Process. Here, our focus is only on PMP.

Figure 5 on page 112 displays the PMP subactivities and roles of performing agents. In the next sections, we describe these subactivities, in more detail.

1. An acronym for Hypothetical Company.

FIGURE 5. An overview of PMP subactivities



6.2.2 PMP subactivities

PMP has the following subactivities:

1. Identify potential order
2. Collect and evaluate customer data
3. Select and assign “transaction manager”
4. Evaluate, drop or select the pre-order

6.2.2.1 Identify potential order

Identify potential order is the first activity of PMP. It starts when one of the employees of HC is informed that a customer has a need to which HC might be able to respond. The role of such an employee is the “potential order identifier”.

The “potential order identifier” is and will remain the contact point for the customer until a “transaction manager” is assigned to the pre-order.

6.2.2.2 Collect and evaluate customer data

The “potential order identifier” collects the customer’s data, including the customer’s business requirements, budget, requested start date, and requested delivery date.

The “potential order identifier” uses a specific set of criteria to evaluate the customer’s needs, based on the collected data. Table 6 summarizes the possible outcomes of the evaluation activity. In each row, the first cell presents an outcome and the second cell presents the activity which is enabled[1] by that outcome.

1. “An activity is enabled by that outcome” means the outcome is the precondition of that activity.

TABLE 6. The possible outcomes and the activity enabled by each outcome

#	Outcome	Activity (enabled by the outcome)
1	The pre-order is promising; i.e. HC is likely to meet the customer's needs, and thus the "potential order identifier" will proceed with the pre-order.	Select and assign "transaction manager"
2	The pre-order is not promising; i.e. HC is not likely to meet the customer's needs, and the "potential order identifier" decides to terminate processing the pre-order.	Drop the pre-order
3	The pre-order does not seem promising, i.e. HC is not likely to meet the customer's needs, but the "potential order identifier" still decides to proceed with the pre-order.	Contact the "product manager"

6.2.2.3 Select and assign "transaction manager"

If the result of the activity "evaluate customer data" is #1 or #3 (see the first column of table 6 on page 114), then an agent should be assigned as the "transaction manager". The "transaction manager" is an important role. The assigned agent to this role has the following responsibilities:

- From the time the "transaction manager" is assigned and introduced to the customer, s/he will be the contact point for the customer.
- S/he has to evaluate the pre-order from the HC's point of view and decides whether or not it is worth while pursuing. See section 6.2.2.4 on page 116.
- If the pre-order is selected then the "transaction manager" will also be responsible for the next process; i.e. Order Management Process comprising of design, implementation and delivery of the order to the customer.

The subactivities of Select and assign "transaction manager" are:

- Identify "transaction manager" required skills
- Identify available "transaction manager" candidates
- Select "transaction manager"

- Assign “transaction manager”

Following, we describe these subactivities.

- **Identify “transaction manager” required skills**

  As mentioned above, if the pre-order is selected then the “transaction manager” will be responsible for the design, implementation and delivery of the order to the customer. For this reason, s/he should have the technical knowledge required to understand the product and service, requested by the customer. As the result of this activity (identify “transaction manager” required skills), the required skills for the “transaction manager” are identified.

  In more detail,

  using the customer’s business requirements and skill templates, the “potential order identifier” recognizes the required skills for the “transaction manager”, sends the result to one of the skills managers, and requests that the “skills manager” specifies the available HC’s employees who can be selected as the “transaction manager”.

- **Identify available “transaction manager” candidates**

  The “skills manager” who receives the request and the result (which states the required skills for the “transaction manager”), performs one of the following tasks:

  1. If the “skills manager” (for any reason) can not identify available “transaction manager” candidates, s/he documents the reason(s) and sends the request to another “skills manager”.
  2. S/he uses the databases, containing the HC’s employees skills, to identify those employees who have the required skills to become the “transaction manager”. S/he verifies the availability of these employees and documents the result. S/he sends the result, i.e. the available “transaction manager” candidates, to the “potential order identifier”.

- **Select “transaction manager”**

  When the “potential order identifier” receives the document containing the available “transaction manager” candidates, s/he selects one of the candidates and sends the result to the “skills manager”.

- **Assign “transaction manager”**

  The “skills manager” informs the employee who has been selected as the “transaction manager” (by the “potential order identifier”), and records the assignment.

6.2.2.4 Evaluate, drop or select the pre-order

The assigned “transaction manager” evaluates profitability, and the accompanying risks of the pre-order. S/he either selects the pre-order for pursuing or decides not to proceed with it. Under both conditions, the customer is informed of the result. The former condition enables the other process, i.e. Order Management Process.

6.3 The Process Integration advisor

We encapsulated the implemented logical models of agent/activity design strategies and design validation into a software tool which we call the Process Integration advisor. The input to this advisor is a process model (represented in TOVE). Answering a total of six questions, the Process Integration advisor evaluates the input process from the aspects of “dangling information”, “case management”, “changeable agent assignments”, and “agent/activity design strategies”. These questions are:

1. Is there a piece of dangling information in this process?
2. Is there a time when no “case manager” exists for this process?
3. Is there a time when a “case manager” exists but s/he is unknown by the customer?

4. Is there a time when an agent should perform an activity in the process and the case manager of the process does not know about it?
5. Is there any activity which can change the assignment of an agent to a role or to a subactivity of this process?
6. What is the redesigned process(es) which satisfies the “agent/activity design strategies”, leading to minimal agent setup time?

On the basis of the advisor’s answers to these questions, the designer can analyze the input process. The questions should be asked in the form of Prolog queries that are listed in appendix B (section B.3.13).

6.4 Analysis of PMP

In this section, we demonstrate the application of our work in enterprise design. For that, we apply the Process Integration advisor to the PMP TOVE[1] model and analyze PMP, based on the results.

6.4.1 Summary of results

Table 7 on page 118 summarizes the results of applying the Process Integration advisor to PMP. In each row, the first cell specifies the name and a brief description of a category of expertise (which is part of the advisor) and the second cell summarizes the results of applying that expertise to PMP.

1. Please note that we have modeled PMP, employing the TOVE ontologies as presented in the previous chapters. The TOVE model of PMP is found in appendix B (sections B.3.7 and B.3.8).

TABLE 7. Applying the Process Integration advisor to PMP; Summary of results

Expertise category	Evaluation
Dangling information The contribution of the information produced by an activity to the continuation of the other activities should be recognizable.	The problem is: the information produced by the activity of “contact product manager” (see table 6 on page 114) is not used by any other activity.
Case management A contact point should exist in all steps of a transaction and the agent who performs this role should be known by the transaction customer.	The strong point is: in all steps of PMP, there is a contact point for the customer. The “potential order identifier” is the contact until the “transaction manager” is assigned and introduced to the customer. The problem is: it is not identified in which step of process, the customer knows that “potential order identifier” is the contact.
This contact point (to which we refer as “case manager”) should be able to trace the transaction. Traceability indicates that the case manager can respond to the customer’s questions, e.g. queries about the transaction status.	The strong point is: the “case manager” of PMP can always know to which agent the transaction is assigned.

TABLE 7. Applying the Process Integration advisor to PMP; Summary of results

Expertise category	Evaluation
Changeable agent assignments The process design should allow the modification or cancellation of "agents assignments".	The strong point is: the reassignment of "skills managers" is defined. The problem is: once a "potential order identifier" or a "transaction manager" is assigned, there does not exist any activity to change the assignment.
Agent/activity design strategies The expertise identifies a variety of agent assignments that lead to minimal process agent setup time.	The following agent assignments improve the PMP agent setup time. One agent (with the help of a computer program) should perform all the PMP's subactivities. One agent (with the help of a computer program) should perform the following subactivities: Identify potential order, Collect and evaluate customer data, and Select and assign "transaction manager". However "Evaluate, drop or select the pre-order" can be performed by another agent.

6.4.2 Results

The next sections describe the results of analyzing the PMP design, in the following format.

- With respect to dangling information, case management, and changeable agent assignments, the results are presented in terms of PMP's strengths and/or problems, and the recommendations to improve the problems. More details and insights will be found in the elaboration.

- In regard to agent/activity design strategies, the results are presented in terms of design alternatives. Again, more details and insights will be found in the elaboration.

The entire session of executing the advisor is recorded and can be found in appendix B (section B.3.2).

6.4.2.1 Dangling information

The term “dangling information” refers to the information which is produced by an activity, but not used by any other activity. Existence of “dangling information” might be an indicator of incompleteness of the process definition.

- **Problem**

  P1. The information obtained by the “potential order identifier” through the activity of “contact product manager” is not used by any other activity.

- **Recommendation**

  R1. The contribution of the information produced by the activity “contact product manager” to the other activities should be defined.

- **Elaboration**

  E1. In the “Collect and evaluate customer data”, “potential order identifier” decides to proceed with the pre-order or not, (see section 6.2.2.2). If the result of evaluation is not promising and the “potential order identifier” still wants to proceed, s/he has to contact the “product manager”. The use of product manager’s advice is not included in the process definition and should be added. The following two examples illustrate some possible usage of this information.

  - The “potential order identifier” will proceed, only if the “product manager” approves.
  - The “potential order identifier” does not need the product manager’s approval to proceed, however, s/he uses the product manager’s advice to re-evaluate his/her decision.

6.4.2.2 Case management

For each transaction moving through a process, one agent should be assigned as the contact point for the customer, and the agent who is assigned to this role should be known by the customer. In the previous chapter, we referred to this role as “case manager”. The existence of “case manager” is an indicator that the enterprise is capable of capturing the customer’s demands through all steps of the transaction.

The “case manager” should be able to trace the transaction; i.e. identify which agent is assigned to which step of the transaction. This indicates that the “case manager” can respond to the customer’s questions about the transaction status.

- **Strengths**

  S1. Through all the steps of PMP, a “case manager” exists.

  S2. The case managers of PMP; i.e. the “potential order identifier” and later on the “transaction manager”, can trace all the transaction’s agents assignments.

- **Problem**

  P1. It is not stated in which step of PMP the “potential order identifier” is introduced to the customer as the “case manager”.

- **Recommendation**

  R1. The activity by which the customer recognizes the “potential order identifier” as the “case manager” should be clearly defined in the process.

- **Elaboration**

  E1. Given a scenario of PMP, the advisor fails to find even one situation in which a “case manager” does not exist. The reasons are:

- We modeled that the “potential order identifier” is the first “case manager”. S/he is also the one who initiates the PMP transaction. Therefore in the beginning of the PMP, a case manager exists.
- We modeled that the “potential order identifier” remains as the case manager until the “transaction manager” is assigned and introduced to the customer. From that point, the “transaction manager” becomes and remains as the “case manager”.

E2. PMP starts when the “potential order identifier” is informed about a potential order and initiates the transaction. The advisor fails to find the situation when the identifier and his/her role (as the contact point) are known by the customer.

In order to perform the activity of “Collect and evaluate customer data”, the identifier has to communicate with the customer. Thus we can assume and model that somewhere along this activity, the “potential order identifier” might describe his/her role to the customer. The problem with this assumption is: it is implicit. If this is a correct assumption, then it should be explicitly added to the process definition.

E3. As we will see in section 6.4.2.3, the activity which results in changing a “transaction manager” is missed and should be included in the process definition. Once included, then it is recommended that the activity which causes the customer to know the new “transaction manager” also be specified.

E4. For the following reasons, the advisor can deduce that the “case manager” can always know the information about the assignments of agents to activities.

- The ways that an agent can know the information are modeled; an agent can know the information if s/he has a read access to a document in which that information is written.
- The fact that “potential order identifier” and the “transaction manager” have always read access to the pre-order record is modeled.

- The ways that an agent becomes or remains as a “case manager” are modeled.
- The fact that the information about the assignments of agents to activities is documented in the pre-order record is modeled. Also, the fact that this information is documented in the pre-order record, as the effect of assignment and reassignments activities, is modeled. This entails that there is no time gap between assigning or reassigning an agent and writing it on the record.

Thus, the advisor fails to find a situation when a process obligation exists but it is not on record or to find a situation when a process obligation is on record but can not be known by the “case manager”.

6.4.2.3 Changeable agent assignments

- **Strength**

S1. The PMP definition includes the activity of reassigning a “skills manager”. This activity cancels the assignment of the previous agent and at the same time assigns a new agent as the “skills manager”.

- **Problems**

P1. No activity is defined to cancel the assignment of an agent as the “potential order identifier”.

P2. Once an agent is assigned as the “transaction manager” (see page 116), no activity is defined which can cancel this assignment.

- **Recommendation**

R1. The activities which can cancel the assignments of “transaction managers” and “potential order identifiers” should be defined and included in the PMP model.

6.4.2.4 Agent/activity design strategies

- **Design alternatives**

  Following are the design options, ordered based on their effect on improving PMP agent setup time:

  1. One agent (with the help of a computer program) should perform all the subactivities.
  2. One agent (with the help of a computer program) should perform "Identify potential order", "Collect and evaluate customer data", and "Select and assign "transaction manager"". However the activity "Evaluate, drop or select the pre-order" can be performed by another agent.

- **Elaboration**

  E1. With regard to the skills required to perform activities, it is very likely that the first design alternative is rejected.

  Any HC employee who is informed about a potential order can initiate PMP, evaluate the customer data, and identify the required skills for the "transaction manager" (with the help of skill templates).

  On the other hand, the "transaction manger" should have enough technical knowledge about the customer's business requirements to evaluate, drop or select the pre-order and later on manage the order (if the pre-order is selected). The required skills for the transaction managers directly depend on the requested product and/or service by the customers. The HC's products/services vary and so do the skills needed by their transaction managers. Due to this variety, it is less likely that the same employee who initiates the PMP transaction, (even with the help of computer programs) will be capable of handling a variety of fairly complicated products and services. Thus the first alternative, stating that one agent performs all the PMP's subactivities, is very likely to be rejected.

E2. With regard to the skills required to perform activities, the second design alternative is more likely to be accepted.

This alternative entails that the two subactivities: 'Identify available "transaction manager" candidates', and 'Assign "transaction manager"' are performed by the same agent who already performed the previous activities; i.e. 'Identify potential order', 'Collect and evaluate customer data', and 'Identify "transaction manager" required skills'.

In order to perform 'Identify available "transaction manager" candidates', the performing agent should know how to use data bases. If we assume that s/he has (or can learn) this skill, (which is a reasonable assumption), then with respect to the required skills to perform activities, it is likely that the second design alternative is accepted.

Above, we evaluated the design alternatives, on the sole basis of skills constraint. It should be mentioned that the acceptance of any alternative can as well depend on other constraints such as the HC's empowerment policies. Such constraints are outside of the scope of this work.

6.5 Summary

1. In the first section of this chapter, we discussed the implementation of the logical model of agent/activity design strategies in Prolog. In the current implementation, the Prolog program proposes alternative agent assignments which would lead to minimal agent setup time. The user can explore the alternatives and choose a subset of them. The program can be enhanced to support the following task in the future.

   Proposing alternative agent assignments that lead to minimal agent setup time and at the same time satisfy agents' capability constraint.

   This will be an implementation of the extended FOL model which was presented in section 4.4 on page 79.

2. One of our goals in this chapter was to demonstrate the use of our research in the area of process design. We achieved this goal by applying the Process Integration advisor (which embeds the logical models of agent/ activity design strategies and design validation) to the TOVE model of a hypothetical process; i.e. Pre-order Management Process (PMP).

Chapter 7

Summary and future work

This chapter summarizes the thesis and highlights those aspects of the work which can be improved.

7.1 Summary of the thesis

The goal of the thesis was to transform process design expertise into an engineering discipline where its principles can be consistently applied across various scenarios. Towards this goal, we performed the following steps:

1. Demonstrate the heuristic nature of process design
2. Identify the dominant emerging theme from the heuristics
3. Create an analytical model of agent setup time
4. Develop the logical model of agent/activity design strategies
5. Develop the design validation model
6. Integrate the FOL models into the Process Integration advisor
7. Demonstrate the application of our work

In the following sections, we briefly discuss each step, its beneficial effects, and also describe the connections between these steps.

7.1.1 Demonstrate the heuristic nature of process design

Improving process design and searching for new process solutions are mostly based on rules of thumb, i.e. heuristics. We reviewed several process design heuristics and concluded that:

- The heuristics are useful at the early stage of design. They introduce some attributes of successful processes and can stimulate companies to look for new ways of design.
- The heuristics are ambiguous and unreliable. These characteristics prevent them to be consistently applied across various processes. We needed to develop formal models that can demonstrate the underlying principles of the process design expertise and enable the consistent application of the practice across many enterprises.

7.1.2 Identify the dominant emerging theme from the heuristics

We identified that a large number of heuristics propose different ways of assigning agents to perform activities. This group includes the heuristics which suggest:

- assigning an individual (with the help of a computer program) or a team to perform a set of activities, or
- shifting the responsibility of performing an activity from an individual or a group to another.

We focused on this group and decided to describe their underlying principles, through using an analytical model of agent setup time.

7.1.3 Create an analytical model of agent setup time

We built an analytical model that highlights the agent setup time and its major components. Following is the outline of our model.

7.1.3.1 An overview of the analytical model of agent setup time

In order to develop our analytical model of agent setup time, we assume that there are two activities, Ac_i and Ac_j. These activities are respectively performed by two different agents, Ag_i and Ag_j. Activity Ac_j uses the information contained in the transaction, T_{ij}, caused by the activity Ac_i.

$\{Ac_i, Ag_i\}$ --- T_{ij} ---> $\{Ac_j, Ag_j\}$

There exists agent setup for Ag_j, if Ag_j requires some amount of preparation before activity Ac_j can be performed.

Equation 64 presents our analytical model of agent setup time.

$$PT_j(T_{ij}) = PT_j(Necessary(T_{ij})) + PT_j(Unnecessary(T_{ij})) + PT_j(Missing(T_{ij})) + PT_j(SKn(Ag_j, Ac_j)) \quad (EQ\ 64)$$

In this model, $PT_j(T_{ij})$ is the agent setup time, $PT_j(Necessary(T_{ij}))$ is the required time to obtain the necessary information contained in the transaction (T_{ij}), $PT_j(Unnecessary(T_{ij}))$ is the time needed to separate the unnecessary information, $PT_j(Missing(T_{ij}))$ is the time required to gather the missing information, and $PT_j(SKn(Ag_j, Ac_j))$ is the required time to acquire new skills in order to perform Ac_j.

7.1.3.2 The application of our analytical model of agent setup time

The model allows us to discuss different strategies that can improve agent setup time. Table 8 on page 133 summarizes the results of this discussion. The agent setup time and its components are presented in the first row, based on the same notations we employed in the model (see the description of EQ 64 on page 131). The strategies are listed in the second column. Each cell presents the effect of a strategy on the agent setup time or one of its components; the strategy can either “eliminate”, or “reduce” the agent setup time (or one of its components) or has “no effect” on it. The first column classifies the strategies under two groups:

- **Manufacturing process strategies**

  Manufacturing process strategies structure the work so that an agent receives the transactions which are either identical or within a predefined range. In these situations the amount of context specific information (contained in the received transaction, T_{ij}) is small and thus the agent setup time $(PT_j(T_{ij}))$ is trivial.

- **Agent/activity design strategies** (highlighted cells in table 8 on page 133)

  It might be the case that one agent receives different transactions. In this case, the agent needs to obtain a certain amount of context specific information, prior to performing the activity. For instance, a designer needs to understand the product requirement before performing the design. In these situations, a group of agent assignment strategies can reduce the agent setup time. We refer to this group as “agent/activity design strategies”.

7.1.3.3 The positive aspects of our agent setup time model

- The model enabled us to identify a variety of strategies that can improve the agent setup time; specifically the agent/activity design strategies which represent the underlying principles of our focal group of heuristics (presented in section 7.1.2).

7.1.3.4 The limitation of our agent setup time model

- The analytical model of agent setup time model neither formally defined each strategy, nor provided a foundation for a reasoning system to explore various designs and to select the ones which can improve the agent setup time. To address these inadequacies, we focused on the agent/activity design strategies and developed a logical model of them.

TABLE 8. **The effects of process and agent assignment strategies on agent setup time**

Strategies		**$PT_j(T_{ij})$[a]**	**$PT_j(Necessary(T_{ij}))$[b]**	**$PT_j(Unnecessary(T_{ij}))$[c]**	**$PT_j(Missing(T_{ij}))$[d]**	**$PT_j(SKn(Ag_j,Ac_j))$[e]**
Man.[f]	Batch orders (items within the batch)	eliminate	eliminate	eliminate	eliminate	eliminate
	Transfer line	eliminate	eliminate	eliminate	eliminate	eliminate
	Products that use common components	reduce	reduce	reduce	reduce	reduce
	Standard interfaces between the components of assembly typed products	reduce	reduce	reduce	reduce	reduce
	Computer controlled equipment	reduce	reduce	reduce	reduce	reduce

TABLE 8. **The effects of process and agent assignment strategies on agent setup time**

Strategies		$PT_i(T_{ij})$[a]	$PT_i(Necessary(T_{ij}))$[b]	$PT_i(Unnecessary(T_{ij}))$[c]	$PT_i(Missing(T_{ij}))$[d]	$PT_i(SKn(Ag_j,Ac_i))$[e]
Agent/ activity design	Assign one agent to perform Ac_i and Ac_j[g]	reduce	eliminate	eliminate	eliminate	no effect
	Assign one agent with the help of a computer program to perform Ac_i and Ac_j[g]	reduce	reduce	reduce	reduce	no effect
	Assign a team to per-form Ac_i and Ac_j[g]	reduce	no effect	reduce	reduce	reduce
	Assign one agent to perform Ac_{j-1} and Ac_{j-2}[h]	reduce	reduce	reduce	reduce	reduce

a. Agent setup time
b. The required time to understand necessary information.
c. The required time to separate unnecessary information.
d. The required time to gather missing information.
e. The required time to learn a new skill.
f. Manufacturing process strategies
g. Ac_j is the activity which uses the information contained in the transaction, caused by the activity Ac_i.
h. Ac_{j-1} and Ac_{j-2} are two activities that use the information contained in the same transaction.

7.1.4 Develop the logical model of agent/activity design strategies

We developed a First Order Logic model of the agent/activity design strategies.

7.1.4.1 The benefits of the logical model of agent/activity design strategies

- The logical model of agent/activity design strategies is important, because 1) it formally defines these strategies through a set of logical axioms, 2) these axioms provide a foundation for a reasoning system to search and find alternative designs which can answer the following question:

  Given a process, what is the redesigned process which satisfies the agent/activity design strategies leading to minimal agent setup time?

  This characteristic enables the model to be employed in designing and redesigning processes.
- The model is extendible; i.e. we showed how to add a new class of constraint, agents' capability, to the model without having to redesign the entire representation. The extended model is capable of answering the following question:

  Given a process, what is the redesigned process which satisfies the "agent/activity design strategies" and at the same time satisfies the "agents' capability constraint"?

  Where satisfying the "agents' capability constraint" means, if an agent is assigned to an activity, then that agent should have the skills required to perform that activity.
- The model is precise, i.e. it formally defines the employed terminology. This addresses the problem of ambiguity in the expertise.

7.1.5 Develop the design validation model

We also developed a First Order Logic model of three categories of BPR expertise; i.e. dangling information, case management and changeable agent assignments. These categories of expertise can be employed to validate an existing process design. For this reason, we referred to this logical model as the design validation model. Table 9 on page 137 summarizes the design validation

model. In each row, the first cell presents the name of the category of expertise and a brief description. With respect to each category, the model is able to answer a question(s). The second cell specifies this question(s).

7.1.5.1 The benefits of our design validation model

- The problem solving capability of the model is explicit. The second column of Table 9 presents the questions that the model is able to answer.
- Answering a total of five questions (see the second column of Table 9), the design validation model enables a reasoning system to determine whether or not there are missing elements in the process design. These missing elements are: an activity which should use a specific piece of information is missing; there is no case manager role for the process at a certain time; a piece of information which is required by the case manager or the process customer is left out from the process definition; and an activity that should cancel a specific agent assignment is not defined.

  A missing element is not necessarily a sign of a problem. A problem develops when the omission is due to a design oversight rather than a design decision.

  A process designer can use the design validation model to identify these missing parts. With respect to each omission the designer has two choices, either altering the process design to eliminate the omission or to accept the design as it is.

TABLE 9. **Summary of the design validation model**

BPR expertise	The question(s) that the model answers
Dangling information The contribution of the information produced by an activity to the continuation of the other activities should be recognizable. Existence of "dangling information" might be an indicator of an incomplete process definition.	Given a set of activities, does there exist a piece of information that is produced by an activity and not used by any other activity?
Case management For each transaction moving through a process, one agent should be assigned as the contact point for the customer and the customer should know this agent. We recognize this agent as the "case manager". This is an indicator that the enterprise is capable of capturing the customer's demands through all the steps of the transaction The "case manager" should be able to trace the transaction. Traceability indicates that the "case manager" can respond to the customer's questions, in specific the questions about the transaction status.	Given an activity, is there a time point when no agent is assigned to the "case manager" role? is there a time point when an agent has the "case manager" role but this agent is not known by the customer? is there a time point when an agent has the obligation to perform one of the subactivities but the "case rmanager" does not know it?
Changeable agent assignments The process design should allow the modification or cancellation of "agents assignments".	Given an activity, its subactivities and a set of agent constraints in the form of roles and process obligations, once a constraint is imposed on an agent, does there exist any subactivity which can remove this constraint?

7.1.6 Integrate the FOL models into the Process Integration advisor

We implemented the logical models in Quintus Prolog. We encapsulated the implemented models into a software tool to which we refer as the Process Integration advisor.

7.1.6.1 The benefits of the Process Integration advisor

- Incorporating the design validation model, the advisor assists designers to improve the design of an existing process or to refine the design of a new process before implementing the design.
- Embedding the logical model of agent/activity design strategies, the advisor automatically generates alternative agent assignments that achieve minimal agent setup time and thus can be employed to design or redesign processes, providing that the design perspective is improving the agent setup time.
- The Process Integration advisor is a software tool which embeds the definitive logical models of BPR expertise. This allows the users to understand the definition of the employed terminology.

  Since the embedded models are characterized by the questions they can answer, the analysis tasks that the advisor can perform are clear. For this reason, the users know what they should or should not expect from the advisor and determine whether or not the advisor can address their needs.

7.1.7 Demonstrate the application of our work

In order to demonstrate the practical use of our research in the area of process design, we applied the Process Integration advisor to a sample process; i.e. Pre-order Management Process (PMP). Using the results, we identified the PMP’s problems and strengths and provided a set of recommendations to improve its design.

7.2 Future work

In this section, we introduce those areas of the work which can be improved.

7.2.1 Analytical model of agent setup time

- We developed an analytical model of agent setup time. The model which characterized some important components of agent setup, was based on two activities and two agents. Further work should be done to elaborate each component of agent setup time and extend the model with respect to a larger group of activities and agents. This might result in recognition of other strategies that can decrease the agent setup time.

7.2.2 Ontologies

- TOVE information ontology should be further developed to represent:
  1. information at different levels of abstraction
  2. the relationship between the information and its storage
  3. different classes of information producing and information using actions.

7.2.3 Implementation

- We implemented the FOL model of agent/activity design strategies. The program currently finds those agent assignments which achieve minimal agent setup time. We can enhance it to find those agent assignments that lead to minimal agent setup time, and at the same time satisfy the agents capability constraint. This will be the implementation of the extended model that was discussed in section 4.4 on page 79.
- We used the “generate-and-test” technique to implement the FOL model of agent/activity design strategies in Prolog. The generator enumerated a possible combination of values for agents. The tester examined the combination to see if it was a solution; i.e. the values of

agents satisfied the constraint. Our program was inefficient because many combinations were generated that had no chance of being solutions. We can improve the program by pushing the tester inside the generator as deeply as possible. This means instead of testing the complete combination, each agent should be checked as it is being assigned a value.

- In this thesis, we did not direct our efforts towards the Process Integration advisor's user interface, its ability to capture the model and translate the utilized terminology to different users. These capabilities which are independent of the advisor's reasoning tasks need to be developed.

7.2.4 Developing other formal models of BPR

- Our work is one step towards formalization of the business process reengineering expertise. Much work still remains in discovering the other underlying principles of BPR and constructing formal models of them.

  One of our next steps can be formalizing the "Concurrency in information intensive processes" which was one of the emerging themes from the reviewed heuristics. We introduced the concept in the conclusion of chapter 2.

Appendix A

In this section, we demonstrate how a FOL sentence which has the similar structure to the "agent/ activity constraint" can be translated into a Prolog axiom. This will rationalize the format of tester routine *not(violate/2)* in our Prolog program (see chapter 6, section 6.1.3 on page 104).

A.1 Translating constraints from FOL into the PROLOG axioms

Consider (EQ 1):

$$(\forall inf,a1,a2,ag1,ag2)\ uses\_information(a2,inf,\ ag2) \wedge produces\_information(a1,inf,ag1) \supset (ag1 = ag2 \vee team(ag1,ag2)). \quad \textbf{(EQ 1)}$$

It states that for all the activities, if one activity uses the information which is produced by the other then their performing agents should be the same or a team. This sentence is a simplified version of the axiom that expressed "agent/activity constraint" (see chapter 4). The translation of this sentence from FOL into Prolog is described through the following steps:

1. We transfer (EQ 1) into its equivalent form in which the universal quantifiers are eliminated. Hence, we obtain:

   $\neg (\exists\ inf,a1,a2,ag1,ag2)\ uses\_information(a2,inf, ag2) \wedge produces\_information(a1, inf, ag1) \wedge \neg(ag1=ag2) \wedge \neg\ team(ag1,ag2).$ **(EQ 2)**

2. In (EQ 3), we introduce a new FOL predicate (i.e. *test*) in which the constraint is falsified:

   $test \equiv (\forall inf,a1,a2,ag1,ag2)\ uses\_information(a2,inf,ag2) \wedge produces\_information(a1,inf,ag1) \wedge\ \neg\ (ag1=ag2) \wedge \neg\ team(ag1,ag2).$ **(EQ 3)**

3. (EQ 3) is equivalent to the following Prolog axiom:

   test:- uses_information(a2,inf,ag2), produces_information(a1,inf, ag1), not(*ag1=ag2), not(team(ag1,ag2)).* **(EQ 4)**

4. Thus, the original sentence (EQ 1) can be represented in Prolog as:

   not(test). **(EQ 5)**

B.3 Files
B.3.1 all_thesis.log

Quintus Prolog Release 3.1.1 (DECstation, Ultrix 4.x)

2100 Geng Road, Palo Alto, California U.S.A. (415) 813-3800

%% thesis_ld_demo.pl compiled in module user, 7.017 sec 109,032 bytes
| ?- dangling_information(ACT, INF).

this query finds the dangling information
meaning- the information produced by an activity is not used
by any other activity in the transaction

ACT = contact_product_market_manager(_17473),
INF = contact_manager_advice(_17473) ;

no
| ?- no_contact(ACT).
This query finds the activities that when they occur-

no agent is assigned as the contact point for the customer
1 Please wait, Prolog is searching the data base
2 Please wait, Prolog is searching the data base
3 Please wait, Prolog is searching the data base
4 Please wait, Prolog is searching the data base
5 Please wait, Prolog is searching the data base
7 Please wait, Prolog is searching the data base
8 Please wait, Prolog is searching the data base
9 Please wait, Prolog is searching the data base
10 Please wait, Prolog is searching the data base
12 Please wait, Prolog is searching the data base
13 Please wait, Prolog is searching the data base
14 Please wait, Prolog is searching the data base
18 Please wait, Prolog is searching the data base

no
| ?- contact_unknown(ACT).
This query finds the activities that when they occur
the contact point exists but the customer does not know this contact point

1

1 Please wait, Prolog is searching the data base

ACT = identify_potential_order(trn) ;

2

2 Please wait, Prolog is searching the data base

ACT = collect_customer_data(trn) ;

3

3 Please wait, Prolog is searching the data base

4

4 Please wait, Prolog is searching the data base

5

5 Please wait, Prolog is searching the data base

7

7 Please wait, Prolog is searching the data base

8

8 Please wait, Prolog is searching the data base

9

9 Please wait, Prolog is searching the data base

10

10 Please wait, Prolog is searching the data base

12

12 Please wait, Prolog is searching the data base

13

13 Please wait, Prolog is searching the data base

14

14 Please wait, Prolog is searching the data base

18

18 Please wait, Prolog is searching the data base

18

18 Please wait, Prolog is searching the data base

no
| ?- contact_loses_agent_trace(ACT, AG, ROLE, TRN_CON, ACTD, S).

This query identifies the situations when an agent
is assigned to perform an activity but
the transaction contact can not know about it

Please wait, Prolog is searching the data base
7 Please wait, Prolog is searching the data base
8 Please wait, Prolog is searching the data base

9 Please wait, Prolog is searching the data base
10 Please wait, Prolog is searching the data base
12 Please wait, Prolog is searching the data base
13 Please wait, Prolog is searching the data base
14 Please wait, Prolog is searching the data base
18 Please wait, Prolog is searching the data base
18 Please wait, Prolog is searching the data base

no
| ?- change_assignment(pmp(trn), ASSGT, ACT).

this query finds those agent assignments for which
modification is defined

ASSGT =
has_process_obligation(identify_transaction_manager_candidates(_17044),skills_manager,ag1,_17044),
ACT = reassign_skills_manager(_17091,_17092,skills_manager,trn) ;

no
| ?- no_change_assignment(P, ASSGT).

this query finds those agent assignments for which
modification is not defined

P = _15817,
ASSGT = transaction_manager(ag2,_17020) ;

P = _15817,
ASSGT = potential_order_identifier(ag2,_17020) ;

no
| ?- agent_assignment(pmp(trn), LAC, G).

LAC =
[collect_customer_data(trn),collect_strategical_data(trn),contact_product_market_manager(trn),drop(trn),evaluate(trn),evaluate_customer_data(trn),identify_potential_order(trn),identify_transaction_manager_candidates(trn),identify_transaction_manager_required_skills(trn),select_transaction_manager(trn),verify_availablity_of_transaction_manager_candidates(trn),assign_transaction_manager(_17182,trn),assign_transaction_manager(_17615,trn)],
G =
[skills_manager,skills_manager,skills_manager,skills_manager,skills_manager,skills_manager,s

kills_manager,skills_manager,skills_manager,skills_manager,skills_manager,skills_manager,skil
ls_manager] ;

LAC =
[collect_customer_data(trn),collect_strategical_data(trn),contact_product_market_manager(trn),d
rop(trn),evaluate(trn),evaluate_customer_data(trn),identify_potential_order(trn),identify_transact
ion_manager_candidates(trn),identify_transaction_manager_required_skills(trn),select_transacti
on_manager(trn),verify_availablity_of_transaction_manager_candidates(trn),assign_transaction_
manager(_17182,trn),assign_transaction_manager(_17615,trn)],
G =
[potential_order_identifier,potential_order_identifier,potential_order_identifier,potential_order_i
dentifier,potential_order_identifier,potential_order_identifier,potential_order_identifier,potential
_order_identifier,potential_order_identifier,potential_order_identifier,potential_order_identifier,
potential_order_identifier,potential_order_identifier] ;

no
| ?- agent_assignment(select_and_assign_transaction_manager(TRN), LAC, G).

TRN = _15825,
LAC =
[identify_transaction_manager_candidates(TRN),identify_transaction_manager_required_skills(
TRN),select_transaction_manager(TRN),verify_availablity_of_transaction_manager_candidates(
TRN),assign_transaction_manager(_17184,TRN)],
G = [skills_manager,skills_manager,skills_manager,skills_manager,skills_manager] ;

TRN = _15825,
LAC =
[identify_transaction_manager_candidates(TRN),identify_transaction_manager_required_skills(
TRN),select_transaction_manager(TRN),verify_availablity_of_transaction_manager_candidates(
TRN),assign_transaction_manager(_17184,TRN)],
G =
[potential_order_identifier,potential_order_identifier,potential_order_identifier,potential_order_i
dentifier,potential_order_identifier] ;

no
| ?-

B.3.2 thesis_ld_demo.pl

```
%The required files for demo.
% Writer: Katayoun Atefi.
% Date: May 1997.

:- ['model.pl'].

:- ['scenario.pl'].
:- ['driver1.pl'].
:- [subactivity_def].
:- [library(strings)].
:- [library(sets)].
:- [library(lists)].
:- [library(basics)].
:- unknown(_,fail).

:- ['main_def.pl'].
:- ['dng.pl'].
:- ['trc.pl'].
:- ['chg.pl'].
:-ensure_loaded(stp).
```

B.3.3 dng.pl

```
%Dangling information expertise
% Writer: Katayoun Atefi.
% Date: May 1997.


:- multifile produces_information/2.
:- multifile uses_information/2.

/*
:- unknown(_,fail).
:- ['model.pl'].
*/

dangling_information(A, Inf) :-
      nl,
      print('this query finds the dangling information'),
      nl,
      print('meaning- the information produced by an activity is not used'),
      nl,
      print('by any other activity in the transaction'),
      nl,
      produces_information(A,Inf),
      \+(test(Inf)).

test(Inf):- uses_information(A, Inf).
```

B.3.4 trc.pl

```
% The Case management expertise.
% Writer: Katayoun Atefi.
% Date: May 1997.


:- dynamic holds/2.
:- multifile holds/2.
:- multifile holdsT/2.
:- multifile occursT/2.

/*
:- ['scenario.pl'].
:- [driver1].
:- [main_def].
*/

/*-------no_contact/1  returns those activities
              that when they occur no agent is defined as
              the contact point for the customer.

               This program also needs the driver  for its execution.
               The Situations in the scenario
               should be fully instantiated, otherwise  when the control
               is transferred to the driver the
               program does not work.
                ------------*/

no_contact(ACT):-
    print('This query finds the activities that when they occur-' ),
    nl, nl,
    print('no agent is assigned as the contact point for the customer'),
    nl,
    occursT(terminate(ACT), S), write(S),
    write('  Please wait, Prolog is searching the data base'), nl,
 \+ holdsT(agent_constraint(AG, contact(AG, TRN)), S).

/*_____________
    contact_un_known/1 returns the activities that at their
    occurrence a contact exists but not known by the customer.
    _____________________*/
```

```
contact_unknown(ACT):-
     print('This query finds the activities that  when they occur'),
     nl,
     print('the contact point exists but the customer does not know this contact point'),
     nl,
     occursT(terminate(ACT), S),
     holdsT(agent_constraint(AG, contact(AG, TRN)), S) ,
     holdsT(agent_constraint(CUST, customer(CUST, TRN)), S), nl, nl, write(S),
 \+ test_1(S) .

test_1(S):-
      nl, nl, write(S), write('  Please wait, Prolog is searching the data base'), nl,
      holdsT(knows(contact(AG, TRN), CUST), S).


/*----contact_loses_agent_trace/6 is a query which identifies---
          an agent  (AG) who has the role (ROLE),
          is assigned to  perform an activity (ACT) and the agent who is
         the Contact Point for the customer does not know about this assignment.
-----------*/

contact_loses_agent_trace(ACT, AG, ROLE, TRN_CON, ACTD, S):-
      nl,
      print('This query identifies the situations when an agent'),
      nl,
      print('is assigned to perform an activity but'),
      nl,
      print('the transaction contact  can not know about it'),
      nl,nl,write('      Please wait, Prolog is searching the data base'), nl,
      occursT(terminate(ACTD), S),
    /*This causes that S is instantiated*/

     holdsT(agent_constraint(AG, has_process_obligation(ACT, ROLE, AG, TRN)), S),
    /*This finds if at  S an  agent is assigned to perform an activity */

      holdsT(agent_constraint(TRN_CON, contact(TRN_CON, TRN)), S),
  /* This identifies who is the contact at S*/

       write(S), write('  Please wait, Prolog is searching the data base'), nl,

     \+ test_2(S).
```

```
test_2(S):-
  holdsT(can_know(has_process_obligation(ACT, ROLE, AG, TRN), TRN_CON), S).
  /* This checks if at S the contact  can not know who is assigned */

%=============================================


contact_traces(ACT, AG, ROLE, TRN_CON, S):-
 occursT(terminate(ACTD),  S),
    /*This causes that S is instantiated*/

     holdsT(agent_constraint(AG, has_process_obligation(ACT, ROLE, AG, TRN)), S),
   /*This finds if at S  an  agent is assigned to perform an activity */

      holdsT(agent_constraint(TRN_CON, contact(TRN_CON, TRN)), S),
     /* This identifies who is the contact at S */

     write(S), write('  Please wait, Prolog is searching the data base'), nl,
     holdsT(can_know(has_process_obligation(ACT, ROLE, AG, TRN), TRN_CON), S).
```

B.3.5 chg.pl

```
%The changeable agent assignments expertise.
% Writer: Katayoun Atefi.
% Date: May 1997.


:- dynamic holds/2.
:- multifile holds/2.
:- multifile has_isubactivity/2.

/*
:- [library(strings)].
:- unknown(_,fail).
:- ['model.pl'].
:- [main_def].
:- ['subactivity_def.pl'].

*/

/* The following Agent Constraints are defined: */

agent_constraint(AG, transaction_manager(AG, TRN)).
agent_constraint(AG, potential_order_identifier(AG, TRN)).
agent_constraint(AG, has_process_obligation(identify_transaction_manager_candidates(TRN),
skills_manager, AG, TRN)).

/*_____________________________________________________________*/

/*--------change_assignment/2 finds if any subactivity (ACT) of process (P)
 is defined to change an agent constraint (ASSGT).
-------------*/


change_assignment(P, ASSGT, ACT):-
   nl,
   print('this query finds those agent assignments for which'),
   nl,
   print('modification is defined'),
   nl,
   gensym(ag, AG),
   agent_constraint(AG, ASSGT),
   assert(holds(agent_constraint(AG, ASSGT), s_1)),
```

```
   holds(agent_constraint(AG, ASSGT), s_1),
   instantiate(P, ACT, s_1),
\+ (holds(agent_constraint(AG, ASSGT),  do(terminate(ACT),  s_1))).


new(P, ASSGT,ACT, AG, S_1):-
     instantiate(P, ACT, S_1),
\+ (holds(agent_constraint(AG, ASSGT), do(terminate(ACT),  S_1))).



instantiate(P, ACT, S):-
  has_isubactivity(P, ACT).



/*--------no_change_assignment/3 finds if no subactivity (ACT)
 of process (P) is defined to change an agent constraint (ASSGT)
-------------*/



no_change_assignment(P, ASSGT):-

                nl,
                print('this query finds those agent assignments for which'),
                nl,
                print('modification is not defined'),
                nl,
                gensym(ag, AG),
                agent_constraint(AG, ASSGT),
                assert(holds(agent_constraint(AG, ASSGT), s_1)),
                holds(agent_constraint(AG, ASSGT), s_1),
          \+ new(P, ASSGT, ACT, AG, s_1) ,
                retract(holds(agent_constraint(AG, ASSGT), s_1)).
```

B.3.6 stp.pl

```
%The agent/activity design strategies expertise.
% Writer: Katayoun Atefi.
% Date: May 1997.


:- multifile uses_information/2.
:- multifile produces_information/2.
:- multifile has_subactivity/2.
:- multifile has_isubactivity/2.

/* -------------------------
      LOADING LIBRARIES
-------------------------- */
:- ensure_loaded(library(lists)).
:- ensure_loaded(library(basics)).
:- ensure_loaded(library(prompt)).
:- ensure_loaded(library(ask)).
:- ensure_loaded(pmp_agents). /* agents file */
:- ensure_loaded(subactivity_def).
:- ensure_loaded(model).
:- unknown(_,fail).
/*
:- ensure_loaded(oosap).

*/

/* ------------------------------------------------
   This program makes a set of  the subactivities of process  A
    which are information related.

    Input =    A      (an instantiated process) .
    Output = LAC   (a set of informtion dependent activities).
-------------------------------------------------------*/
make_inf_dep_set(A, LAC):-
                  setof(A2,  A1^inf_rel(A,A1,A2), LAC).
%88888888888888888888888888888888888888888888888888888888

/*  ----------------------------------------
   This program finds if  two subactivities (A1 & A2)
    of process A, are information related; i.e.
    either A1 is infrmation dependent on A2
```

```
    or visa versa.
------------------------------------------*/
inf_rel(A,A1,A2):-
                  has_isubactivity(A, A1),
                  has_isubactivity(A, A2),
                  uses_information(A2, Inf),
                  produces_information(A1, Inf).


inf_rel(A,A1,A2):-
                  has_isubactivity(A, A1),
                  has_isubactivity(A, A2),
                  uses_information(A1,Inf),
                  produces_information(A2,Inf).


%8888888888888888888888888888888888888888


/* ----------------------------------------------------

   This program makes a set of  the possible values for agents
    that are specified (instantiated) individually by the user
    in the form of agent(ag).

     Output = LAG (a list)
-------------------------------------------------------*/
make_agent_set(LAG):-
                  findall(Ag, agent(Ag) , LAG).
%888888888888888888888888888888888888888888888888888888




/* ------------------------------------
 This prog. generates sublists  [X|Xs] from list Ys.
--------------------------------------  */

generate([X|Xs],Ys):-
                  member(X,Ys), generate(Xs,Ys).

generate([],Ys).
```

```
%88888888888888888888888888888888888888888888

/*-----------------------------------
  This is the program which is invoked by the user.

  This program generates a set of instantiated agents
   and tests if the set satisfies the constraint.
   A should be instantiated.
   The program either returns "no" which means
    no agent assignment can be found to satisfy the constraint
    (either no agents are stated in the possible_agents file,
     no activity produces the information used by the other
     activity)
    OR
    returns LAC and  G, meaning that there is a solution and that
    is G. The ith member of G is the performing agent of the ith
    member of LAC.

   It performs the follwing tasks:
    1) make_inf_dep_set/2 which establishes a set LAC from
        information dependent sub-activities of process A.

    2) make_agent_set/1 which puts all individual
       agents into a set. (this becomes the domain set of each agent).

    3) same_length/2 which makes a list G which
        has the same length as  LAC.

    4) generate which instantiates G from the domain; i.e. LAG.

    5) checks to see if the falsification of the constraint is false;
        i.e. the constraint is satisfied.

  Input     =  A      (a process)
  Output   =  LAC  (a set of subactivities which are information dependent,
  Output   =  G      (a set of instantiated agents which will be the solution)
----------------------------------------*/
agent_assignment(P, LAC, G):-
                    make_inf_dep_set(P, LAC),
                    make_agent_set(LAG),
                    same_length(G, LAC),
                    generate(G, LAG),
```

```
                \+ violate(LAC, G).



/*-------------------------------------------------
This program checks if the instantiated G violate the constraint,
  through the following steps:
                    1)  choosing three members of LAC
                  (i.e. the information related list).

               2) checking  if one of these members
                  produces the information used by
                  two others.

               3) finding the performing agents
                        of these activities from the list G.

                    4) checking if these agents violate the
                  constraint.

               If no, then back track to step (1) to see if other
               agents of G also vioalte the constraint.
               if no, then G is a solution because none of its
               members violated the constraint.
               If yes, then G is not a solution because one
              of its members violate the constraint, thus
               back track to program "generate" to
               instantiate another G.

-------------------------------------------------*/
violate(LAC, G):-
                  uses_information(XAC,Inf),
                member(XAC,LAC),
                  produces_information(YAC,Inf),
                  member(YAC,LAC),

             /* checking if one uses the information
                produced by the other*/

                  nth1(NX, LAC, XAC),
                  nth1(NY, LAC, YAC),
```

```
        /* finding the number of members of G
      who perform the activities:
      XAC, YAC.
 */

        nth1(NX, G, X),
        nth1(NY, G, Y),

 /* finding the members of G
      who perform the activities:
      XAC, YAC.
 */


        \+ ((X = Y)),


 /* checking if those members of G
      who perform the activities:
      XAC, & YAC, do not falsify the constraint.
 */

        /* the agent of the activity which produces
     the inf and the  agents of two activities
     which use it
     are not the same. */

        \+ ( any_team(X,Y)),
/* the agent of the activity  which produces
     the information does not have a team relationship
     with  the agents of two activities
     which use the information. */

        \+ (uses_information(ZAC,Inf),
      member(ZAC,LAC),
      XAC \==ZAC,
      nth1(NZ, LAC, ZAC),
      nth1(NZ, G, Z),
      X = Z).

 /* the agents of the different activities which use
      the same information are not the
      same.
```

```
        */


/*-------------------------------------
This program establishes a team relationship
between two agents (X and Y), no matter in the database
X is the one who is stated has a team relationship
with Y (i.e. X is the first argument)
or, Y is the one who is stated has a team relationship
with X (i.e. X is the second argument).
 -------------------------------------*/
any_team(X,Y):-
                team(X,Y);
                team(Y,X).
```

B.3.7 model.pl

```
%The PMP TOVE model.
% Writer: Katayoun Atefi.
% Date: May 1997.


:- multifile has_subactivity/2.

:- multifile holds/2.
:- dynamic holds/2.

:- multifile uses_information/2.
:- multifile produces_information/2.

activity(pmp(TRN)).



has_subactivity(pmp(TRN),  identify_potential_order(TRN)).
has_subactivity(pmp(TRN),  collect_and_evaluate_customer_data(TRN)).
has_subactivity(pmp(TRN),  select_and_assign_transaction_manager(TRN)).
has_subactivity(pmp(TRN),  evaluate_drop_select(TRN)).
has_subactivity(pmp(TRN),  drop(TRN)).

has_subactivity(collect_and_evaluate_customer_data(TRN), collect_customer_data(TRN)).
has_subactivity(collect_and_evaluate_customer_data(TRN), evaluate_customer_data(TRN)).
has_subactivity(collect_and_evaluate_customer_data(TRN), contact_product_manager(TRN)).
has_subactivity(collect_and_evaluate_customer_data(TRN), drop(TRN)).


has_subactivity(select_and_assign_transaction_manager(TRN),
identify_transaction_manager_required_skills(TRN)).
has_subactivity(select_and_assign_transaction_manager(TRN),
assign_skills_manager_for_available_transaction_manager_candidates(AG0, AG, TRN)).
has_subactivity(select_and_assign_transaction_manager(TRN), reassign_skills_manager(AG,
AG1, skills_manager, TRN)).
has_subactivity(select_and_assign_transaction_manager(TRN),
identify_transaction_manager_candidates(TRN)).
has_subactivity(select_and_assign_transaction_manager(TRN),
verify_availablity_of_transaction_manager_candidates(TRN)).
has_subactivity(select_and_assign_transaction_manager(TRN),
select_transaction_manager(TRN)).
```

```
has_subactivity(select_and_assign_transaction_manager(TRN),
assign_transaction_manager(AG, TRN)).

has_subactivity(select_and_assign_transaction_manager(TRN),
introduce_transaction_manager_to_customer(AG, CUST, TRN)).

has_subactivity(evaluate_drop_select(TRN), collect_strategical_data(TRN)).
has_subactivity(evaluate_drop_select(TRN), evaluate(TRN)).
has_subactivity(evaluate_drop_select(TRN), drop(TRN)).

%-------------------------------------------------
/* uses_information/2  specifies that first argument which is
   an activity uses the information, specified by the second argument.*/


uses_information(customer_call(TRN), contact(AG,TEL, TRN)).

uses_information(drop(TRN), not_acceptable_and_drop(TRN)).
uses_information(drop(TRN),  dropped(TRN)).


uses_information(analysis, reason_drop(TRN)).


uses_information(evaluate_customer_data(TRN), customer_business_needs(TRN)).
uses_information(evaluate_customer_data(TRN), customer_budget(TRN)).
uses_information(evaluate_customer_data(TRN), customer_exp_start_date(TRN)).
uses_information(evaluate_customer_data(TRN), customer_exp_end_date(TRN)).
uses_information(evaluate_customer_data(TRN), products_table).

uses_information(contact_product_manager(TRN), not_acceptable_but_proceed(TRN)).

uses_information(identify_transaction_manager_required_skills(TRN),
acceptable_and_proceed(TRN)).
uses_information(identify_transaction_manager_required_skills(TRN),
not_acceptable_but_proceed(TRN)).

uses_information(identify_transaction_manager_required_skills(TRN),
customer_business_needs(TRN)).
uses_information(identify_transaction_manager_candidates(TRN),
transaction_manager_required_skills(TRN)).
uses_information(identify_transaction_manager_candidates(TRN), employees_skills).
```

```
uses_information(verify_availablity_of_transaction_manager_candidates(TRN),
potential_transaction_managers(TRN)).
uses_information(verify_availablity_of_transaction_manager_candidates(TRN),
employees_availability).
uses_information(select_transaction_manager(TRN),
available_potential_transaction_managers(TRN)).
uses_information(assign_transaction_manager(AG, TRN), selected_transaction_manager(TRN)).



uses_information(collect_strategical_data(TRN), assigned_transaction_manager(AG,TRN)).
uses_information(collect_strategical_data(TRN), acceptable_and_proceed(TRN)).
uses_information(collect_strategical_data(TRN), not_acceptable_but_proceed(TRN)).

uses_information(evaluate(TRN), acceptable_and_proceed(TRN)).
uses_information(evaluate(TRN), not_acceptable_but_proceed(TRN)).
uses_information(evaluate(TRN), profit(TRN)).
uses_information(evaluate(TRN), producibility(TRN)).
uses_information(evaluate(TRN), allignment_with_goals(TRN)).

uses_information(order_management(TRN), selected(TRN)).
%-------------------------------------------------
/* produces_information/2  specifies that first argument which is
   an activity produces the information, specified by the second argument.*/
produces_information(drop(TRN), reason_drop(TRN)).



produces_information(collect_customer_data(TRN), customer_business_needs(TRN)).
produces_information(collect_customer_data(TRN), customer_budget(TRN)).
produces_information(collect_customer_data(TRN), customer_exp_start_date(TRN)).
produces_information(collect_customer_data(TRN), customer_exp_end_date(TRN)).



produces_information(evaluate_customer_data(TRN), acceptable_and_proceed(TRN)).
produces_information(evaluate_customer_data(TRN), not_acceptable_and_drop(TRN)).
produces_information(evaluate_customer_data(TRN), not_acceptable_but_proceed(TRN)).
```

produces_information(contact_product_manager(TRN), contact_manager_advice(TRN)).

produces_information(identify_transaction_manager_required_skills(TRN),
transaction_manager_required_skills(TRN)).
produces_information(identify_transaction_manager_candidates(TRN),
potential_transaction_managers(TRN)).
produces_information(verify_availablity_of_transaction_manager_candidates(TRN),
available_potential_transaction_managers(TRN)).
produces_information(select_transaction_manager(TRN), selected_transaction_manager(TRN)).
produces_information(assign_transaction_manager(AG,TRN),
assigned_transaction_manager(AG,TRN)).
produces_information(introduce_transaction_manager_to_customer(AG, CUST, TRN),
contact(AG, TEL, TRN)).

produces_information(collect_strategical_data(TRN), profit(TRN)).
produces_information(collect_strategical_data(TRN), producibility(TRN)).
produces_information(collect_strategical_data(TRN), allignment_with_goals(TRN)).

produces_information(evaluate(TRN), selected(TRN)).
produces_information(evaluate(TRN), dropped(TRN)).

B.3.8 main_def.pl

```
%Roles in PMP.
% Writer: Katayoun Atefi.
% Date: May 1997.


:- dynamic holds/2.
:- multifile holds/2.



/*-------------------contact pointS---------------*/
/* This group of axioms define as the effect of what activities an agent is
assigned as the  contact point for the customer.*/


%
/*From the start of a transaction, identifier is introduced as the contact point
   for the customer.*/
holds(agent_constraint(AG, contact(AG, TRN)), s_0):-
    holds(agent_constraint(AG, potential_order_identifier(AG, TRN)), s_0).



%
/*The identifier remains the contact point for the customer
until the transaction manger  is introduced to the customer. */


holds(agent_constraint(AG1, contact(AG1, TRN)), do(terminate(A), S)):-
  holds(agent_constraint(AG1, contact(AG1, TRN)), S),
  not(A = introduce_transaction_manager_to_customer(AG, CUST, TRN)).


%
/*When the transaction manager is introduced to the customer,  s/he
   becomes the contact point for the customer. */

holds(agent_constraint(AG, contact(AG,TRN)), do(terminate(A), S)):-
  A = introduce_transaction_manager_to_customer(AG, CUST, TRN).

/*From the time that the transaction  manager is introduced till the end of the process,
```

```
s/he  remains the contact point. */

holds(agent_constraint(AG, contact(AG, TRN)), do(terminate(A),  S)):-
   holds(agent_constraint(AG, transaction_manager(AG, TRN)), S),
   holds(agent_constraint(AG, contact(AG, TRN)), S).


/* This group of axioms defines how an agent is assigned to
    or is discharged from the  role of potenial order Identifier and-or transaction_manager.*/

%
/*----------------potenial_order _identifier----------------*/
/*Once an agent identifies a potenial order, s/he is and
      will be recognized as the potential order identifier.*/

holds(agent_constraint(AG, potential_order_identifier(AG, TRN)), do(terminate(A),  S)):-
  holds(agent_constraint(AG, potential_order_identifier(AG, TRN)), S).


%
/*----------------transaction_manager----------------*/
/*--An agent becomes an transaction  manager as the effect of the
activity  assign_transaction_manager ----*/

holds(agent_constraint(AG, transaction_manager(AG, TRN)), do(terminate(A), S)):-
  A = assign_transaction_manager(AG, TRN).



/*----The process does not consider any reassignment for the transaction_manager; i.e
        Once the transaction manager is assigned, the role  will remain with him/her.---*/

holds(agent_constraint(AG, transaction_manager(AG, TRN)), do(terminate(A), S)):-
  holds(agent_constraint(AG, transaction_manager(AG, TRN)), S).

/*---------This axiom defines the customer role --------------*/

holds(agent_constraint(AG, customer(CUST, TRN)), do(terminate(A), S)):-
  holds(agent_constraint(AG, customer(CUST,TRN)), S).

/*---This group of axiom defines as the effect of
```

```
        which activities  the customer  knows the contact point.
       ---------------*/

/* ----The customer  knows an agent is customer contact point when
              1) the customer information is collected, or
              2) the transaction manager is introduced to the customer.
     ----------*/


holds(knows(contact(AG, TRN), CUST), do(terminate(A), S)):-
  A = collect_customer_data(TRN).

holds(knows(contact(AG, TRN), CUST), do(terminate(A), S)) :-
  A = introduce_transaction_manager_to_customer(AG, CUST, TRN).


holds(knows(contact(AG, TRN), CUST), do(terminate(A), S)) :-
   holds(knows(contact(AG, TRN),CUST), S).


/*_____________________________________________________________________________*/

/*-----This group of axioms specifies as the effect of what
          activities an agent is assigned, discarded or reassigned
          to perform one of the activities of
          the Skills Manager.
-----------*/

%
/* ----An agent is assigned  to
           identify  transaction manager  candidates if s/he is
           a Skills Manager and s/he receives a request.
-----------*/

holds(agent_constraint(AG,
has_process_obligation(identify_transaction_manager_candidates(TRN),
           skills_manager, AG, TRN)), do(terminate(A),  S)):-
          A = (assign_skills_manager_for_available_transaction_manager_candidates(AG0, AG,
TRN)),
      \+ AG= AG0.

%
/* ----An agent  is assigned to identify  transaction manager  candidates
```

```
            if s/he a is skills manager and reassigned by another
             skills manager.
---------------*/
holds(agent_constraint(AG,
has_process_obligation(identify_transaction_manager_candidates(TRN),
         skills_manager, AG, TRN)), do(terminate(A), S)):-
        A = (reassign_skills_manager(AG0, AG, skills_manager , TRN)),
   \+ AG= AG0.


%
/* ----An agent  has the obligation to
          identify   transaction manager  candidates if the agent  is already assigned to do it and
          until the agent does not assign another  agent to do it.
----------------*/

holds(agent_constraint(AG,
has_process_obligation(identify_transaction_manager_candidates(TRN),
         skills_manager, AG, TRN)), do(terminate(A), S)):-
        holds(agent_constraint(AG,
has_process_obligation(identify_transaction_manager_candidates(TRN),
                 skills_manager, AG, TRN)), S),
  \+ (A = (reassign_skills_manager(AG, AG1, skills_manager, TRN))).


/*_____________________________________________________________________*/
/*-----This group of axiom specifies as the effect of which
          activity the assignment of an agent to an activity is recorded.

          In general, as soon as an activity is assigned to an agent, the
          assignment is recorded on the Opportunity Record.
--------------*/

holds(recorded(has_process_obligation(identify_transaction_manager_candidates(TRN),
skills_manager, AG, TRN), rec(TRN)), do(terminate(A),  S)):-
    A = (assign_skills_manager_for_available_transaction_manager_candidates(AG0, AG, TRN)).

holds(recorded(has_process_obligation(identify_transaction_manager_candidates(TRN),
skills_manager, AG, TRN), rec(TRN)), do(terminate(A), S)):-
    A = (reassign_skills_manager(AG0, AG,  skills_manager, TRN)).

holds(recorded(has_process_obligation(identify_transaction_manager_candidates(TRN),
```

```
skills_manager, AG, TRN), rec(TRN)), do(terminate(A),  S)):-
      holds(recorded(has_process_obligation(identify_transaction_manager_candidates(TRN),
       skills_manager, AG, TRN), rec(TRN)), S),
  \+ A = reassign_skills_manager(AG, AG1, skills_manager, TRN).

/*_____________WRITE & READ ACCESS _____________________*/
/*------This group of axioms spesify who has read and write access
    to the Opportunity Record. Owner and Identifier both have read access,
      Before that the transaction has a maanger, the identifier has the write
      access.
     Once the transaction has a manager , that manager has the write access.
 ----------------*/

 holds(has_access(read, AG, rec(TRN)), S):-
   holds(agent_constraint(AG, transaction_manager(AG, TRN)), S).

 holds(has_access(write, AG, rec(TRN)), S):-
   holds(agent_constraint(AG, transaction_manager(AG, TRN)), S).

 holds(has_access(read, AG, rec(TRN)),  S):-
   holds(agent_constraint(AG, potential_order_identifier(AG, TRN)), S).

 holds(has_access(write, AG, rec(TRN)), S):-
      holds(agent_constraint(AG, potential_order_identifier(AG, TRN)), S),
  \+ holds(agent_constraint(AG, transaction_manager(AG1, TRN)), S).



/*-------CAN KNOW----------------*/
/* ----can_know/2 states that an agent
           can know the information if the information
            is documented on a record
            and s/he has read access to the record.
-----------*/

holds(can_know(F,  AG), S)  :-
       holds(recorded(F, rec(TRN)), S),
       holds(has_access(read, AG, rec(TRN)), S).
```

B.3.9 scenario.pl

```
%A scenario of PMP.
% Writer: Katayoun Atefi.
% Date: May 1997.


:- multifile holds/2.
:- dynamic holds/2.
:- multifile occursT/2.


/*SCENARIO  for Case Manager*/

%-----------------------------------
holds(agent_constraint(iden, potential_order_identifier(iden, trn)), s_0).
holds(agent_constraint(cust, customer(cust, trn)), s_0).

occursT(terminate(identify_potential_order(trn)), 1).
occursT(terminate(collect_customer_data(trn)), 2).
occursT(terminate(evaluate_customer_data(trn)),  3).
occursT(terminate(identify_transaction_manager_required_skills(trn)),  4).

occursT(terminate(assign_skills_manager_for_available_transaction_manager_candidates(iden,
skm1, trn)), 5).

occursT(terminate(reassign_skills_manager(skm1, skm2, skills_manager, trn)), 7).

occursT(terminate(identify_transaction_manager_candidates(trn)), 8).
occursT(terminate(verify_availablity_of_transaction_manager_candidates(trn)), 9).
occursT(terminate(select_transaction_manager(trn)), 10).

occursT(terminate(assign_transaction_manager(trm, trn)), 12 ).
occursT(terminate(introduce_transaction_manager_to_customer(trm, cust, trn)), 13).

occursT(terminate( collect_strategical_data(trn)), 14).

occursT(terminate(evaluate(trn)), 18).
```

B.3.10 driver1.pl

```
%driver of the scenario, (this is a utility program for Temporal Progection).
% Writer: Mike Gruninger.



:- unknown(_,fail).
:- dynamic holds/2.
:- multifile holds/2, holdsT/2, occursT/2.

holds(n(F), S) :- not(holds(F, S)).

holdsT(F,T) :- during(T, S), holds(F, S).

actual(s_0).
actual(do(A, S)) :- occurs(A, S).

occurs(A, S) :-
                occursT(A,T),
                start(do(A, S),T).

start(s_0, 0).

start(do(A, S), T) :-
                occursT(A,T),
                (occursT(Ap,Ts) ; Ts = 0),
                Ts < T,
                not(occursBetTp(Ts,T)),
                start(S,Ts).

occursBetTp(Tp,T) :- occursBetT(E, Tp,T).

occursBetT(E,  Tp,T) :- occursT(E, Tpp), Tp < Tpp, Tpp < T.

during(T, S) :-
                start(S, T1),
                start(do(A, S), T2),
                T1 < T, T =< T2.
during(T,S) :-
                start(S,T1),
                T1 < T,
                not(actualAft(S)).
```

```
actualAft(S) :- actual(do(A, S)).

not(P) :- P, !, fail; true.
```

B.3.11 subactivity_def.pl

```
%Definition of Subactivity.
:- multifile has_isubactivity/2.

:- multifile has_subactivity/2.


/*Definition of subactivity*/


has_isubactivity(P,S):-
has_subactivity(P,S).

has_isubactivity(P,S):-
has_subactivity(P,SS),
has_isubactivity(SS,S).
```

B.3.12 pmp_agents

%The potential values for assigned agents
/* This file has all the possible values for agents.*/

/*The domain of possible values of the newly assigned
agents are the same as the domain of PMP current agents.
Currently each PMP transaction is processed by three different roles.
These roles are often performed by different agents.
To simplify the representation, we refer to these agents by their
organization roles, i.e.
potential order identifier, transaction manager and skills manager.

We did not assume a team relationship between
the potential order identifier, 'transaction manger,
and skills manager. */

agent(skills_manager).
agent(potential_order_identifier).

B.3.13 thesis_Queries.txt

%For Thesis demo:
==========
==========

Our FOL model for demo is loaded by the file:
/waterloo/atefi/3LAST/THESIS-CODE/thesis_ld_demo.pl

The file loads the following files:
:- ['model.pl'].

:- ['scenario.pl'].
:- [driver1].
:- [subactivity_def].
:- [library(strings)].
:- [library(sets)].
:- [library(lists)].
:- [library(basics)].
:- unknown(_,fail).

:- [main_def].
:- [dng].
:- [trc].
:- [chg].

==========

file: /waterloo/atefi/3LAST/THESIS-CODE/dng.pl

Query: dangling_information(ACT, INF).

** ACT is the activity which provides the information
but no activity uses it**
** INF is the information **

==========
==========
file: /waterloo/atefi/3LAST/THESIS-CODE/trc.pl

** incluing three queries **

Question: Is there an activity that at its start,
no agent is assigned as the contact point for the customer?

Query: no_contact(ACT).
** ACT is the one for which no contact exists**

Question: Is there an activity that at its start a contact
exists but the customer does not know the contact?

Query: contact_unknown(ACT).
** ACT is the one for which contact exists but not known by the customer **

==========

Question: Is there any activity that when it occurs an agent has been assigned to perform an activity
but the agent who is the contact point for the customer cannot know about it?

Query: contact_loses_agent_trace(ACT, AG, ROLE, TRN_CON, ACTD, S).

** ACT is the one to which an agent is assigned**
** AG is the assigned agent to Activity**
** ROLE is the role of the assigned agent for that activity**
** TRN_CON is the agent who is the contact point for the customer**
** S refers to the situation in which activity ACTD occurs but the customer
does not know about the assignment **

==========

file: /waterloo/atefi/ibm_demo/chg.pl

Question: Is there an agent constraint for which a change is defined?

Query: change_assignment(P, ASSGT, ACT).
change_assignment(pmp(trn), ASSGT, ACT).
** P is the aggregate activity or a process **
** ASSGT is the agent constraint such as role**
** ACT is the activity which modifies the constraint**

Question: Is there an agent constraint for which a change is not defined?

Query: no_change_assignment(P, ASSGT).
no_change_assignment(pmp(trn), ASSGT).
** P is the aggregate activity or a process **
** ASSGT is the agent constraint such as role**

==========

file: /waterloo/atefi/ibm_demo/stp.pl

Question: What are the agent assignments within process P that lead
to P's minimum agent setup time?

Query: agent_assignment(P, LAC, G).
agent_assignment(pmp(TRN), LAC, G).
agent_assignment(select_and_assign_transaction_manager(TRN), LAC, G).
** P is the aggregate activity or a process **
** LAC is the list of activities**
** G is the list of assigned agents**

References

Arend 93	Arend, M. (1993). Do You Really Need to “Reengineer”?, *ABA Banking Journal*, Dec. 1993, pp. 46-50.
Booth 94	Booth, R. (1994). Simple as ABC, What on Earth is Business Process Re-engineering?, *Management Accounting*, September 1994, p. 18.
Booth 95	Booth, R. (1995). In the Market, Manufacture Engineer, *Engineering Management*, October 1995.
Brierley 93	Brierley, E. (1993) Workflow Today and Tomorrow, *OIS Management; Proceedings of the Conference*, Editor: IIendnley, T. 1993, pp. 216-221.
Business Process Reengineering Tool Repository 96	Business Process Reengineering Advisory Group, Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto. (1996). Business Process Reengineering Tool Repository, *http://www.ie.utoronto.ca/EIL/tool/list.html.*
Clegg et al. 96	Clegg, BT., Buckingham, AD. (1996). Factory Process Improvement Using a Human centered Approach, Advances in Concurrent Engineering, CE 96, presented at the *Third ISPE International Conference on Concurrent Engineering: Research and Applications*, 1996, Toronto, Ontario, Canada, pp. 326-334
Davenport 93	Davenport, T.H. (1993). Process Innovation, Reengineering Work through Information Technology, Harvard Business School Press, Boston, Massachusetts.
Davenport 96	Davenport, T.H. (1996). How a Business Fad Went Wrong, *The Globe and Mail*, January 31, 1996.

Drew 94	Drew, S. (1994). BBP in Financial Services; Factors for Success, *Long Range Planning* Vol. 2, No. 5, 1994, pp. 25-41.
Earl 94	Earl, M. (1994). The New and Old of Business Process Redesign, *Journal of strategic Information Systems*, Vol. 3. No. 1, 1994, pp. 5-22.
Fitzgerald et al. 96	Fitzgerald, B., Murphy, C. (1996). Business Process Reengineering: Putting Theory into Practice, *Infor* Vol. 34, No. 1, 1996, pp. 3-14.
Fox 92	Fox, M.S. (1992). "The TOVE Project: A Common-sense Model of the Enterprise", Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, *Lecture Notes in Artificial Intelligence* # 604, Editors: Belli, F., Radermacher, F.J. Berlin: Springer-Verlag, 1992, pp. 25-34.
Fox 93	Fox, M.S. (1993). Issues in Enterprise Modelling, appeared in: *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 1993.
Fox et al. 93	Fox, M.S., Gruninger, M., Zhan, Y. (1993). Enterprise Engineering an Information Systems Perspective, Technical Report, Enterprise Engineering Laboratory, University of Toronto, 1993.
Gilmore 95	Gilmore, J. (1995). How to Make Reengineering Truly Effective?, *Planning Review*, May/June 1995.
Gonzalez & Dankel 93	Gonzalez, J., Dankel, D.D. (1993). *The Engineering of Knowledge Based Systems Theory and Practice*, 1993, Prentice Hall Inc.
Ghani 96	Ghani, U.A. (1996). Holistic Reengineering, *American Management Association*, Jan 1996, p 62.
Gruber 93	Gruber, T.R. (1993). Towards Principles for the Design of Ontologies Used for Knowledge Sharing, Technical Report, Knowledge System Laboratory, Stanford University, 1993.

Gruninger et al. 94	Gruninger, M., Fox, M.S. (1994). "An Activity Ontology for Enterprise Modelling", Submitted to: *Workshop on Enabling Technologies - Infrastructures for Collaborative Enterprises*, West Virginia University, 1994.
Gruninger et al. 95a	Gruninger, M., Fox, M.S. (1995). "Methodology for the Design and Evaluation of Ontologies", *Workshop on Basic Ontological Issues in Knowledge Sharing*, IJCAI-95, Montreal.
Gruninger et al. 95b	Gruninger, M., Fox, M.S. (1995). The Logic of Enterprise Modelling, Re-engineering the Enterprise, *Proceedings of the IFIP TC5/ WG5.7 Working Conference on Re-engineering the Enterprise*, Editors: Brown, J., O'sullivan, D. Galway, Ireland, 1995, pp. 83-98
Gruninger 95a	Gruninger, M. (1995). Characterization of Tools for Business Process Reengineering, [Manuscript], 1995, The paper would appear in Group documents of Business Process Reengineering Advisory Group at: http://www.ie.utoronto.ca/EIL/tool/frmwrk/stage/defmod/defmod.html.
Gruninger 95b	Gruninger, M. (1995). Designing Tools to Support Business Process Reengineering, Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto, *http://www.ie.utoronto.ca/EIL/grpdoc/bprtool.html*#HDR3.
Gruninger 95c	Gruninger, M. (1995). Org-theories, Enterprise Integration Laboratory, Department of Industrial Engineering, University of Toronto, http://www.ie.utoronto.ca/EIL/papers/model.html.
Guha 93	Guha, K. (1993). Aimed to Provide a Standard Framework for Large Scale Organizational Change. 1993, p. 16.
Hale 96	Hale, A.J., Cragg, P.B. (1996). Business Process Reengineering in the Small Firm, *Infor*, Vol. 34, No. 1, 1996, pp. 15-27.

Hammer 91	Hammer, M. (1990). Reengineering Work: Don't Automate, Obliterate, *Harvard Business Review*, July-August 1990, pp. 104-111.
Hammer et al. 93	Hammer, M., Champy, J. (1993). *Reengineering the Corporation: A manifesto for Business Revolution*, Harper Business, New York.
Harrington 91	Harrington, H.J. (1991). *Business Process Improvement*, McGraw-Hill, New York.
Hirshheim 86	Hirshheim, R.A. (1986). Perspectives and Views of the Office; Alternative Approaches to Understand the office, appeared in *Office Systems*, Editors: Verrijn-Stuart, A.S., Hirshheim, R.A. Elsevier Science Publishers B.V. (North Holland) IFIP, 1986.
Kelleher 95	Kelleher, D. (1995). Business Programmes and Information Systems Methodologies, *Info System*s J(1995)5, pp. 137-157
Khoong 96	Khoong, C.M. (1996). Culture-sensitive, Strategy-level Reengineering, *Infor* Vol. 34, No. 1.
Kim et al. 94	Kim, H., Fox, M.S. (1994). Formal Models of Quality and ISO9000 Compliance: An Information Systems Approach, American Quality Congress (AQC) Conference, American Society for Quality Control, Milwaukee WI: American Society for Quality Control, pp. 17-23.
Klein 95	Klein, M.M. (1995). Requirements for Successful Reengineering, *Infor* 33(4), 1995, pp. 225-233.
Klein 95	Klein, M.M. (1995). 10 Principles of Reengineering, Executive Excellence, Feb. 1995, p. 20.

Laakso et al. 95	Laakso, T., Hakamaki, J. (1995). Process Assessment and Simulation Games- Methods and Software Supported Tools in Business Process Reengineering, Re-engineering the Enterprise, Editors: Brown, J., O'sullivan, D. *Proceedings of the IFIP TC5/WG5.7 Working Conference on Re-engineering the Enterprise*, Galway, Ireland, 1995, pp. 302-311.
Ligus 93	Ligus, R.G. (1993). Methods to Help Reengineer Your Company for Improved Agility, *Industrial Engineering*, Jan. 93, Vol. 25, No. 1, pp. 58-59.
Luger & Stubblefield 93	Luger, G.F., Stubblefield, W.A. (1993). *Artificial Intelligence Structures and Strategies for Complex Problem Solving*, 1993, Second Edition, The Bejamin/Cummings Publishing Company Inc.
Miller 95	Miller, G. (1995). Reengineering: Forty u$eful hints, *Hospital Material Management Quarterly*, 1995, Vol. 17, No. 2, pp. 37-46.
Mintzeburg 79	Mintzeburg, M. (1979). *The Structuring of Organizations* (Englewood Cliffs, N.J.: Prentice-Hall.
Shoham 94	Shoham, Y. (1994). *Artificial Intelligence Techniques in Prolog*, 1994, Morgan Kaufmann Publishers, Inc.San Francisco, California, pp. 143-145.
Simons 95	Simons, M.L. (1995). Human side of Reengineering *Executive Excellence*, Feb. 1995, p. 19.
Slagle 71	Slagle, J.R. (1971). Artificial Intelligence: *The Heuristic Programming Approach*. New York, NY: McGraw-Hill.
Smith 1850	Smith, A. (1850). *The Wealth of Nations*, Edinburgh: Adam and Charles Black.
Spurr et al. 94	Spurr, K., Layzell, P. (1994). *Software Assisted for Business Reengineering*, Editors: Spurr, K., Layzell, P., Jennison, L., Richards, N. John Wiley & Sons, 1994.

Sterling et al. 86	Sterling, L., Shapiro, E. (1986). *The Art of Prolog*, MIT Press, Cambridge, MA 1986.
Strassman 93	Strassman, P. (1993). Rebottling Old Medicine: Origins and Relevance of Reengineering, *American Programmer*, Vol. 6, No. 11, pp. 3-9.
Tham et al. 94	Tham, D., Fox, M.S., Gruninger, M. (1994). A Cost Ontology for Enterprise Modelling *Third Workshop on Enabling Technologies-Infrastructures Collaborative Enterprises*, IEEE Computer Society Press, pp. 197-210.
Talwar 93	Talwar, R. (1993). Business Re-engineering-a Strategy-driven Approach, *Long Range Planning*, Vol. 26, No. 6, 1993, pp. 22 -40.
Venkatraman 94	Venkatraman, N. (1994). IT-Enabled Business Transformation: From Automation to Business Scope Redefinition, *Sloan Management Review*, winter 1994, pp. 73-87.
Vredde et al. 96	Vredde, G.J., Eijck, D.T.T., Sol, H.G. (1996). Dynamic Modelling for Re-engineering Organizations, *Infor* Vol. 34, No. 1, 1996, pp. 28-42.
Wagner et al. 94	S. Wagner, J. Liu, P. Jain (from Andersen Consulting), (1994). Discontinuous Transformations (DT); Transformational Approach to Business Process Redesign, appeared in *AAAI BPR Workshop*, 1994.
Weston et al. 95	Weston, R.H., Gilders, P.J. (1995). Enterprise Engineering Methods which Facilitate Simulation, Emulation, and Enactment via Formal Models, Modelling Methodologies for Enterprise Integration, *Proceedings of the IFIP TC5 working conference on models and methodologies for enterprise integration*, Editors: Bernus, P., Nemes, L. 1995, Queensland Australia, November 1995, pp. 218-233.

Weston 96	Weston, R.H. (1996). Model Driven Configuration of Manufacturing Systems in Support of the Dynamic, Virtual Enterprise, Advances in Concurrent Engineering, CE 96, presented at *The Third ISPE International Conference on Concurrent Engineering: Research and Applications*, Toronto, Ontario, Canada, pp. 425-438.
Yu 94	Yu, E.J. (1994). Modelling Strategies relationships for Process Reengineering, Thesis, 1994.
Yu et al. 93	Yu, E.S., Mylopoulos, J. (1993). An Actor Dependency Model of Organizational Work - With Application to Business Process Engineering, - *Proceeding Conf. on Organizational Computing Systems (COOCS 93)*, Nov. 1-4, 19993, Milpitas, California, USA, pp. 258-268.

Bibliography

Baiman 82	Baiman, S. Agency Research in Management Accounting, *A Survey in Modern Accounting Research: History, Survey and Guide*, Mattesich, R. 1983, pp. 251-292. Reprinted in *The Journal of accounting Literature*, 1 spring 1982, pp. 154-213.
Blackwell 53	Blackwell, D. (1953). Equivalent Comparisons of Experiments, *Annals of Mathematical Statistics*, 1953, pp. 267-272.
Brilouin 62	Brilouin, L. (1962). Science and Information Theory, 1962.
Druker 88	Drucker, P.F. (1988). The Coming Of The New Organization, *Harvard Business Review*, January-February 1988, pp. 45-53.
Druker 91	Drucker, P.F. (1991). The New Productivity Challenge, *Harvard Business Review*, November-December 1991, pp. 69-79.
Feltham 83	Feltham, G.A. (1983). Financial Accounting Research: Contributions of Information Economics and Agency Theory, in Modern Accounting Research: History, survey and Guide, Richard Mattesich, 1983, pp. 179-207.
Hyvarinen 68	Hyvarinen, L.P. (1968). Information Theory for Systems Engineers, 1968.
Malone et al. 93	Malone, T.W., Crowston, K. (1993). Working Paper #141 Center for Coordination Science Massachusetts, Institute of Technology May 199, In *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*, Morgantown, WV, April 1993.

Marschak and Radner 72	Marschak, J., Radner, R. (1972). Economic Theory of Teams, New Haven and London, Yale University Press, 1972.
Nicoletti 96	Nicoletti, S., Nicolo, F. (1996). A Concurrent Engineering Decision Model: Management of the Project Activities Information Flows, CE 96, presented at *the Third ISPE international conference on concurrent engineering: research and applications*, Toronto, Ontario, Canada, pp. 256-263.
Simon 81	Simon, H. (1981). *The Sciences of the Artificial*, 1981, Second edition, Cambridge MIT press.