# Matching Enterprise Requirements with ERP System Capabilities

By

**Pnina Soffer**

Supervised by

**Boaz Golany and Dov Dori**

Faculty of Industrial Engineering and Management
The Technion, Israel Institute of Technology
Haifa, 32000 Israel

## Abstract

The doctoral research deals with the main problem that characterizes ERP implementation projects: how to align off-the-shelf software with the business processes of the enterprise implementing it. The new solution approach, developed in this research, matches a model of a given ERP software package with a model representation of a set of enterprise requirements, which may have logical dependencies among them. The matching procedure involves a single requirement match, which uses an inaccurate *match in essence* search algorithm, and a bottom-up aggregation of the match results through the ERP system model. The aggregation provides a list of feasible configuration options of the ERP software, satisfying the given enterprise requirements, along with their matching scores. We use the Object Process Methodology (OPM) to model both the ERP system and the enterprise requirements, and utilize OPM's structure for the search and aggregation procedure that is developed here.

The research methodology includes developing the mechanism described above, and validating its effectiveness by applying it to a case study of a real ERP implementation project.

## 1. Introduction

Enterprise Resource Planning (ERP) systems have become widely used in the past few years and replaced legacy systems as the leading information systems in industrial enterprises.

ERP implementation raises a new problem that so far has hardly been dealt with in the literature: how to align off-the-shelf software with the business processes of the enterprise implementing it.

An ERP system is designed to serve a large variety of enterprises. As such, it has many options for supporting a variety of business processes used in different enterprises. The system configuration, defined by the values assigned to the system parameters during the implementation, determines the exact operation and the processes supported by the system in the specific enterprise.

Implementing an ERP system is often accompanied by a Business Process Reengineering (BPR), that changes the way the enterprise operates. Unlike traditional BPR, referred to as "fundamental rethinking" (Hammer and Champy, 1993), in our case the business processes have to be designed within the context of available ERP solutions, preferably without any extensions of the system's capabilities (Curran and Ladd, 1999).

Two separate domains co-exist in the initial stage of an ERP system implementation: one is the enterprise and the other is the ERP system. The enterprise has its existing business processes and a wish list of defined requirements. The ERP system has its available variety of supported business processes. The problem addressed in this research is one of comparing these two domains, analyzing the gap between them and defining the best solution to the enterprise needs within the system capabilities. At the end of this process the two separate domains are merged into one: the enterprise with its defined business processes, supported by the ERP system, with a properly defined configuration. In some cases this also includes an enhanced functionality developed as a customization of the package to specific requirements which are not satisfied by the available options.

This problem exists virtually in every ERP implementation project. Until now, it has been tackled intuitively without proper support tools. An adequate solution constitutes a crucial ingredient in the success of an ERP implementation project, since its output determines the future processes of the enterprise and the way the system will support them. Adapting standard business processes may influence the competitive position of the enterprise (Davenport, 1998); on the other hand, unnecessary software customizations may require resources that exceed the planned budget and may harm the system integrity, especially through future upgrades.

## 2. Related work

As noted, the problem addressed in this research has received relatively little attention in the literature. Gibson *et. al.* (1999) argue that it requires a departure from traditional system design methodologies. They present case studies, in which

three different aspects of the implementation process are studied. One of them, referred to as "business process design and software configuration", is the research problem. However, they only describe the outcome of the decisions made in the cases they studied and indicate that no formal tools were used in either case for this process.

Two opposite solution attitudes are discussed in the literature: double modeling and standard models.

The double modeling approach (Maimon, 1998) draws a detailed model of the enterprise (either reflecting the current state or a desired state after BPR). Then, it thoroughly matches the enterprise model with the ERP system model, in order to identify the matched parts and the gaps. This approach has two major problems: one is the demanding and costly work of building a detailed enterprise model (Curran and Ladd, 1999). This is often redundant, because the matching procedure is bound to change the processes' design. The second problem is that since the enterprise model is full and detailed, many of the gaps found by the matching procedure are of minor nature. For example, a demand might exist for a link between two entities. The system has a connection between these two entities that goes through a third entity that does not change the functionality of the link. The double modeling approach lacks the ability to identify crucial gaps while disregarding minor ones. Therefore, a lot of work is still required after the matching procedure is implemented in order to assess the validity of the many gaps that were identified. The advantage of this approach is its ability to capture unique requirements of the enterprise.

The second approach – standard models - uses a number of standard models supported by the system, and matches each such model to the enterprise based on its business characteristics. It does not apply an explicit representation of the enterprise requirements. There are two possible ways of matching. One is based on a rough characterization of a logistic typology or an industry type and refining the model is then done with the aid of enterprise modeling tools (Post and Van Es, 1996; Van Es, 1998; Curran and Ladd, 1999). The other way is a direct selection of a standard model, identified by a predefined detailed questionnaire. The standard model approach is common in commercial tools, and is also mentioned regarding enterprise modeling (Reithofer and Naeger, 1997). A very ambitious attempt in this direction is done by Scheer (1998), who presents the ARIS modeling framework and draws a comprehensive “Reference model” of business processes, which are common in industrial enterprises. The weakness of the direct selection is in the assumption that a predefined questionnaire can capture all the necessary information in order to match a model to the enterprise. In the cases where the model is roughly selected, the refining work is very exhausting and not properly supported. Daneva (1999) reports a reuse metrics associated with the SAP reference model. She defines these metrics in order to measure requirements reuse in SAP implementations, regarding the rich business reference model embedded in the software and four cases of enterprise requirements as given in the implementation blueprint. The results indicate that a full reuse was not achieved, although in some cases the rate of reuse was remarkably high. Therefore, we conclude that relying on a standard model is not sufficient, and an explicit representation of the enterprise domain is needed in order to capture the

full scope of requirements. The rationale in the standard model approach is that the enterprise should adapt itself to the system rather than the other way around. Davenport (1998) discusses the idea of a standard solution and states that it cannot be applied automatically without a thorough consideration. He stresses that it is crucial for an enterprise to identify its main competitive advantages and make sure that they are maintained through the implementation and BPR.

The strength of the standard solution approach is in the reuse of an existing model - gaining quality, saving time and avoiding the "reinvention of the wheel" phenomenon.

In practice, none of these two attitudes is applied in its pure form. The common practice is to focus on a known business domain within the ERP system and examine it with respect to a set of mandatory requirements. As noted, this is done intuitively and without proper support tools.

The mandatory requirements are obtained through a process of requirement elicitation and prioritization. In ERP implementation, unlike in software development projects, completeness of the requirements is not mandatory. Rather, these requirements should reflect the expected outcome of a BPR. The incompleteness of the requirements provides us with the flexibility to adapt an existing process of the ERP system and choose among several options that satisfy these requirements within the system.

The problem discussed here, though typical of ERP implementation projects, is not exclusive to this situation only. In fact, it appears in similar forms in every generic and configurable off-the-shelf software implementation project. In this broader context it is addressed by Cheong and Jarzabek (1999), who suggest a Customization Decision Tree (CDT) in order to identify the available customization options. However, a tree structure is insufficient for our purpose, which requires a more elaborate data and control structure.

## 3. Research Aim and Methodology

This research addresses the problem of process design and software configuration in ERP implementation. Its aim is to develop a tool and an underlying methodology that supports the practical approach of matching the ERP system to a set of enterprise mandatory requirements. The developed methodology entails an analytical matching procedure, using this set of requirements and the ERP system model (SM) as inputs.

The main issues addressed in the research:

1. Problem understanding and formulation.
2. Solution approach and methodology:
    - 2.1 Inputs structure – modeling framework, characterization and formalization of the SM and the enterprise requirements.
    - 2.2 Matching mechanism of a single requirement with the SM.
    - 2.3 Aggregation procedure providing an overall evaluation of the requirements' satisfaction within the system.

2.4 Outputs structure and analysis.

These four aspects are discussed in detail in the following section.

3. Validation and assessment of the methodology developed, using a real implementation project as a case study. The validation plan is: (a) to model the enterprise requirements on the basis of the documented requirements of a real project. (b) to construct the SM of the ERP system implemented in that project (the Baan ERP system). (c) to apply the methodology and compare the generated output with the actual implementation blueprint. The output comparison will be evaluated according to defined measures (for example, the number of gaps detected divided by the number of gaps identified in the project) It is expected to demonstrate the methodology's ability to support the solution process, which so far has been done manually. The case study is also expected to provide insights as to how to model and how to improve the methodology.

## 4. Solution Approach

### 4.1 Modeling framework of the SM and the enterprise requirements

The Object Process Methodology (OPM), described in detail in Dori (1995) and Dori and Goodman (1996) is used for modeling both the ERP system and the enterprise requirements.

OPM uses a single graphic model – a set of Object Process Diagrams (OPDs) to capture both the structural and the dynamic aspects of a system. There are two major differences between OPM and Object Oriented (OO) analysis methods. The first one is that while OO methods employ objects as major entities, with processes embedded as "methods", OPM treats objects and processes as equally important classes of entities, and this premise makes it suitable for enterprise modeling. The other difference is the number of different models the methodology requires. OO methods require the use of a set of models, each with its diagramming symbols and conventions, to describe different aspects of the system. The currently accepted UML standard, for example, uses no less than eight different models. OPM, in contrast, uses a single graphic tool, the Object-Process Diagram (OPD) set, as a single model of all the system aspects. This eliminates the proven model multiplicity problem of OO methods (Peleg and Dori, 2000), which requires a special effort to integrate the various views into a coherent specification of the system and to keep consistency among them.

For the purpose of detecting a match between a requirement model and the ERP system model (SM), the single view model has the advantage of capturing all the required information in a single type of diagram. Thus preventing the need for different search mechanisms or semantics for different diagram types, whose match results may need further integration.

While using a single model representation, OPM keeps simplicity through a scaling mechanism that controls the visibility of the system details. The scaling mechanism unfolds the details of the entities (objects or processes) in the top-level diagram through "descendant" diagrams, and thus constructs a network of Object-Process Diagrams.

The SM is, therefore, a hierarchically-structured OPD set, in which the scaling mechanism creates the “descendants” of the top-level OPDs. The descendant diagrams may represent alternative forms of the original diagram (denoted as "specialization" branches in OPD terminology and related to alternative options of the ERP system) or they may represent independent sub-entities of the original diagram that are revealed when that diagram is unfolded.

We may therefore represent each diagram as a node in a graph, with arcs connecting it to other nodes (either at its own level or to descendant diagrams), and explore the structure of this graph.

The notation used in this graph is as follows: each node is labeled by the entity specified by its diagram; each arc is a tuple <*o, t*>, where *o* is the origin node and *t* is the destination node. The top-level diagram is labeled as the *zero node*. Each diagram, being the detailed specification of an entity at a higher level diagram, has also an arc connecting it to its ancestor diagram. Since the origin entity may appear in several diagrams, there may be several arcs leading to each node. This makes the diagram graph under discussion a directed acyclic graph (which is not necessarily a tree).

An example of such a diagram graph structure is given in Figure 1. The illustrated part of an SM consists of 11 diagrams, each one is represented as a node in the graph. The arcs represent the hierarchical relations among the diagrams. For example: nodes 1, 2, 3, 4, 5 and 6 represent descendant diagrams of the top-level diagram, specifying its entities. The entity specified by the diagram in node 6 appears also in the diagrams 2 and 7.

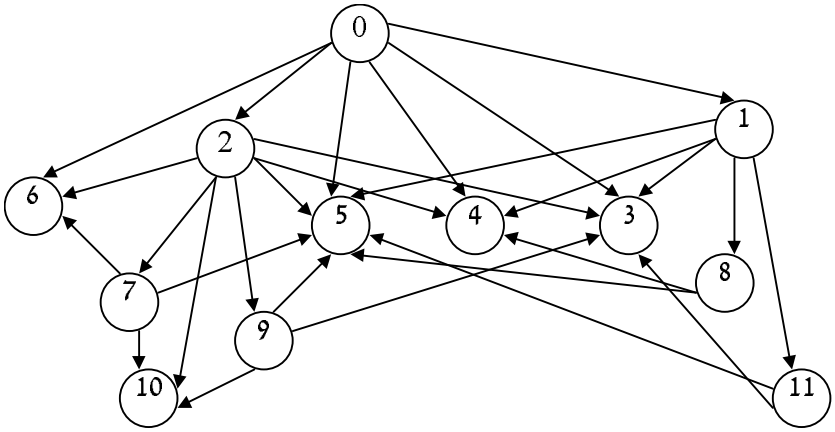


Figure 1: Diagram graph of an example SM part

The dependencies among the arcs can be represented as logical functions related to the various nodes, whose operators are determined according to the content of the diagram. For example, the function “A AND B AND [(C AND D) XOR (E AND F)]” is related to a diagram consisting of the entities “A”, “B”, “C”, “D”, “E” and “F”. According to the function, the entities “A” and “B” are linked to two alternative options, represented by entities “C” and “E”, and each of the

alternative entities has another entity related to itself only: “D” to “C” and “F” to “E”.

As to the enterprise requirements, each of them can be represented as a single OPD. However, in order to capture the full scope of the enterprise needs, and to verify their satisfaction in a specific ERP configuration, we need the ability to express the dependencies that exist among the requirements. Some of the requirements may be alternatives, and some are mandatory only in conjuncture with others.

We therefore represent the full set of enterprise requirements (noted as ER) as a logical combination of all the single requirements.

### 4.2 Matching mechanism of a single requirement with the SM

The research problem combines two sub-problems: (1) matching a single requirement, and (2) matching a combination of requirements. The first sub-problem to be discussed is the single requirement matching:

Given a model representation of a single enterprise requirement as an OPD and a model representation of the ERP system (the SM) as an OPD set, we regard each OPD as a labeled graph, and ask whether the requirement is met by the system. The simplest case is when the requirement is found as a subgraph of the SM. In this case, the requirement is definitely met by the system; but although sufficient, this is not a necessary condition.

As noted, the mandatory requirements are partial rather than a complete and detailed enterprise model. Therefore, the SM may be much more detailed than a requirement model and a more relaxed matching method should be developed. For example, a requirement may specify a link between a certain object and a certain process, while in the ERP system the link is implemented via other processes or objects. In this case, although the requirement model is not a subgraph of the SM, the requirement is still met. We call this a *match in essence*. To define this concept, we employ the following definitions.

*Definition 1:* A *path* between entities A and B is a sequence of OPM links and entities, connecting the origin A to the destination B.

*Definition 2:* A path between A and B is *equivalent to a link* of a given type C between A and B if:

(1) the path sequence contains this type of link only,

or

(2) the sequence A - path - B can be replaced by a sequence A’ - link C - B’, where A’ and B’ are entities of the same type as A and B, containing A and B respectively.

*Definition 3:* Diagram A *matches* diagram B *in essence* if and only if:

(1) B is a subgraph of A,

or

(2) All the entities of B are also included in A and for each link in B there is an equivalent path in A.

The match in essence is illustrated in Figure 2. The Object-Process diagram(OPD) in Figure 2 (b) matches the OPD in Figure 2 (a) in essence, since objects B, C and D comprise a B' with an equivalent path to the effect link in Figure 2 (a). In the OPD in Figure 2 (c), the processes A and C make an A' with an effect link with the object B; therefore this OPD matches the Figure 2 (a) OPD in essence as well.

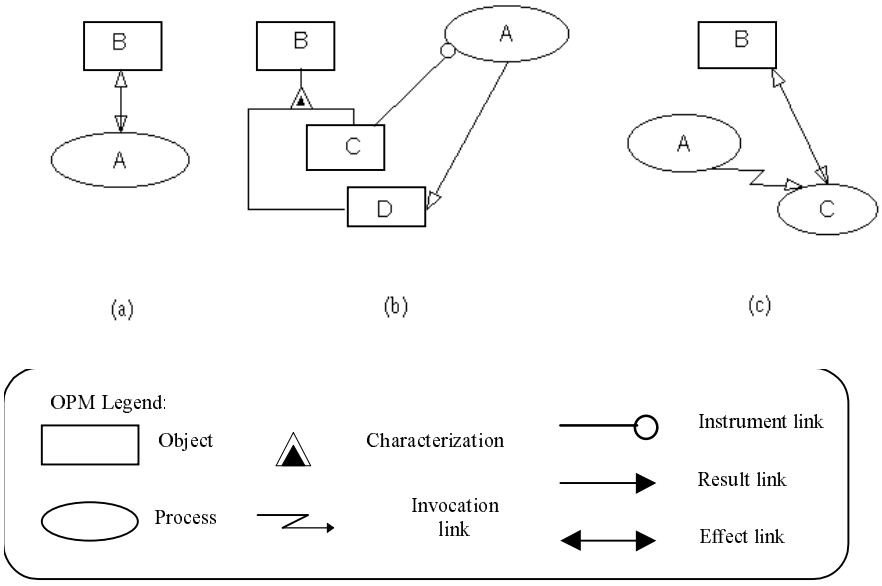


Figure 2: Example of a match in essence

Some similar problems are discussed in the literature. For example, Bunke (1998) discusses the problem of error-tolerant graph matching and presents some algorithms for this problem. The graph-theoretic disjoint paths problem, discussed and solved by Robertson and Seymour (1995), is of deciding whether k pairs of nodes in a graph have k mutually disjoint paths joining them. These results, however, relate to general labeled graphs and do not take into account the semantic difference of the various links in an OPD. Therefore, they are not helpful in identifying a match in essence. OPM has already been used as a basis for image retrieval (Dori and Hel-Or, 1998), but the matching that is sought there is basically a full match, not a match in essence.

In practice, partial matching means that some customization may be needed. That is a rather common outcome of a matching evaluation. Therefore, instead of requiring a Boolean answer, we are seeking an assessment of the match level as a similarity measure or a matching score.

The mechanism developed for the Single Requirement Match (SRM) is a search procedure that is executed for each single requirement separately. The inputs for this procedure are (1) a single requirement expressed as an OPD, (2) the SM, and

(3) a matching lower bound (LB) given by the user. The purpose of the LB is to define a threshold that limits the matching score, so that the search provides only the diagrams whose matching score exceeds it.

The search procedure has two steps: (1) Component matching - this step is a "where-used" query, identifying the diagrams in the SM whose components are identical to those in the requirement. It provides a list of the diagrams in which the portion of the requirement components is higher than the matching LB.

(2) Essential matching - this step computes the matching score for each diagram that has been identified in the previous step. The essential matching examines each link between pairs of entities in the requirement model and uses an algorithm that considers the various link types of the OPM, to search and identify an equivalent path in the investigated diagram. The matching algorithm has a polynomial run time in the size of the problem. The matching score is then computed, based on the total match in essence found. The output of the SRM is the requirement results: a set of diagrams with the matching score associated to each diagram.

### 4.3 The aggregation procedure

Even when each requirement is matched separately, this does not necessarily imply a full match between the ERP system and the requirements. Different requirements may be met by different options of the system, and there is not necessarily a single configuration in which all the requirements are satisfied at the same time.

Therefore, after applying the SRM for each single requirement, the single requirement results are aggregated up the SM. The Bottom-up Aggregation (BUA) process combines the SRM results for all the requirements into an overall evaluation of the requirements' satisfaction within the system.

The BUA algorithm relies on two main attributes of each node in the SM diagram graph. One is the logical function relating the entities of the diagram in each node $d$ (noted as $F_d[V]$) and the other is a logical expression combining all the requirements results aggregated up to this node (noted as $C(d)$).

The algorithm begins when the matching results (SRM output) are given, that is, there is a set of nodes in the graph with a known $C(d)$. It arranges the nodes of that set according to their distance from the zero node, and starts with those whose distance is maximal. At each step it goes one level up and aggregates the results for the next level.

While going up the set of arcs $<o,t>$ from all the descendant diagrams $t$ up to their mutual origin diagram $o$, $C(o)$ is computed by collecting all the relevant $C(t)$ and assigning them as a vector in $F_o[V]$.

This procedure is repeated until the top-level diagram (the zero node) is reached. Its combined result $C(0)$ holds all the requirements results aggregated up the SM and their logical relations, thus providing the matching options of the ERP system regarding the enterprise requirements.

The complexity of the BUA algorithm is polynomial in the size of the SM and the ER.

### 4.4 The matching options analysis

The matching options, given as $C(0)$, are the aggregated logical combination of requirements and their matching scores. They form the solution space, defined as the system options that satisfy the enterprise requirements together with their matching scores, and should be equivalent to the enterprise requirements (ER) combination. The analysis of the matching options consists of the following steps:

(1) Find out whether the ERP system matches the requirements by checking if C(*0*) gives TRUE for ER, regardless of the scores.

(2) Identify the feasible options for the enterprise within the ERP system – an option is feasible if after eliminating its alternatives from C(*0*) it still gives TRUE for ER.

(3) Evaluate the alternative feasible options by computing a total matching score to each feasible option.

(4) If the requirements are not satisfied, or if partial match is not sufficient for the enterprise, then software customizations may be required. The customizations can be designed based on the closest matching option and extending it.

In the context of the ERP implementation decision making, the total score refers to the requirement matching only. The decision of which option to select may be based not only on the matching score, but on many other criteria, such as user friendliness, sensitivity to mistakes, expected operational cost and others.

## 5. The Research Contribution

The problem addressed by this research is at the heart of ERP systems implementation and its adequate solution is crucial for the implementation success and for the future of the implementing enterprise.

From a theoretical point of view, this problem lies in between the disciplines of Information Systems Analysis and Operations Management, and does not belong entirely to either one.

The existing support tools are far from being satisfactory, and are based on one of two known approaches. One is the double modeling approach, which involves a redundant enterprise modeling and identifies many non-essential gaps between the ERP system and the enterprise needs. The second is the standard model approach, that either ignores non-standard requirements, or requires spending an exhaustive effort (relying mostly on intuition) to meet them. The approach developed in this research is innovative in the support it provides for matching the ERP system with a set of mandatory requirements.

The procedure developed, although designed for ERP systems, may be used for configuring a variety of enterprise-wide off-the-shelf software packages. It may also be used for retrieval of reusable software components from a repository.

Two other conceptual contributions of this research: (1) the use of OPM for ERP modeling and the SM structure definition as a diagram graph; (2) the concept of a *match in essence*, and the ability to detect its existence in OPDs.

Further research may extend the ERP implementation support tool and combine the advantages of the procedure developed in this research with the advantages of a predefined questionnaire, characterizing common features of an enterprise. The features identified by the questionnaire can be examined through a one time off-line search and located in the SM as parts of the overall result C(*0*), thereby eliminating the on-line search for obvious requirements covered by the questionnaire, and enable focusing on unique requirements of the enterprise only.

## References


Bunke, H., 1998, Error-Tolerant Graph Matching: A Formal Framework and Algorithms, A*dvances in Pattern Recognition, Joint IAPR International Workshop*, p. 1-14

Cheong, Y. C. and Jarzabek, S., 1999, Frame-based Method for Generic Software Architectures, *SSR'99, Proceedings of the Fifth Symposium on Software Reusability,* p. 103-112

Curran, T. A and Ladd, A., 1999, *SAP R/3 Business Blueprint: Understanding Enterprise Supply Chain Management*, 2nd ed., Prentice Hall

Daneva, M, 1999, Measuring Reuse in SAP Requirements: a Model-based Approach, *SSR'99, Proceedings of the Fifth Symposium on Software Reusability,* p. 141-150

Davenport, T. H., 1998, Putting the Enterprise into the Enterprise system, *Harvard Business Review*, p. 121-131

Dori, D., 1995, Object Process Analysis: Maintaining the Balance Between System Structure and Behavior, *Journal of Logic Computation*, p. 227-249

Dori, D. and Goodman, M., 1996, From Object Process Analysis to Object Process Design, *Annals of Software Engineering*, 2, p. 20-25

Dori, D. and Hel-Or, H., 1998, Semantic Content Based Image Retrieval Using Object-Process Diagrams, A*dvances in Pattern Recognition, Joint IAPR International Workshop*, p. 15-29

Gibson, N., Holland, C.P., Light, B., 1999, Enterprise Resource Planning: A Business Approach to Systems Development, *Proceedings of the 32nd Hawaii International Conference on System Sciences*

Hammer, M. and Champy, J., 1993, *Reengineering the Corporation*, New York, Harper Collins

Maimon, O., 1998, The next Generation of ERP Systems, *IE&M '98, 10th Industrial Engineering and Management Conference*, p. 30-34

Peleg, M. and Dori, D., 2000, The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods. *IEEE Transactions on Software Engineering* (to appear)

Post, H. A. and Van Es, R. (Eds), 1996, *Dynamic Enterprise Modeling: A Paradigm Shift in Software Implementation*, Kluwer, Dordrecht

Reithofer, W. and Neager, G., 1997, Bottom-up Planning Approaches in Enterprise Modeling - the Need and the State of the Art, *Computers in Industry*, 33, p. 223-235

Robertson, N. and Seymour, P. D., 1995, Graph Minors XIII. The Disjoint Paths Problem, *Journal of Combinatorial Theory, Series B*, 63, p. 65-110

Scheer, A.W., 1998, *Business Process Engineering, Reference Models for Industrial Enterprises*, Springer, Berlin

Van Es, R., 1998, *Dynamic Enterprise Innovation*, Baan Business Innovation B.V., The Netherlands.