INFORMATION INTEGRATION FOR CONCURRENT ENGINEERING (IICE)
TOWARD A METHOD FOR BUSINESS CONSTRAINT DISCOVERY (IDEF9)

Richard J. Mayer, PhD

Michael K. Painter

Madhavi Lingineni

Knowledge Based Systems, Inc.
One KBSI Place
1500 University Drive East
College Station, Texas 77840-2335

HUMAN RESOURCES DIRECTORATE
LOGISTICS RESEARCH DIVISION

2698 G Street

Wright-Patterson Air Force Base, Ohio 45433-7604

APRIL 1995

Interim Technical Paper for Period February 1991 to March 1995

AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-7604

INFORMATION INTEGRATION FOR CONCURRENT ENGINEERING (IICE)
TOWARD A METHOD FOR BUSINESS CONSTRAINT DISCOVERY (IDEF9)

Version 1.0

Richard J. Mayer, PhD
Michael K. Painter
Madhavi Lingineni

Knowledge Based Systems, Inc.
One KBSI Place
1500 University Drive East
College Station TX 77840-2335
(409) 260-5274

APRIL 1995

Contract No.: F33615-90-C-0012

Prepared for:
Armstrong Laboratory
Logistics Research Division
Wright-Patterson Air Force Base, Ohio 45433-7604
(513) 255-7775

Report Documentation Page

Abstract (Maximum 200 words)

Policies, rules, conventions, procedures, contracts, agreements, regulations, societal and physical laws are the defining structure for an enterprise. These items forge relationships between people, information, material, and machines to make a system. In this report, we refer collectively to these items as constraints. Constraints initiate, enable, govern, and limit the behavior of objects and agents to accomplish the goals or purposes of a system. If we want to change the behavior of a system for whatever reason (e.g., improve its performance, efficiency, or effectiveness) we need to know what the relevant constraints are. However, the collection of constraints that forge an enterprise system is generally poorly defined. That is to say, the knowledge of what constraints exists and how those constraints interact is at best incomplete, disjoint, distributed, and often completely unknown. The IDEF9 Business Constraint Discovery method described in this report was designed to assist in the discovery and analysis of constraints in a business system. Once these constraints have been cataloged they can be systematically examined and, if necessary, tuned or replaced to improve the performance of the system.

Subject terms: Integration, Integration Definition, IDEF, method, methodology, modeling, knowledge engineering, knowledge acquisition, constraint, systems engineering

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This document provides a summary of research toward development of an IDEF9 Business Constraint Discovery method performed under the Information Integration for Concurrent Engineering (IICE) project, contract # F33615-90-C-0012, funded by Armstrong Laboratory, Logistics Research Division, Wright-Patterson Air Force Base, Ohio, under the technical direction of United States Air Force Captain JoAnn Sartor and Mr. James McManus. The prime contractor for IICE is Knowledge Based Systems, Inc. (KBSI), College Station, Texas. Dr. Paula S. deWitte was the IICE Project Manager at KBSI. Dr. Richard J. Mayer was the Principal Investigator on this project. Mr Thomas Blinn was the IICE Technical Manager and also served as the Project Manager during the final close out of this effort. Michael K. Painter is the Methods Engineering thrust manager. The authors gratefully acknowledge the technical support of the IDEF9 Business Constraint Discovery Method Development Team whose names are listed below.

Perakath Benjamin, Ph.D.

Bruce E. Caraway

John W. Crump, IV

Florence Fillion

Mike Graul, Ph.D.

Umesh Hari

Arthur Keen, Ph.D.

Madhavi Lingineni

Richard J. Mayer, Ph.D.

Christopher P. Menzel, Ph.D.

Michael K. Painter

FOREWORD

The Department of Defense (DoD) has long recognized the opportunity for significant technological, economic, and strategic benefits attainable through the effective capture, control, and management of information and knowledge resources. Like manpower, materials, and machines, information and knowledge assets are recognized as vital resources that can be leveraged to achieve a competitive advantage. The Air Force Information Integration for Concurrent Engineering (IICE) program, sponsored by the Armstrong Laboratory's Logistic Research Division, was established as part of a commitment to further the development of technologies that will enable full utilization of these resources.

The IICE program was chartered with developing the theoretical foundations, methods, and tools to successfully evolve toward an information-integrated enterprise. These technologies are designed to leverage information and knowledge resources as the *key* enablers for high quality systems that achieve better performance in terms of both life cycle cost and efficiency. The subject of this report, the IDEF9 Business Constraint Discovery method, is one of a family of methods that collectively constitute a technology for leveraging available information and knowledge assets. The name IDEF originates from the Air Force program for Integrated Computer-Aided Manufacturing (ICAM) from which the first ICAM Definition, or IDEF, methods emerged. It was in recognition of this foundational work, and in support of an overall strategy to provide a family of mutually-supportive methods for enterprise integration, that continued development of IDEF technology was undertaken. More recently, with the expanded focus and use of IDEF methods as part of Concurrent Engineering, Total Quality Management (TQM), and business re-engineering initiatives, the IDEF acronym has been re-cast, referring now to an integrated family of Integration Definition methods. Before discussing the development strategy for providing an integrated family of IDEF methods, however, the following paragraphs will briefly describe what constitutes a method.

Method Anatomy

A method is an organized, single-purpose discipline or practice (Coleman, 1989). A method may have a formal theoretic foundation, although this is not a requirement. Generally, methods evolve as a distillation of the *best-practice* experience in a particular domain of cognitive or physical activity. The term methodology has at least two common uses. The first use is in reference to a class of similar methods. According to this use, one may, for example, hear reference to the *function modeling methodology* when discussing methods such as IDEFØ[1] and LDFD.[2] In the second common use, *methodology* is used to refer to *a collection of methods and tools, the use of which is governed by a process superimposed on the whole* (Coleman, 1989). Thus, it is common to hear the criticism that a tool (or method) has no underlying methodology when the tool (or method) has a graphical language but no underlying procedure for the appropriate application of the language or use

[1] ICAM Definition method for Function Modeling.

[2] Logical Data Flow Diagramming method.

of the resulting models. *Tool* is used to refer to a software system designed to support the application of a method.

Although a method may be informally thought of as a procedure for performing a task and, perhaps, a representational notation, it can be more formally described as consisting of three components as illustrated in Figure 1. Each method has (1) a definition, (2) a discipline, and (3) many uses. The definition specifies the basic intuitions and motivation behind the method, the concepts involved, and the theory of its operation. The discipline includes the procedure by which the method is applied and the method's language, or syntax. The procedure associated with the method discipline provides the practitioner with a reliable process for consistently achieving good results. The method syntax eliminates ambiguity among those involved in the development of complex engineering products. Many system analysis and engineering methods use a graphical syntax to provide visualization of collected data in such a way that key information can be easily extracted.[3] The third element of the method anatomy, the *use* component, focuses on the context-specific application of the method.

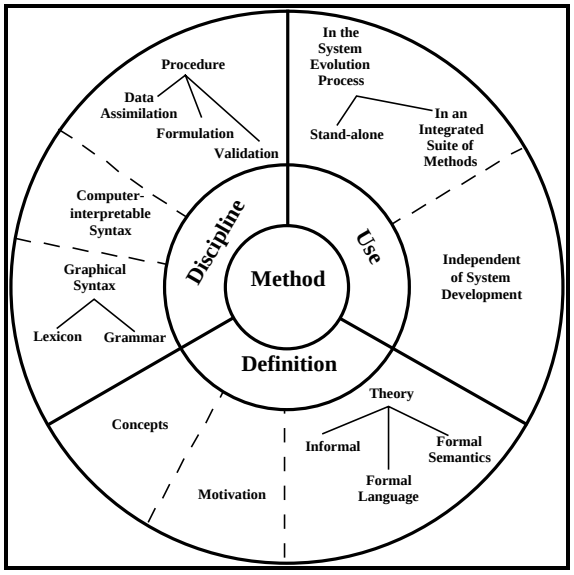


Figure 1
Anatomy of a Method

Ultimately, methods are designed to facilitate a scientific approach to problem solving. This goal is accomplished by first helping one understand the important objects, relations, and constraints that must be discovered, considered, or decided on. Second, scientific problem solving occurs by guiding the method practitioner through a disciplined approach that is consistent with good-practice experience and leads toward the desired result.

[3] Graphical facilities provided by a method language serve not only to document the analysis or design process undertaken, but more importantly, to highlight important decisions or relationships that must be considered during method application. The uniformities to which an expert, through experience, has become attuned are thus formally encoded in visualizations that emulate expert sensitivities.

Formal methods, then, are specifically designed to raise the performance level (quality and productivity) of the novice practitioner to a level comparable with that of an expert (Mayer, 1987).

Family of Methods

As Mr. John Zachman, in his seminal work on information systems architecture, observed:

> [T]here is not an architecture, but a set of architectural representations. One is not right and another wrong. The architectures are different. They are additive, complementary. There are reasons for electing to expend the resources for developing each architectural representation. And, there are risks associated with not developing any one of the architectural representations.

The consistent, reliable creation of correct architectural representations, whether they be artificial approximations of a system (models) or purely descriptive representations, requires the use of a guiding method. These observations underscore the need for many "architectural representations," and, correspondingly, many methods.

Methods, and their associated architectural representations, focus on a limited set of system characteristics and explicitly ignore those that are not directly pertinent to the task at hand. Methods were never intended to evaluate and represent every possible state or behavioral characteristic of the system under study. If such a goal were achievable, the exercise would itself constitute building the actual system, thus negating the benefits to be gained through method application (e.g., problem simplification, low cost, rapid evaluation of anticipated performance, etc.).

The search for a single method, or modeling language, to represent all relevant system life cycle and behavioral characteristics, therefore, would necessitate skipping the design process altogether. Similarly, the search for a single method to facilitate conceptualization, system analysis, and design continues to frustrate those making the attempt.

The plethora of special-purpose methods which typically provide few, if any, explicit mechanisms for integration with other methods is equally frustrating. The IDEF family of methods is intended to strike a favorable balance between special-purpose methods whose effective application is limited to specific problem types, and "super methods" which attempt to include all that could ever be needed. This balance is maintained within the IDEF family of methods by providing explicit mechanisms for integrating the results of individual method applications.

Critical method needs identified through previous studies and research and development activities[4] have given rise to renewed effort in IDEF method integration and

[4]Of particular note is the Knowledge-Based Integrated Information Systems Engineering (KBIISE) Project conducted at the Massachusetts Institute of Technology (MIT) in 1987 where a collection of highly qualified

development activities, with an explicit mandate for compatibility among the family of IDEF methods. Providing for known method needs with a family of IDEF methods was not, however, the principal goal of the methods engineering activity within the IICE program. The primary emphasis for these efforts was directed toward establishing the foundations for an engineering discipline guiding the appropriate selection, use, extension, and creation of methods that support integrated systems development in a cost-effective and reliable manner.

New methods development has struck out where known and obvious method voids existed (rather than reinventing existing methods) with the explicit mission to forge integration links among existing IDEF methods. When applied in a stand-alone fashion, IDEF methods serve to embody knowledge of good practice for the targeted fact collection, analysis, design, or fabrication activity. As with any good method, the IDEF methods are designed to raise the performance level of novice practitioners by focusing attention on important decisions while masking out irrelevant information and unneeded complexity. Viewed collectively as a toolbox of complementary methods technology, the IDEF family of methods is designed to promote integration of effort in an environment in which global competitiveness has become increasingly dependent upon the effective capture, management, and use of enterprise information and knowledge assets.

---

experts from academic and research organizations, government agencies, computer companies, and other corporations identified method and tool needs for large-scale, heterogeneous, distributed systems integration. See Defense Technical Information Center (DTIC) reports A195851 and A195857.

EXECUTIVE SUMMARY

Policies, rules, conventions, procedures, contracts, agreements, regulations, societal and physical laws are the defining structure for an enterprise. These items are the mechanisms for forging relationships between people, information, material, and machines to make a system. In this report, we refer collectively to these items as constraints. If you view an enterprise as a machine, constraints form the architecture and the programming language that define the behavior of that machine. If you view the enterprise as an organism, they form the control structure of that organism, from the genetic code level through the autonomous stimulus response level, to the cognitive behavior level.

The IDEF9 Business Constraint Discovery method described in this report was designed to assist in the discovery and analysis of constraints in a business system. A primary motivation driving the development of IDEF9 was an acknowledgement that the collection of constraints that forge an enterprise system is generally poorly defined. That is to say, the knowledge of what constraints exists and how those constraints interact is at best incomplete, disjoint, distributed, and often completely unknown. This situation is not necessarily alarming, just as a human as a living organism need not be aware of the genetic or autonomous constraints that govern certain behaviors, an organization can (and most do) perform well without explicit knowledge of the glue that structures the system. However, if the desire exists to modify the business to improve its performance or adapt to market, product, or process changes in a predictable manner, then knowledge of these constraints is as critical as knowledge of genetics is to the genetic engineer.

Constraints are the mechanisms by which humans and nature form systems. Constraints initiate, enable, govern, and limit the behavior of objects and agents to accomplish the goals or purposes of a system. If we want to change the behavior of a system for whatever reason (e.g., improve its performance, efficiency, or effectiveness) we need to know what the relevant constraints are. The payoff of the IDEF9 technology to an organization is that it facilitates the discovery and mapping of the relevant constraints in an organizational system. Once these constraints have been cataloged they can be systematically examined and, if necessary, tuned or replaced to improve the performance of the system. Constraints often serve a dual role as both the glue and the rationale for a system. That is, the collection of relevant constraints often constitutes the description of why the system behaves as it does. From this perspective, IDEF9 provides a reverse engineering tool for the business engineer. It can assist him in discovery of the “logic” behind the design of an existing system. It also provides a mechanism for specifying the logic of a “To-Be” system.

INTRODUCTION

It is easy to think of instances of policies, rules, laws, or methods etc., that govern the behavior of parts of an enterprise. But in order to define a method for discovery and analysis of such phenomena we need to step back and form a perspective for understanding the nature of how these and other constraint forming mechanisms work.

What is a constraint?

A *constraint* is a relationship that is maintained or enforced in a given context. The term *relationship* refers to an abstract, general association or connection that holds between two or more conceptual or physical objects. A constraint is simply a special kind of relationship—one that is checked, restricted, or compelled to exist under a given set of conditions. The term *context* refers to such a distinguished set of conditions. A constraint is thus said to *hold* in a given context when a relationship is maintained or enforced in that context. Conversely, a constraint does not hold in a given context if for that set of conditions the constraint is not maintained or enforced.

Examples of constraints are found all around us. There are constraints between objects, between objects and processes, between processes, and between objects, object properties, and the value of those properties. Constraints are expressed in *constraint statements*. For example, a constraint statement expressing a constraint between objects might be "Only project managers are authorized to sign pre-trip requests." An example of a constraint between objects and processes might be something like, "All travel requires an approved pre-trip request." "Drilling precedes reaming" or "Leasing requires making monthly payments" reflect constraints between processes. "The maximum occupancy of the building complex is two hundred people" or "Product A requires 3600 hours of machining time" are examples of constraint statements involving objects, object properties, and their values.

In this framework, the notion of a *system* is characterized as: a collection of objects standing in particular relations and exhibiting particular behavior prescribed by a collection of constraints. In manmade systems, this characterization is usually extended to include the notion of achievement of some goal. It is the behavior and relationship influencing role of the collection of constraints that distinguishes the notion of a system from the more general notion of a situation or state of affairs. Specifically, not all of the properties of, or relationships among, objects in a system are relevant to the system. In fact, particularly in manmade systems, the constraints may specify that particular properties and relations (e.g., equal opportunity constraints) cannot influence the behavior of the system. Because of their distinguished behavior-determining role, constraints have long been the focus of studies directed at both understanding and better controlling our natural environment. In ecology, for example, constraints between living organisms are studied to better understand and maintain the delicate balance of nature. In chemistry, discovering constraints among basic elements and reactive processes comprise much of the discipline. Similar examples could be considered in physics, thermodynamics, medical physiology, and so forth. The study of constraints, however, is not confined to the natural sciences. Similar examples can be found

in the study of man-made objects and systems. Constraints manifest in the design process, for example, are relations among properties or variables of the proposed artifact and its environment or context [Maher 89]. Design constraints establish the rules, requirements, relations, conventions, and principles that designers must use to synthesize design solutions [Gross 87].

Consistent with the above framework, *business systems* can be viewed as a collection of objects behaving to perform one or more business functions under the influence of constraints to accomplish a particular goal. Figure 2 illustrates the variety of business systems found in typical manufacturing enterprises. In the analysis of the collection of constraints relevant to a particular business system, it is useful to characterize where a constraint is enforced and controlled relative to a system. Therefore we characterize constraints as: *in* (enforced *in*ternal to the system), *on* (enforced externally on or controlled externally to the system), *of* (possessed by and hence controllable within the system) and *between* (among) (constraints which link business systems as objects together to form larger systems). The performance of a business system, whether operating independently or in concert with other business systems, is governed by constraints.

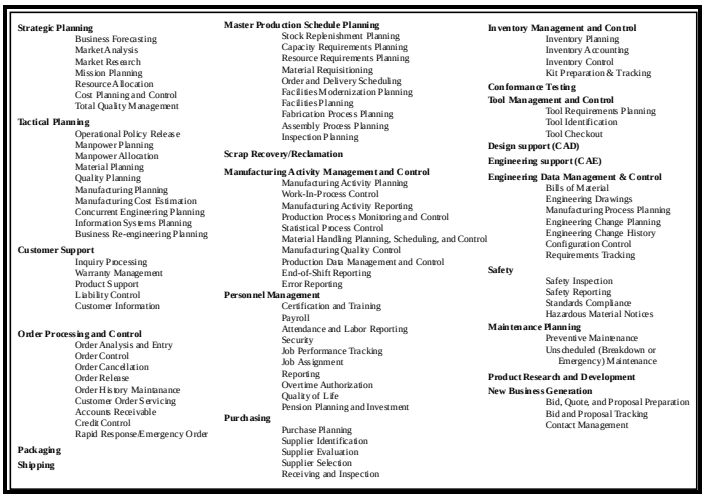


Figure 2
Typical business systems

Constraints can be broadly classified as either *enabling* or *limiting* within a given context (See Figure 3). Although the term "constraint" often evokes images of negative influence or control, constraints serve the vital enabling role of establishing the system. Tolerances between mating parts, for example, establish the constraints required to ensure correct fit. Fiscal management policies and accounting procedures maintained within a company not only keep a pulse on the health of the business but facilitate the prevention of unmanageable debt, fraud, and waste. Both the limiting and enabling aspects or constraints are evidenced in alternative definitions for a constraint in the literature. For example, Eliyahu Goldratt defines a constraint as "anything that limits a system from achieving higher performance versus its goal" [Goldratt]. Other definitions state that constraints are "the rules, requirements, relations, conventions, and principles that define the context of designing

[Gross, Ervin, Anderson, Fleisher]," "specifications, requirements, needs, performance measures, and objectives" [Ullman, Dietrich, and Stauffer], and "a characteristic of the environment, or of the artifact as currently conceived, [that] rules out or against potential settings of design variables" [G.F. Smith and G.J. Browne]. In other words, whenever the term is used there is an implicit thought that a constraint can be an enabling or limiting factor in design or on performance.

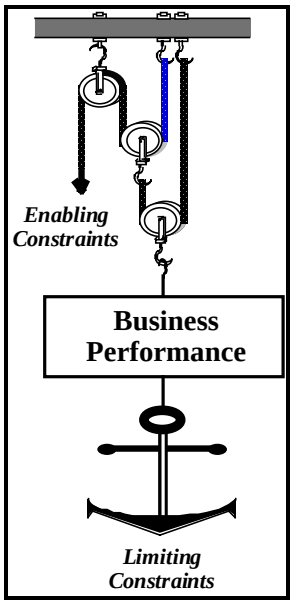


Figure 3
Constraints can be enabling or limiting

Limiting constraints are, of course, the most obvious since they tend to manifest themselves through problematic symptoms. Bottlenecks, excessive costs, low quality, long development lead times, waste, and inefficiency are all symptoms of limiting constraints. Symptoms are one form of *evidence* that constraints exist within the system.

We define *evidence* as an indication, sign, or manifestation that supports or proves the existence of a constraint in a given context. Evidence of both enabling and limiting constraints can take many forms. Some of these include symptoms (observable evidence of a system failing to meet goals), operating instructions, procedure manuals, employee handbooks, regulations, specifications, policy manuals, project files, design models, and so forth. These are not the constraints themselves but an indication, sign, or manifestation of possible constraints. Policy statements written in a policy manual, for example, would provide evidence supporting the proposition that there are constraints whose description is found in the policy manual. If the policies are not maintained or enforced, however, no constraint exists.

The need for maintaining or enforcing a relation to qualify as a constraint implies the need for some agent or system; i.e., the existence of a constraint implies the existence of a system that maintains or enforces the relationship. One should be cautions, however, not to confuse a constraint with the system that maintains the constraint. A constraint, after all, is

simply a special kind of association between a relationship that is maintained and the system that maintains that relationship.

It should be obvious, however, that since constraints involve the use of a system in their maintenance or enforcement, constraints come at a cost. In the natural world, balance sheets and cash flows are not used to reflect the cost of constraints. Instead, the maintenance of natural constraints is reflected in terms of energy expended or transformed. Business constraints, however, are maintained by business systems, and operating business systems costs time and money.

Motivation for collecting and managing business constraints

Given the influence constraints have on overall business performance, one would expect typical organizations to spend as much time and effort on managing constraints as, for example, on managing the objects that comprise business systems. Evidence of some level of constraint management activity is abundant, although it may not be so recognized. Authority is given or denied, policies are changed and assignments made, standards are developed and enforced, etc. However, decision-makers and system developers alike often have only limited support to identify and manage constraints effectively.

Effective constraint visibility and management enables decision-makers to recognize and respond appropriately to constraint problems typical of many businesses, as illustrated in Table 1.

Type I problem	The cost of maintaining a constraint exceeds the value of the constraint.
Type II problem	Constraints exist that no longer support the organization's goals.
Type III problem	The constraint causes unintended or undesirable effects.
Type IV problem	The agent or system (mechanism) responsible for maintaining the constraint fails to consistently or correctly enforce the constraint.
Type V problem	What was believed or intended to be a constraint lacks any explicit maintenance or enforcement mechanism.

Table 1
Some typical problems

Lacking visibility of constraints and effective constraint management support, business owners, strategic and tactical planners, and process owners soon fall prey to outdated or ill-suited constraints that both limit performance and frustrate attempts to seize opportunity. Systems overburdened with outdated or inappropriate constraints unnecessarily

levy costs in overall business performance and consume resources that could be better used elsewhere. According to Eliyahu Goldratt, a notable industry consultant in constraint-driven change, outdated rules, policies and procedures are the main cause of limited organizational performance. In such an environment, decision-makers soon fall prey to a reactive stance toward constraints. Problematic symptoms, and hence limiting constraints, take center stage, demanding most if not all of the decision-maker's attention. Lacking visibility on the current constellation of constraints under which the business operates, decisions are made without the benefit of tools to anticipate or predict downstream effects. Increasingly, the limitations of heavy reliance on intuition and experience alone become obvious as decisions made to solve problems at a local level create unanticipated negative effects on the overall enterprise. Opportunities, too, take a back row seat as resources and time become increasingly scarce.

Interestingly, while many negative symptoms are most effectively dealt with by changing or eliminating existing constraints, many decision-makers instead add new business constraints. This tendency may actually exacerbate the challenge of business constraint identification and management. Informal mechanisms of business constraint management thus tend to grow increasingly inadequate as the organization adapts to changes in the business environment, as the number and scope of business systems increase, and as the corporate memory of constraint development history fades with time or is lost through personnel turnover.

Benefits of constraint identification

Constraints initiate, enable, govern, and limit the behavior of objects and agents to accomplish the goals or purposes of a system. If we want to change the behavior of a system for what ever reason (e.g., improve its performance, efficiency, or effectiveness) we need to know what the relevant constraints are. Business constraint identification and management provide decision-makers with increased visibility on the business constraints that govern achievable performance. Discovering, cataloging, and maintaining business constraints thus enables decision-makers to deal more effectively with constraint-related problems (See Table 2).

The ability to simply catalog business constraints can assist decision-makers in both designing and prioritizing constraints relative to organizational goals. Knowledge of interrelationships among constraints also enables more reliable performance predictions and change impact assessments. By identifying and eliminating outdated or unnecessary constraints, costs are eliminated, improvements in performance (e.g., schedule and quality gains) can be realized, and freed resources can be used to leverage new opportunities. Often these benefits can be realized without any additional investment in automation or information systems.

Constraint-related problem	Benefit of constraint discovery
The cost of maintaining a constraint exceeds the value of the constraint.	Enable decision-makers to identify and eliminate constraints that exceed the value they provide.
Constraints exist that no longer support organizational goals.	Enable decision-makers to identify and eliminate outdated or unnecessary constraints.
The constraint causes unintended or undesirable effects.	Enable decision makers to reengineer or eliminate constraints that produce unintended or undesirable effects.
The agent or system (mechanism) responsible for maintaining the constraint fails to consistently or correctly enforce the constraint.	Enable systems developers to identify and remedy system designs that fail to appropriately enforce constraints.
What was believed or intended to be a constraint lacks any explicit maintenance or enforcement mechanism.	Enable decision-makers to identify missing systems needed to maintain both precautionary and enabling constraints.

Table 2
Some benefits of constraint discovery

Somewhat less obvious, perhaps, is the fact that knowledge of constraints can yield new sources of information and expose misinformation. For example, knowledge of the constraint that trees add one ring each year to their circumference, when coupled with knowledge of the number of rings in a given tree, yields new information—the age of the tree. Similarly, knowledge of business constraints can be used to yield otherwise inaccessible information about the health and productivity of the business, to determine how quickly and cost effectively products within the company can be produced, to estimate what it takes competitors to produce their products, and to identify where changes can be made to achieve competitive advantage.

Motivation for a method to collect constraints

While it is easy to see the value of identifying and managing business constraints, it is not so easy to discover and catalog them. For one thing, it is common for people to confuse constraints with policy statements, symptoms, or the object(s) that maintain a constraint. Furthermore, like a river flowing over a riverbed filled with rocks of varying sizes at various depths, constraints often don't manifest themselves until they become a visible obstacle in the path of workflow. As the tides of business ebb and flow, constraints become more or less obvious. Recognizing and circumventing these obvious obstacles is only part of the constraint discovery undertaking. Successful navigation through the politics of the organization is also challenging in a constraint surfacing endeavor. Constraints are

mechanisms for extension of power. They also provide a framework that bolsters a sense of security within the workplace. These factors can result in reluctance or even animosity for any initiative directed toward surfacing and examination of constraints.

The purpose of a method for constraint discovery is to provide a systematic and reliable approach for business owners, strategic and tactical planners, systems developers, project leaders, and decision-makers to identify and document business constraints. Indeed, the most significant failing of current reengineering practice is the lack of such a method. The developments documented in this report toward establishing an IDEF9 Business Constraint Discovery method are intended to satisfy these needs. A number of methods provide partial support for constraint discovery. There are no methods, however, that explicitly distinguish between simple relations and constraints or which don't confine themselves to a very restricted set of constraint types (e.g., precedence constraints in IDEF3). Additionally, there are no methods that provide a systematic approach to discover, document, validate, and refine both enabling and limiting business constraints.

The payoff to an organization in having a method for business constraint discovery is that it facilitates mapping relevant constraints in an organizational system. Once these constraints have been cataloged they can be systematically examined and, if necessary, tuned or replaced to improve the performance of the system. Constraints often serve a dual role as both the glue and the rationale for a system. That is, the collection of relevant constraints often constitutes the description of why the system behaves as it does. From this perspective, IDEF9 provides a reverse engineering tool for the business engineer. It can assist him in discovery of the "logic" behind the design of an existing system or for specifying the logic of a "To-Be" system.

The intended contribution of the IDEF9 method is to guide practitioners in rapidly and reliably 1) discovering, 2) displaying, 3) characterizing, and 4) validating business constraints. Given the wide range of anticipated users, we have designed the prototype IDEF9 method to be easily used by a personnel with varying skill levels and representing all segments of the business.

Users and beneficiaries of constraint discovery

Two broad categories of users for an IDEF9 Business Constraint Discovery method can be considered. First among these users are those who would directly apply the method to identify, document, validate, and refine the corporate library of constraints. This class of users ranges from those who manage constraint discovery efforts to those who perform the detailed evidence collection, analysis, interviewing, and so forth. The individuals targeted as direct users of IDEF9 include managers and practitioners of enterprise improvement initiatives (e.g., Total Quality Management, Business Reengineering), strategic and tactical planners, knowledge workers within the organization, and systems developers (both internal and external to the organization). Strategic and tactical planners will gain tremendous leverage through the use of IDEF9 as critical assumptions are questioned, external and internal forces are identified and analyzed, future challenges and opportunities are predicted, and as needs and requirements for supporting business systems are developed. System developers are supported by providing a mechanism to discover and refine constraints

through the process of specification and design. The second class of users represent those individuals who do not necessarily make direct use of the method but use the products of an IDEF9 application effort. Business owners and managers are among this set of users. For them, the most visible element of an IDEF9 Business Constraint Discovery method is not the set of techniques used to extract, validate, and refine constraints from the domain but the graphical language facilities used to display constraints. These users will apply the knowledge captured in constraint libraries to identify patterns, perform change impact assessments, and to identify new avenues of potential improvement.

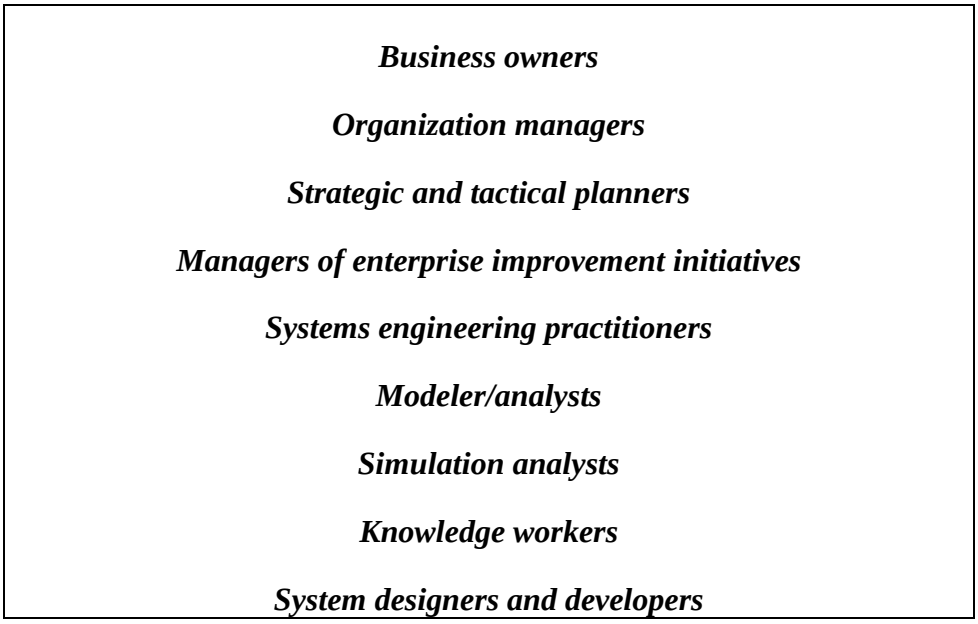


Figure 4
Targeted users of IDEF9

Potential Applications for IDEF9

The IDEF9 Business Constraint Discovery method was designed to assist in the discovery and analysis of constraints in a business system. A primary motivation driving the development of IDEF9 was an acknowledgement that the collection of constraints that forge an enterprise system is generally poorly defined. That is to say, the knowledge of what constraints exists and how those constraints interact is at best incomplete, disjoint, distributed, and often times completely unknown. This situation is not necessarily alarming, just as a living organism need not be aware of the genetic or autonomous constraints that govern certain behaviors, an organization can (and most do) perform well without explicit knowledge of the glue that structures the system. However, if the desire exists to modify the business to improve its performance or adapt to market, product, or process changes in a predictable manner, then knowledge of these constraints is as critical as knowledge of genetics is to the genetic engineer.

Knowledge of business constraints can benefit or aid efforts like Business Process Reengineering (BPR), Total Quality Management (TQM), strategic planning, constraint-

driven information systems design, Activity Based Costing (ABC), and so forth. Each of these efforts represent activities aimed at improving organization performance by striving to break away from outdated rules governing how we organize and conduct business. Recognizing the constraints in which the business currently operates is the first step to finding imaginative new ways to conduct business. Once those constraints have been recognized, outdated or burdensome constraints can be eliminated—providing an effective means for achieving both incremental and quantum leaps in performance. An on-line library of constraints, indexed so as to be able to display the constraints relevant to a given business context (e.g., doing business with DoD, doing business in the semiconductor manufacturing industry).could then provide additional support for leveraging new opportunities. Using the knowledge contained in an on-line library of constraints, business owners and system developers could explore opportunities to expand into new areas of business and rapidly determine the costs of maintaining the constraints necessary to operate in those environments.

APPROACH OVERVIEW

The approach taken by the IICE methods engineering team to initiate developments toward a business constraint discovery method leveraged both the 1) knowledge gained through studying common method engineering practice and 2) experience in developing other analysis and design methods. The purpose of this section is to provide a general overview of that approach.

Figure 5 provides a process-oriented view of the approach used to develop prototype IDEF9 method concepts, a procedure, and candidate graphical language elements.

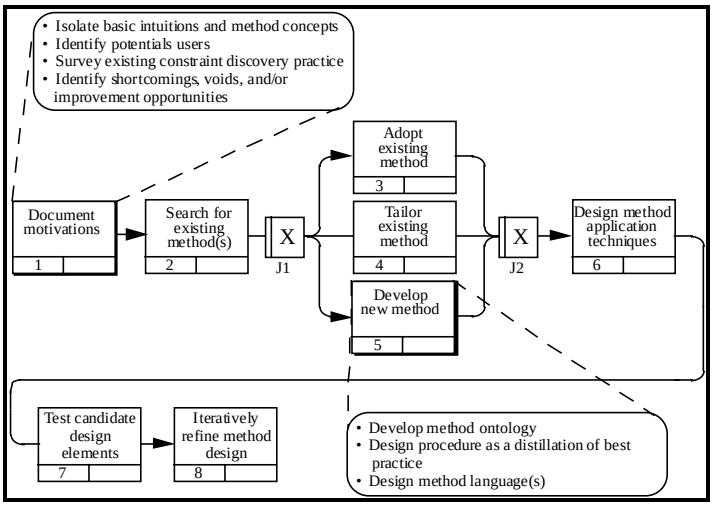


Figure 5
Process description of the IDEF9 development approach

As is evidenced by this figure, one of the basic strategies of methods engineering is reuse. Whenever possible, existing methods that address the identified needs are adopted. The next option is to find methods that with minor modification can satisfy the identified needs. This option is an attractive one provided the scope of modification does not necessitate a fundamental change in the basic concepts or design goals of the method. Only when neither of these options is viable should method designers seek to develop a new method.

A knowledge engineering approach is used as the predominant mechanism for both method enhancement and new method development. In other words, with very few exceptions, method development involves isolating, documenting, and packaging existing practice for a given task in a form that promotes reliable success among novice practitioners. Expert attunements are first characterized in the form of basic intuitions and method concepts. These are often first identified through analysis of the techniques, diagrams, and expressions used by experts. These discoveries aid in the search for existing methods that can be leveraged to support novice practitioners in acquiring the same attunements and skills. New method development is accomplished by establishing the scope of the method, refining characterizations of the method concepts and intuitions, designing a procedure that provides

both task accomplishment and basic apprenticeship support to novice practitioners, and by developing a language(s) of expression. Method application techniques are then developed outlining guidelines for use in a stand-alone mode and in concert with other methods. Each element of the method then undergoes iterative refinement through both laboratory and field testing.

Although much progress has been made toward the development of an IDEF9 Business Constraint Discovery method, considerable testing, analysis, and refinement is needed to achieve full maturity. A summary of the current state of IDEF9 is provided below.

SUMMARY OF RESEARCH FINDINGS

Basic concepts for constraint discovery

A thorough understanding of IDEF9's basic concepts is needed to effectively apply the procedural and language components of the prototype method. Among these are the following concepts:

1. Constraint
2. Context
3. Evidence
4. Effect(s) of a constraint
5. Symptom
6. System
7. Rationale for the constraint
8. Goal

The purpose of this section is to describe these concepts and provide some examples of each.

Constraint

In the IDEF9 method, a constraint is defined as a relationship that is maintained or enforced in a given context. Policies, rules, conventions, procedures, contracts, agreements, regulations, and societal and physical laws that are maintained within the enterprise establish the defining structure for the enterprise. These items are the mechanisms for forging relationships between people, information, material, and machines to make a system. If you view an enterprise as a machine, constraints form the architecture and the programming language that define the behavior of that machine. If you view the enterprise as an organism, they form the control structure of that organism, from the genetic code level through the autonomous stimulus response level, to the cognitive behavior level.

Constraints are expressed in *constraint statements*. Some examples of constraint statements might include:

1. Protect against liability.
2. Minimize in-process inventory.
3. Maximize cost recovery.

4. Load first-class passengers, families with small children, and disabled passengers before all others.

5. Load the aircraft beginning with passengers seated in rear.

6. All projects produce a final report.

7. Projects whose contract value exceeds $10M require an additional cost report.

8. An individual must maintain a current license to operate a vehicle.

9. Only cleared personnel may enter the secure area.

10. A group leader must be a member of the management team.

Notice that use of the imperative form is not necessarily the only form that a constraint statement might take. Furthermore, finding a statement that uses the imperative form does not necessarily ensure that one has found a constraint. For example, constraint #9 above may have appeared as the command "Challenge all unidentified personnel within the secure area."

Throughout the process of constraint discovery, analysts identify *candidate constraints*, or relations suspected to be ones that are maintained or enforced in a given context. Candidate constraints are first substantiated by the analyst through examination of the data collected. Candidate constraints are then challenged by domain experts. This process eventually leads to a refined collection of validated constraints and context characterizations establishing under what conditions the constraints hold.

Context

The term *context* refers to a distinguished set of conditions. Each context must be uniquely distinguished within a constraint model. A *context label* is a short descriptive phrase (e.g., when removing asbestos, procuring computer hardware and software, [being] at the construction site) used to convey a general understanding of the context boundaries. A more in-depth look at what comprises the context is generally needed, though, to distinguish one context from another. In other words, a label alone may not suffice in distinguishing one context from another. For this reason, those applying IDEF9 should catalog the minimal set of *essential* facts or conditions that uniquely distinguish one context from another. *Accidental* facts and conditions may also be noted. However, those facts or conditions that must hold (i.e., essential facts or conditions) define the minimal set needed to bound the context.

Evidence

We define *evidence* as an indication, sign, or manifestation that supports or proves the existence of a constraint in a given context. Just as chemical reactions are manifested by the resultant by-products, the existence of a constraint will manifest itself through some form(s) of evidence. For example, a candidate constraint statement such as "All purchase requests

require both project manager approval and senior company official authorization" might be hypothesized from the presence of one signature block for the project manager and another for an authorization official on the company's Purchase Request form. The Purchase Request form is said to be evidence supporting the proposition that there is a constraint requiring project manager approval and senior company official authorization on all purchase requests. Evidence must be grounded, however, in the appropriate context and validated before the existence of the constraint can be established. For example, purchase requests requiring authorization may be applicable only to purchases made under contracts that disallow certain purchases (e.g., government contracts precluding the purchase of computer hardware and software). Overhead projects, on the other hand, may require only project manager approval to purchase the necessary material.

The most easily identified evidence of a constraint is the existence of a problematic symptom. Such a condition may indicate the existence of a limiting constraint, the lack of an enabling constraint, or both. For example, excess in-process inventory and the existence of bottlenecks at various points in the production line may indicate the existence of constraints that serve to maximize the efficiency of individual processes in the production line or the lack of existence of constraints on the manner in which raw material is ordered and jobs released to the shop floor. Limiting capacity constraints might be identified by finding the places in the production line where bottlenecks occur. An enabling constraint wherein the entire process is set to match the rate at which the slowest process in the production line would maintain the same throughput and minimize in-process inventory. Another source of evidence are documents outlining policies, procedures, requirements, designs, and so forth. Forms with signature blocks, operating instructions, procedure manuals, handbooks, regulations, standards, specifications, policy manuals, project files, and design models are a representative set of possible evidence.

Effect(s) of a constraint

The term *effect* is used to describe, in general terms, something that inevitably follows an antecedent (as a cause or agent). Constraints initiate, enable, govern, and limit the behavior of objects and agents in the system. These behaviors are generally referred to as *effects*. Constraints also establish cause-effect chains that propagate effects through the system. Thus, effects are often considered either *direct* or *indirect*. Effects may also be *intended* or *unintended*, and *desirable* or *undesirable* in a given context. For example, a manufacturing company may have a constraint to collect metrics reflecting the productivity of individual shops. Presumably, this constraint would provide decision-makers with the visibility needed to identify where excess capacity exists and where additional resources are needed. This constraint may give rise to an unintended, direct effect wherein shop foremen create work for their people to keep them busy, and thus inflate the data reviewed by decision-makers. An indirect effect is the creation of excess in-process inventory and downstream bottlenecks.

Symptom

Symptoms are something characteristic or indicative of a condition impairing normal functioning of a system. Symptoms provide subjective evidence of a condition that is used to

aid in correctly diagnosing the condition. That is, the same symptoms may be indicative of one or more conditions impairing normal functioning. A fever, for example, may indicate the presence of an infection or the flu. In business systems, the lack of some core competency may give rise to poor quality products, late delivery, and so forth. While symptoms are observed failings of a system they are often confused with *concerns* which are "possible" failings of a system.

Although symptoms are not the condition themselves, they are often problematic. Perhaps for this reason, domain experts often refer to symptoms as *problems*. *Problem*, however, often connotes the *source* of distress or difficulty and may therefore lead to the wrong conclusions. The IDEF9 constraint discovery process leverages domain expert attunements to symptoms, probable causes, and effects to assist users in identifying candidate constraints or the lack of needed constraints. Symptoms are the point from which cause-effect constraints are hypothesized relative to other symptoms. For example, a production supervisor may describe frequent failures of Integrated Circuits (ICs) as among the key problems that they are attempting to address. The production workforce, however, may view those failures as a symptom of high turnover rates, contamination, improper handling, insufficient training, and so forth. Manufacturing engineers may attribute the failures to poor moisture control, cracked dies, or broken and bent leads. Design engineers may attribute failures to oxide or silicon defects. Each of these observations reflects knowledge of constraining relations which themselves may be symptoms of other conditions.

System

A *system* is defined as a collection of objects standing in particular relations and exhibiting particular behavior prescribed by a collection of constraints. In manmade systems this characterization is usually extended to include the notion of achievement of some goal. *Business systems* can be viewed as a collection of objects behaving to perform one or more business functions under the influence of constraints to accomplish a particular goal. Several examples of business systems comprised of multiple objects are provided in Figure 2 above.

The systems of primary interest within IDEF9 are systems that maintain or enforce a given constraint or set of constraints. Examples of a system that maintains or enforces a constraint might include individual objects (e.g., particularly an active object or agent such as an account manager or an authorization official) or collections of objects (e.g., an accounting system or a prime contractor).

Rationale for the constraint

The set of beliefs motivating the establishment and maintenance of a constraining relationship is called the *rationale* of a constraint. This set of beliefs includes those held to be true when the constraint exists as well as those held to be true in situations where the constraint does not exist. The rationale of a constraint is documented to assist with periodic review of the currency and relevance of the constraint.

Goal

A *goal* is defined as an object or end that one strives to achieve. Business systems exist to accomplish a particular goal or set of goals. Goals may stand in a number of relationships with other goals. Among these are *depends-on-existentially*, *implies*, *is-part-of*, and so forth. Ideally, each goal within the business will be oriented so as to contribute to the accomplishment of an overall goal or set of goals. Goals, however, are highly dependent on the environment and are therefore subject to frequent changes and reprioritization. Changes in the organization's goals in turn motivate changes to the constraint set used to direct the organization toward achieving those goals.

IDEF9 Procedure[5]

This section presents a prototype procedure for constraint discovery, validation, and refinement. The procedure presented in this section assumes a large constraint discovery effort involving a team approach. Projects that are narrower in scope may not require all the activities described herein. As with all methods, the procedure of application depends largely on the purpose for which the method is being used. Those undertaking a constraint discovery project are therefore encouraged to prepare a detailed method application guide at the beginning of the project.

Constraint discovery is an evolutionary process through which candidate constraints are identified, validated, and refined. In general, when using IDEF9 to discover and document constraints, the following six steps are applied recursively:

1. Collect - Acquire observations and sources of evidence for constraints.

2. Classify - Individuate contexts, objects, object types, properties, and relations.

3. Hypothesize - Postulate candidate constraints from the data and evidence acquired.

4. Substantiate - Generate or collect examples to determine which candidate constraints can be shown to merit promotion to the status of a constraint.

5. Challenge - Involve domain experts in testing the conclusions of analysts as to the validity of their conclusions.

6. Refine - Filtering, improving, adjusting, and adding detail to constraint characterizations.

These steps are embodied in the prototype IDEF9 procedure presented below. The activities comprising IDEF9's procedure should be considered "modes of thought" rather

[5] Significant reuse of the procedural components from the IDEF3 Process Description Capture and IDEF5 Ontology Description Capture methods facilitated the development of the prototype procedure description that follows.

than sequential steps. Users should not expect to apply these activities in a strictly sequentially manner, or that organizing activities by project phases necessarily defines when those activities start or stop. Rather, phases reflect which modes should or do predominate during a given interval of time. Thus, modes of activity may be organized into phases to assist with management of the project. The following section provides a functional description of the modes of activity constituting IDEF9's procedure, thus establishing a basic framework for constraint discovery.

Mode Zero: Define the Project

The constraint discovery team must establish the purpose and scope[6] of the constraint discovery effort as early as possible in the project. The purpose statement provides a "completion criteria" for the constraint discovery effort. The purpose is usually established by 1) prioritized objective statements for the effort, 2) statements of needs that the constraint discovery effort must satisfy, and 3) questions or findings that the client wants answered. The scope of the project is established by a set of statements that bound or delimit the area of the domain addressed by the project. In other words, scope statements identify the specifically targeted areas of constraint discovery activity and identify those that are explicitly ignored.

The purpose and scope can rarely be determined completely and accurately in advance. The client often revises his list of needed findings or questions as the data starts being compiled. The area an analyst thinks will lead to the answer often turns up leads in other areas that were not considered within the scope. The purpose and scope generally evolve during the initial part of the project. The purpose and scope of an IDEF9 effort are captured on an IDEF9 Project Summary Form similar to the one shown in Figure 6.

IDEF9 Project Summary Form

Project Title: _____

Project Leader: _____

Purpose: _____

Context: _____

Major in-scope situations:	Major out-of-scope situations:

Figure 6

[6]One of the central concepts in constraint discovery is the notion of a context in which a constraint holds. A different meaning for *context* is generally applied among IDEF method practitioners using other IDEF methods. *Context* is used with other IDEF methods to describe the scope or boundary of the project. To avoid unnecessary confusion, *scope* has been adopted for IDEF9 when describing the boundaries of the project.

IDEF9 Project Summary Form

Define the purpose

Defining the purpose is an important initial step in the constraint discovery effort. Normally a purpose statement will be centered If taken for granted or ignored, project personnel are likely to find the results of their efforts ignored by or of little use to the client. Without a purpose statement, the only completion criteria is the budget and time allocated to the effort. Conversely, with a regularly reviewed and clearly defined purpose, the project can often be completed in a budget much less than that anticipated. Defining the purpose involves 1) listing the stated objectives of the client and the specific source(s) of each (e.g., person, project, or organization), 2) defining the information goals of the project in terms of how the constraint information will be used, and 3) establishing priorities among the stated objectives and information goals of the effort. The process of developing a purpose statement can be facilitated by involving the client in answering questions like the following:

1. What problematic symptoms, concerns, or opportunities are of the greatest interest to the client?

2. Who will use the constraint information once it is available?

3. What question(s) does the client need answered?

4. What issues are behind the need for constraint discovery?

5. What decisions are behind the need to identify constraints?

Establish the Scope

Once the purpose of the effort has been characterized, it is possible to define the scope of the project. Defining the scope of the project begins with delineating the boundaries of the constraint discovery effort and documenting those boundaries in a set of scope statements. Ideally, scope definition should identify only those areas that are relevant to the needs of the client.

An effective mechanism for defining the scope of the project is identifying the important situation type(s) or context(s) to be considered and those that, although related, fall outside the project boundaries. Characterizing the context(s) of interest may begin at a very course-grained level by developing a descriptive phrase (such as with an adverb phrase like *working* for government agencies, *disposing* of hazardous materials) for the context(s) of interest accompanied by a brief description. Characterizing the key context(s) of interest involves achieving a consensus among constraint discovery team members on both the title and paragraph description for the context(s). It is common for differently named contexts to be nearly identical. Conversely, it is also common for different contexts to be named the same. The similarity or dissimilarity among contexts considered in scoping the project will initially become evident through the development of paragraph descriptions. Consensus among team members may require more fine-grained definition of the context(s), particularly

as the team members review the purpose and scope statements periodically through the project. When necessary, more detailed characterizations of the context(s) of interest can be accomplished by identifying the participating objects, relations, and facts that must hold in the context(s). Additionally, those contexts that impact or are directly related to the in-scope context(s), but which are outside the scope of the project, should be identified. Those intimately familiar with the domain must be relied upon to actually identify the context(s).

Scope and level of detail decisions are tentative at this stage of the project and should be updated as the constraint data becomes available. An astute project leader will regularly assess the adequacy of the constraint data captured against the specified needs and information goals of the client.

Mode One: Organize for Data Collection

Once the initial project purpose and context have been determined, the task of organizing for data collection can begin in earnest. At this point, the makeup of the project team will be solidified, team member roles will be established, and scenario development responsibilities will be assigned to team members.

The following roles are normally assumed by personnel involved in a constraint discovery effort.

1. Analyst: The IDEF9 expert who will be the primary developer of the IDEF9 constraint models.

2. Client: The person or organization requesting the constraint discovery effort development.

3. Domain expert: The knowledge source person in the application domain of interest.

4. Primary contact: The individual who acts as the interface between the analyst and the domain expert.

5. Project leader: The person ultimately responsible for the entire constraint discovery effort.

6. Reviewers: Persons knowledgeable in the domain and/or the IDEF9 method responsible for reviewing and approving draft models and documents. Reviewers authorized to make written critiques of IDEF9 schematics are *commentors*. The remainder are *readers*. Both team members and domain experts can be reviewers.

7. Librarian: A person assigned the responsibility of maintaining source material logs and files of documents, making copies, distributing kits, and keeping records.

8. Team members: All personnel involved with the IDEF9 constraint discovery effort.

Among the roles assumed within the team is that of the project librarian. With large systems, the role of the librarian is essential. In smaller efforts, that role may be assumed by the analyst. In establishing the librarian function, the project leader assigns an individual(s) to be responsible for collecting, cataloging, controlling, and distributing source material, kits, glossaries, files, and so forth throughout the project. Additionally, the librarian function is responsible for assembling reference models and materials from external sources that can be used to accelerate team efforts. A glossary of terms may also be maintained by the librarian as a reference to be used during interviews to ensure that analysts understand terminology that is unique to a discipline, industry sector, company, or company segment. Whether maintained by the librarian, or informally shared among analysts, the glossary of terms will grow and undergo incremental refinement throughout the project.

A pivotal task in organizing the data collection effort is identifying the key sources of knowledge and information in the domain. Working with the primary contact, the project leader or analyst compiles a list of experts to be interviewed. In compiling this list, it is helpful to obtain background information about each expert from the primary contact. This includes information about the responsibilities, current assignments, and other areas within or related to the domain in which the expert has experience. The name, location, and telephone number of the expert(s) should also be recorded.

Throughout the data collection effort, other valuable sources of information will be sought and identified. Some of these include operating instructions, procedure manuals, employee handbooks, regulations, policy manuals, project files, reusable IDEF models, and models derived through the use of other methods and techniques. These items often constitute evidence of constraints themselves or provide references to available evidence within the domain.

In addition to organizing the structure of the team, the project leader also needs to organize the activities of the team. Organizing constraint discovery activity may begin by casting the general IDEF9 procedure into a more formalized method application guide tailored to the specific needs of the project. A method application guide is intended for use by the analysts on the constraint discovery team and outlines a project-specific application of the IDEF9 procedure tailored to meet the needs of the effort. Among the items that may be included in the method application guide are modeling conventions to be used, standard outlines for interviewing domain experts, method and tool interface specifications, project library use procedures, and a standard glossary of terms. This guide may be accompanied by a project plan. A typical project plan will delineate phases of effort with clearly established tasks and milestones, intermediate and final deliverables, individual team member assignments, informal and formal reporting structures, and so forth.

Mode Two: Collect and Analyze Evidence

Having organized the team and outlined the approach, the team will begin constraint discovery by engaging in evidence collection. By direct interaction with domain experts,

constraint discovery team members document expert observations and collect evidence of constraints. These are later analyzed to form the basis for hypothesizing constraints within the domain of interest.

Prepare for Interviews

The most valuable mechanism of evidence collection is the interview. Interviews with domain experts afford the interviewer an opportunity to collect special insights, both into normal situations and the exceptions to the normal situations within the domain. Direct observation techniques, although often used to augment interviews with the domain expert, generally only afford the interviewer an opportunity to observe normal situations.

While the specific interviewing approach and format is likely to vary across projects, a few simple guidelines are recommended. Before the interview, the analyst should prepare a tentative agenda and some specific questions. Analysts are encouraged to prepare a brief outline of: 1) the purpose of the interview with the expert, 2) the topics to be covered, 3) the types of information being sought, 4) the authority for requesting the interview, and 5) the probing questions that can be used to motivate discussion. On large projects, project leaders may wish to include more formalized interview preparation guidelines and standards in a method application guide—including standard interview planning sheets, question templates, glossaries of terms, and so forth.

The ultimate success of the interview depends largely on the preparation made by the analyst. A number of activities contribute to successful preparation, each of which is left to the discretion of the analyst as dictated by the needs of the project. In general, the following activities are accomplished prior to the interview:

1. Schedule the interview and make necessary logistics preparations.

2. Establish the goal(s) of the interview.

3. Prepare candidate questions.

4. Anticipate the probable questions and concerns of the person being interviewed and be prepared to resolve concerns.

Once a list of experts to be interviewed has been compiled, an interview schedule can be developed. Interviews are normally scheduled with domain experts through the primary contact. Whether done through the primary contact or by more direct means, the analyst should make sure that the scheduled time and duration of the interview is coordinated with the person being interviewed and his or her supervisor.

Additional logistics considerations are also important to the success of the interview, such as finding and reserving a suitable location to conduct the interview and arranging for the necessary supplies. Analysts also generally find it useful to plan the attire they wear to the interview in order to convey a professional appearance and still set the interviewee at ease.

The goal(s) of the interview should also be established up front. In establishing the interview goal(s), analysts establish why the interview is being scheduled and what information is needed from the domain expert. Preparing a goal statement is often helpful if it is kept as succinct as possible so as to provide a general direction for the interview line of questioning.

Once the goal(s) of the interview has been established, candidate questions can be formulated. Candidate questions should be written down and organized into a logical sequence. With experience and practice, analysts will eventually become proficient in developing questions that are clear, that use words and phrases appropriate to the educational level and cultural background of the person being interviewed, and which invite rather than lead answers. In preparing candidate questions, it is often useful to explore the following topics:

1. What are the organizations goals and objectives?

2. What are the organization's Critical Success Factors (CSFs) and performance measures?

3. What are the organization's problematic symptoms?

The answers to these questions often provide valuable guidance in searching for and identifying business constraints. Statements of goals and objectives give strong indications of perceived environmental constraints and, with appropriate follow-through, can lead to the discovery of deeply-rooted belief systems and undocumented constraints. Quite often, many of the constraints discovered through lines of questioning centered around organization goals and objectives will reveal both enabling constraints and constraints that no longer support current goals. When this line of questioning is applied across different organization levels, hidden transformations between strategic and tactical goal structures may be revealed that in turn may be used to identify missing and/or inappropriate constraints. In a similar fashion, exploration of the organization's CSFs and performance measures yields important constraint information. In fact, there is not likely to be any more obvious evidence of existing constraints than artifacts of performance measurement (e.g., graphs, charts, reports). Constraints arising through the establishment of performance measures, while generally intended to be enabling, often drive unintended and undesirable behavior. Finally, valuable guidance in discovering constraints may be obtained through listing problematic symptoms and the influencing factors believed to be the underlying cause(s) of those symptoms. Symptoms of problems may be manifest in business systems as bottlenecks, excessively long cycle times, poor quality, high cost, and so forth.

In preparing candidate questions for the interview, analysts should be cautious not to over prepare. The exercise of writing questions down and analyzing the way they are formed, however, serves to build good interviewing skills. The time invested to this activity must be balanced, however, against the possibility that the questions formulated may or may not actually be used. Their necessity may be eliminated through the discovery of new information or the interview may follow a line of discussion that was not previously anticipated.

An element of preparation often overlooked by inexperienced analysts is the need to provide the person being interviewed with the information necessary to understand why they are being interviewed, what will be done with the information they provide, and what they can expect in return. Each interview, and particularly the first, should begin by establishing a mutual understanding of these items before attempting to satisfy the information needs of the analyst. The following list is representative of the topics and concerns that the analyst should be prepared to address [Harrington 1991].

1. Why the interview is being conducted.
2. Who authorized the interview
3. Who else is being interviewed.
4. How the interviewee was selected and by whom.
5. How the information will be used.
6. Whether the person will be anonymous.
7. Whether the person will be quoted in summary findings.
8. What feedback the person will receive.
9. How the person might participate in the outcome of the process.
10. What is in it for the interviewee.
11. Why highly detailed, accurate information is important to the success of the interview and the project.
12. How the person plays a key role in an important process.

Interview Domain Experts

Interviews may be conducted at any time throughout the project with one or more of the following goals in mind:

1. To collect additional information.
2. To confirm and/or clarify previously collected information.
3. To validate candidate constraints with the domain expert.
4. To obtain leads for acquiring additional information.

Interviews with domain experts are critical. The analyst (interviewer) should create a positive and friendly atmosphere during the interview. The interviewer should attempt to convey to the domain expert the feeling that they are working together to discover constraints

and solve some problem for the organization. A novice interviewer should constantly remind himself that the expert is the one with the knowledge of how a organization should or does work. Generally, the expert is interested in helping and will often provide questions and lines of investigation that the interviewer had not thought of pursuing. The well-prepared interviewer will find that the expert will provide far more information than was expected, often covering topics the interviewer had not anticipated. In constraint discovery, this is the bonus for good preparation.

Analysts should be aware that domain experts often begin by describing rules, policies, procedures, and relations that should be maintained and those that actually are. Questions that help to distinguish desired operating conditions from the norm, and normal operating conditions from work-arounds or special cases can be helpful in guiding the interview. The main focus of the interview should be on rules, policies, procedures, and relations that are currently maintained or enforced (i.e., constraints), rather than "Should-Be" conditions that may not be maintained. When focused on constraints, analysts should also be cautious to avoid talking about "To-Be" constraints to avoid introducing bias in the domain expert's answers. Throughout the interview, constraint information provided by the domain expert needs to be faithfully recorded in a form that can be shared among all team members. Analysts should pay particular attention to the use of the imperative form in the description or in documents provided by the domain expert (e.g., *Complete* the attached job application form). Words like *must*, *will*, *shall*, *always*, and *never* are often included in imperative phrases (e.g., Applicants must complete the attached job application form before they can become eligible for an interview). However, neither the absence nor presence of these terms necessarily indicates a constraint. Logical quantifiers like *all*, *every*, *some*, and *none* also provide clues for discovering candidate constraints.

Collect and Catalog Evidence

As appropriate, analysts should also request copies of artifacts that constitute forms of evidence for constraints. Evidence of business constraints can take many forms including procedure manuals, instruction sheets, forms with fields for approval signatures, handbooks, policy manuals, regulations, specification documents, standards documents, strategic and tactical plans, organization charters or mission statements, efficiency reports, and so forth.

Other sources of evidence include analysis models (e.g., IDEFØ function models, IDEF1 information models, IDEF3 process descriptions, IDEF5 ontology descriptions) and design models (e.g., IDEF1X semantic data models, IDEF4 object-oriented design models) developed within areas of the domain relevant to the project. Displayed directly in IDEFØ function models are objects classified as controls and mechanisms with respect to a given activity. Objects modeled as controls often list artifacts within the domain (e.g., documents containing information about rules, policies, and procedures) that govern how the activity is or should be performed. Objects classified as mechanisms represent the means by which the activity is accomplished, thus providing valuable assistance in cataloging and validating constraints. The IDEFØ method thus captures some constraint-related information, albeit at a relatively course-grained level. Even IDEFØ controls identified below the artifact level cannot be assumed to necessarily represent constraints without further validation since the IDEFØ language does not explicitly capture which mechanisms maintain or enforce which

controls. Nevertheless, IDEFØ models remain a valuable source of information to the constraint discovery team. IDEF1 information models capture and display a specialized class of constraints; specifically, those constraints that are maintained in the domain through the information system. In other words, IDEF1 is used to model constraints for which some information objects have been designed and implemented. Similarly, IDEF3 process descriptions explicitly capture another specialized class of constraints. IDEF3 captures precedence and causality relations among processes and events within the environment. IDEF3 also captures constraints relative to the state change behavior among objects participating in a process. Ontologies developed using IDEF5 include characterizations of objects, object properties, and relations, thus providing a solid foundation for constraint discovery. IDEF5 ontologies also distinguish between defining versus non-defining and essential versus accidental properties and relations. Accessibility to IDEF5 descriptions can therefore greatly accelerate the process of constraint discovery. Design models (e.g., IDEF1X and IDEF4 models) also capture reusable constraint information. IDEF1X models capture the design constraints to which information system developers must conform. These constraints reflect business rules and policies that are to be implemented through the information system. IDEF4 models capture similar information with a specific target toward implementation in an object-oriented language.

All data that is collected during the course of the project should be logged on an IDEF9 Evidence Log as illustrated in Figure 7.

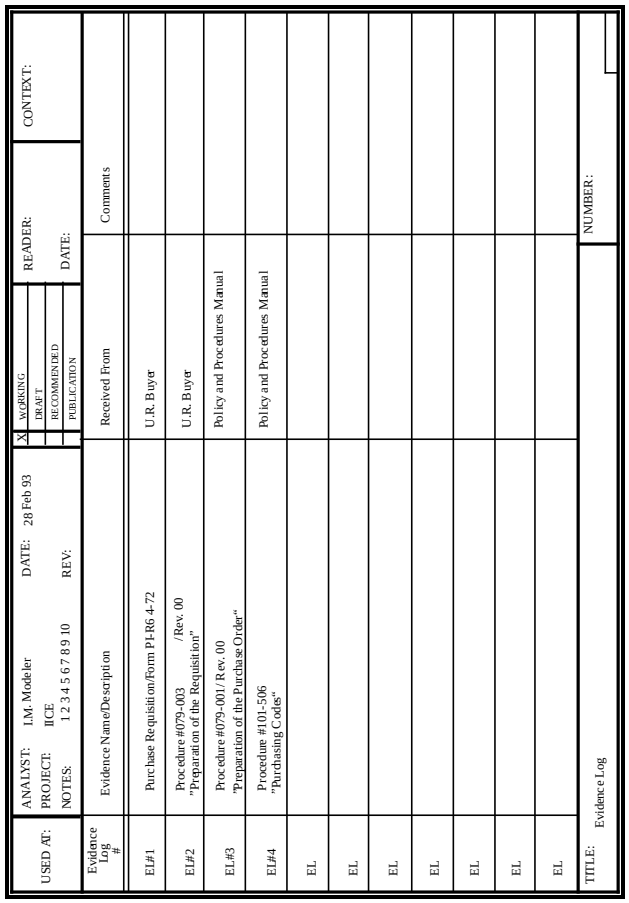


Figure 7
Evidence Log

Analyze Collected Data

Following data collection, interview notes are compiled, the Evidence Log is updated to reflect newly collected evidence, and the initial findings are cataloged into lists called pools. Among the pools found to be potentially useful in organizing and analyzing constraint-related information are the following:

1. Business goals pool
2. Performance measure pool
3. Symptom pool
4. Source statement pool
5. Context or situation type pool
6. System or object pool

7. System or object property pool

8. Relation pool.

In analyzing the collected data, analysts may also perform the following activities:

- Further refine and individuate contexts (e.g., working for KBSI, [being] at the construction site, acquiring replacement parts, [conducting] Programmed Depot Maintenance (PDM)).
- Associate existing business goals with context(s).
- Associate existing performance measures with context(s).
- Associate existing symptoms with context(s).
- Catalog the objects involved with context(s).
- Catalog object properties.
- Identify relations.

A focus on relations can be of great assistance in uncovering candidate constraints during analysis. Relations may be found between contexts, between objects and contexts, between object types and object instances, between object types and property values, and so forth. Table 3 below illustrates candidate constraint statements illustrating different kinds of relations.

	Context	System	Property
Context	• Working government contracts involves maintaining an auditable accounting system. • Drilling precedes reaming.	• Carry a current driver's license while operating a motor vehicle. • People are not allowed on the construction site without a hard hat.	• Children under thirteen are to be accompanied by a parent when attending PG-13 rated movies.
System		• People own cars. • John owns the Le Baron. • The tensile strength of steel is greater than that of iron.	• Birds have feathers. • Bill's hourly wage is $7.25. • First-line supervisors sign off all flight safety-critical maintenance operations.
Property			• The number of rings in a living tree corresponds directly to its age in years.

Table 3
Different relations illustrated by constraint statements

It is also often useful to classify constraints to assist in both discovery and downstream reuse of constraint information. Two criteria should be considered when developing a classification or taxonomy of constraints. First, the taxonomy should express *orthogonality* among categories, i.e., each category of the taxonomy should disjoint such that every element within the taxonomy's domain can be uniquely assigned. Second, the taxonomy should be *exhaustive* over the specific domain.

Both constraint granularity and coupling among constraints should also be considered when developing a taxonomy. Granularity is the level of abstraction used to represent the constraint. The more abstract the representation, the more difficult it is to assign a constraint to a unique category. For example, one of the constraints used in production management is "in-process inventory between stations should be kept balanced." The level of granularity represented by this constraint statement is very high. There are many factors that contribute to a balanced production line, e.g., machine capabilities, down-time, set-up time, material routing strategies, and so forth. More fine-grained constraint statements such as "the work in

process (WIP) for station A is 4 units or less" and "WIP for station B is under 10 units" permit unique classification to a subcategory of balanced production constraints called "WIP limits." Coupling among constraints may also need to be considered. That is, constraints often stand in relationships with one another that make it difficult to divide them into separate categories. Again, refining the granularity of the constraint representation often helps one to effectively categorize constraints.

Several possible classifications for constraints may be considered as a means to systematically analyze and manage both candidate and validated constraints. For example one might find it useful to classify constraints in terms of the measure on control that the organization has over the constraint's structure or very existence. Constraints could then be divided between those that are *volitional* (imposed by choice) and those that are *non-volitional* (no choice) in a given context. Constraints may also be classified as *enabling* or *limiting* relative to some goal in a given context. Constraints dealing with resources may broadly be categorized as *resource constraints*; those dealing with the capacity properties of systems as *capacity constraints*; those arising from the selection of one design strategy versus another as *design constraints*; those for which the rationale is largely unknown or poorly justified as *status quo constraints*; and so on.

The IDEF9 method does not prescribe one classification scheme or set of classification schemes over any others. The most appropriate classification scheme(s) will be determined by how the constraint information needs to be used. It is generally useful, however, to adopt several classifications to permit analysis of the constraints within and across those classifications. This analysis often leads to the discovery of new constraints and to previously unrecognized opportunities for improvement.

Mode Three: Hypothesize Candidate Constraints

Using source statements and the evidence collected, members of the constraint discovery team hypothesize *candidate constraints*. A candidate constraint can be thought of as a "first pass" at a constraint. A candidate constraint that can be supported by data collected is said to be *substantiated*. A candidate constraint is said to *mature* into a *constraint* upon successfully passing further validation testing.

Common ways in which candidate constraints emerge are as follows:

1. Candidate constraints are obvious to the modeler can be substantiated based on the evidences collected.

2. Constraints that emerge due to personal belief systems of the domain experts can be substantiated based on interview notes or the evidence collected.

3. Constraints that emerge from characteristic functions (e.g., height of a table) and can be substantiated by the data collected.

4. Candidate constraints that are suspected by the modeler but which cannot be supported by the evidence collected so far can be hypothesized and later substantiated (or unsubstantiated) through additional evidence collection.

For each candidate constraint identified, the following information will be recorded on a candidate constraint specification form:

1. Candidate Constraint ID#
2. Constraint statement
3. Constraint description.
4. Context ID#(s) (Context(s) in which the constraint holds)
5. Arguments of the candidate constraint
6. System or object(s) that maintain(s) the constraint
7. Constraint violation consequences
8. List of supporting evidence.

Mode Four: Validate and Refine Constraints

Candidate constraints must undergo a validation process to ensure that they are in fact constraints. The validation process can be divided into two parts, both of which should are applied to promote candidate constraints to the constraint pool. The first element of validation is characterized by analysts attempting to *substantiate* candidate constraints using collected evidence, interview notes, and direct observations collected by the team. The second element of validation involves engaging domain experts in challenging the candidate constraints that remain. Throughout the validation process, candidate constraints and constraints alike undergo a refinement process wherein derivative versions of constraint statements are proposed and tested, pool data is extended, and more in-depth characterizations are developed.

Substantiate Candidate Constraints

Once candidate constraints have been hypothesized, analysts will attempt to substantiate their hypotheses. In effect, analysts set out to prove to themselves that they have actually discovered constraints. Substantiating candidate constraints involves testing them against example instances of contexts to determine whether the relations thought to be constraining are in fact maintained. Any one of the following situations may arise while attempting to substantiate candidate constraints:

1. The candidate constraint is substantiated.
2. The candidate constraint is accepted after refining the proposed characterization of the context(s) in which the constraint holds.
3. The candidate constraint can be substantiated with slight modification.

4. Both candidate constraint and the context(s) in which it holds undergo slight modification to support substantiation.

5. The hypothesis of a candidate constraint is found to be unsubstantiated.

When discrepancies surface, analysts may simply need to refine their characterization of the holding contexts to establish the validity of the constraint given the currently available evidence. Alternatively, they may need to refine their characterization of the candidate constraint.

Either situation generally involves the need for additional data. A number of approaches are available for collecting additional information. Approaches that can be considered include:

1. Conducting follow-up interviews to answer questions and/or identify additional evidence.

2. Arranging for direct observation of the situation(s) included in the scope of the effort.

3. Revisiting source material with a new focus of analysis.

4. Conducting facilitated workshops.

The approach or combination of approaches used will be determined by both the nature of the information needed and the purpose for which IDEF9 is being used. Any additional data collection activity will involve making appropriate updates to previously collected data (e.g., updating the evidence log).

Candidate constraints having undergone this step in the analysis are migrated to an intermediate classification as either *substantiated* and *unsubstantiated*.

Challenge candidate constraints

Both substantiated and unsubstantiated candidate constraints are subject to domain expert review and validation. Thus, domain experts *challenge* the analyst's conclusions. If the constraint discovery team has been successful in collecting a strong body of evidence to justify its hypothesis, there is high probability that substantiated candidate constraints will be promoted to the status of a constraint. On occasion, unsubstantiated constraints will also be supported by new evidence provided by the domain expert at this stage of the process. The various steps involved in the validation of a constraint are:

1. The constraint discovery team provides domain experts with a list of substantiated and unsubstantiated candidate constraints together with the supporting evidence for their hypothesis.

2. The constraint discovery team interacts with domain experts to obtain and record feedback.

3. The constraint discovery team analyzes feedback obtained from the domain experts.

4. The constraint discovery team refines validated constraints and their associated context descriptions based on the acquired feedback.

Refine constraints

Refinement is a process of filtering, improving, and adding value to a product. The process of constraint discovery is itself a refinement process. Hence, refining constraints is an ongoing activity that occurs throughout the constraint discovery effort. Treating this activity as one of prominence, however, allows more focused attention on what is involved in the refinement of constraints. More precisely, the constraints themselves are not refined. Rather, the characterization of those constraints is refined. The resulting characterizations obviate the true value of having engaged in the constraint discovery process in the first place. The extent of refinement is largely determined by the purpose of the project, although companies interested in maintaining libraries of constraints will want to adopt standards for the information to be managed about business constraints. Among the items aiding in a full characterization of the discovered constraints, if not completed during analysis, are the following:

1. Identify correlations between contexts and business constraints that hold in those contexts. Contexts can be classified using any number of classification schemes. It is often most useful, however, to classify contexts based on the degree to which they share constraints.

2. Identify correlations between business constraints and the system(s) or object(s) responsible for maintaining the constraints. Because business constraints are defined as relations that are maintained or enforced in a given context, it should be obvious that it is important to identify the object(s) responsible for maintaining business constraints. Knowledge of the object(s) that maintain constraints is useful in helping identify and resolve resource contention problems, determining the impact of absent individuals, systems, and processes, and exploring alternative mechanisms to maintain a desired constraint.

3. Document the motivation(s) for the business constraint. The motivation of a constraint characterizes the justifications, intuitions, assumptions, and judgments giving rise to its existence. Among the assumptions that should be captured are the presumed consequences of the constraint not being in place. The elimination or reversal of any of these elements will help to identify those constraints that no longer need to be maintained.

4. Identify correlations between business constraints and organization goals and objectives. Both positive and negative correlations can be established between business constraints and the goals and objectives of the organization for a given context. Alternative contexts are likely to reveal different correlations than those for another context. However, the task of identifying these correlations

may be used to prioritize constraints of the system, provided the purpose of the project calls for leveraging constraint knowledge in this manner. Correlations between business constraints and goals enable downstream analysis of the impact of constraints on organization goals in addition to providing support for sensitivity, cause-effect, and influence analyses. Prioritization of constraints can also be performed for a given business situation.

5. Document correlations between constraints, performance measures, and effects. Performance measures are among the most obvious evidence of candidate business constraints. Performance measures often serve to drive behavior patterns within the company, at times in ways that were not previously anticipated or desired. When undesirable effects surface, it is often valuable to revisit both the performance measures chosen and the constraints maintained to provide management visibility on those performance measures.

6. Correlate observed effects (intended and unintended) and symptoms with business constraints. Cause-effect chains can be established by linking constraint interrelationships across contexts. Documenting the effects of a constraint helps establish these correlations.

7. Further classify constraints using classification schemes that are likely to provide the greatest downstream value to the organization. Several potentially useful classification schemes are presented in the *Basic Concepts* section.

IDEF9 language design research and development

The method language design process

The method language design process is highly iterative and experimental in nature. Unlike procedure development—where a set of heuristics and techniques representative of existing practice can be identified, merged, and refined—language designers rarely encounter well-developed graphical display or textual information capture mechanisms. When potentially reusable language structures can be found, they are often poorly defined or only partially suited to the needs of the method.

Critical to the design of a method language is clearly establishing the purpose and scope of the method. The purpose of the method establishes what needs the method must address. This, in turn, is used to determine the expressive power required of the supporting language. The scope of the method establishes the range and depth of coverage which must also be established before one can design an appropriate language design strategy. Scope determination also involves deciding what cognitive activities will be supported through the process of method application. For example, language design can be confined to only *display* the final results of method application (as in providing IDEF9 with graphical and textual language facilities that capture the logic and structure of constraints). Alternatively, there may be a need for in-process language support facilitating information *collection* and *analysis*. In those situations, specific language constructs may be designed to help method

practitioners organize, classify, and represent information that will later be synthesized into additional representation structures intended for display.

With this foundation, language designers begin the process of deciding what needs to be expressed in the language and how it should be expressed. Language design can begin by developing a textual language capable of representing the full range of information to be addressed. Graphical language structures designed to display select portions of the textual language can then be developed. Alternatively, graphical language structures may evolve prior to, or in parallel with, the development of the textual language. The sequence of these activities largely depends on the degree of understanding of the language requirements held among language developers. These may become clear only after several iterations of both graphical and textual language design.

Graphical language design begins by identifying a preliminary set of schematics and the purpose or goals of each in terms of where and how they will support the method application process. For each schematic, the central item of focus is determined. For example, in experimenting with alternative graphical language designs for IDEF9, a *Context Schematic* was envisioned as a mechanism to efficiently classify the varying environmental contexts in which constraints may apply. The central focus of this schematic was the *context*. After deciding on the central focus for the schematic, any additional information that should be captured or conveyed needs to be identified. Other concepts or relations to be included in the schematic are then identified.

At this point, two sets of graphical symbols have been explored. The primary focus has been on the information that should be displayed in a given schematic to achieve the goals of the schematic. This is where the language designer must determine which items identified for possible inclusion in the schematic are amenable to graphical representation and will serve to keep the user focused on the desired information content. With this general understanding, previously developed graphical language structures are explored to identify potential reuse opportunities. For example, while exploring candidate graphical language designs for IDEF9, a wide range of diagrams both from within the IDEF family and without were identified and explored.

Quite often, even some of the central concepts of a method will have no graphical language element in the method. For example, the IDEF1 Information Modeling method includes the notion of an entity but has no syntactic element for an entity in the graphical language[7]. When the language designer decides that a syntactic element should be included for a method concept, candidate symbols are designed and evaluated.

Thoughout the graphical language design process, the language designer applies a number of guiding principles to assist in developing high quality designs. Among these, the language designer seeks to avoid overlapping concept classes or poorly defined ones. They also seek to establish intuitive mechanisms to convey the direction for reading the

[7] The IDEF1X Semantic Data Modeling method uses *'entity'* to describe a different concept than that used in the IDEF1 Information Modeling method. In IDEF1X, an entity is represented explicitly in the method language and represents a set of things that share the same set of attributes.

schematics. For example, schematics may be designed to be read from left to right, in a bottom-up fashion, or center-out. The potential for clutter or overwhelmingly large amounts of information on a single schematic is also considered as either condition often makes reading and understanding the schematic extremely difficult.

Each candidate design is then tested by developing a wide range of examples designed to explore the utility of the designs relative to the purpose defined for each schematic. Initial attempts at method development, and the development of supporting language structures in particular, are usually complicated. With successive iterations on the design, unnecessary and unnecessarily complex language structures are eliminated.[8]

As the graphical language design approaches a level of maturity, attention turns again to the textual language. The purposes served by textual languages range from providing a mechanism for expressing information that has explicitly been left out of the graphical language to providing a mechanism for standard data exchange and automated model interpretation. Thus, the textual language supporting the method may be simple and unstructured (in terms of computer interpretability) or it may emerge as a highly structured, and possibly somewhat complex, language. The purpose of the method largely determines what level of structure will be required of the textual language.

In addition, as the method language begins to approach some level of maturity, mathematical formalization techniques are employed to help ensure that the emerging language has a clear syntax and semantics. Potential ambiguities with the correct interpretation of a given schematic, the presence of awkward language structures, and other language streamlining and simplification opportunities are often uncovered by scrutinizing the method language through the formalization process.

These general activities culminate in a language that helps focus user attention on the important information that needs to be discovered, analyzed, transformed, or communicated in the course of accomplishing the task for which the method was designed. Both the procedure and language components of the method also serve to apprentice users in developing the necessary skills and attunements required to achieve consistently high quality results for the targeted task.

Candidate schematic designs explored

A number of candidate schematic types were explored to support the constraint discovery process. The general questions used to guide the development of candidate schematics are as follows:

1. What step(s) of the procedure is a schematic needed or desired?

2. What information (type, amount, etc.) should the schematic convey?

[8] Language development activity for an IDEF9 method reached the initial levels of this stage of development during the IICE project. Some of the language designs explored during this process are presented later in the report.

3. How is the information to be conveyed? What will be the view (perspective) adopted? What graphical metaphor will be used (hierarchical, spider's web, network, sequential, etc.)

4. What is the syntax for the graphical language? List the different elements to be represented and the graphical symbol used for each element.

5. What is the semantics of each graphical construct?

In answer to these questions, six candidate schematics have been identified, subject to further testing and analysis. Among these were the following:

1. Context schematic

2. Constraint resource schematic

3. Constraint-relationship schematic

4. Constraint effects schematic

5. Goal schematic

6. Goal-relationship schematic

7. Symptom schematic

Each of the candidate schematics explored is described in the following sections.

Context Schematic

Purpose: The purpose for this schematic is to help users (1) establish and incrementally refine the scope of a constraint discovery effort, (2) display the constraints that hold in a given situation, and (3) rapidly identify shared constraints among distinguished contexts.

Viewpoint: Context-centered

Procedure component(s) supported:

- Mode Zero, Establish project context
- Mode Two, Collect and analyze evidence
- Mode Three, Hypothesize candidate constraints
- Mode Four, Validate and refine constraints

Graphical metaphor: Hierarchical

Candidate syntax:

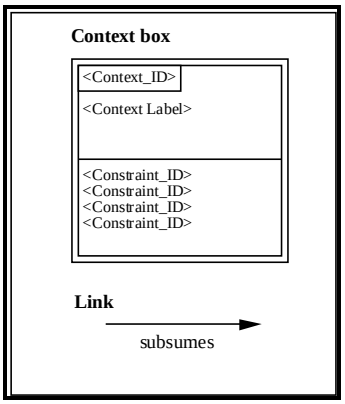


Figure 8
Candidate syntax for the context schematic

Example:

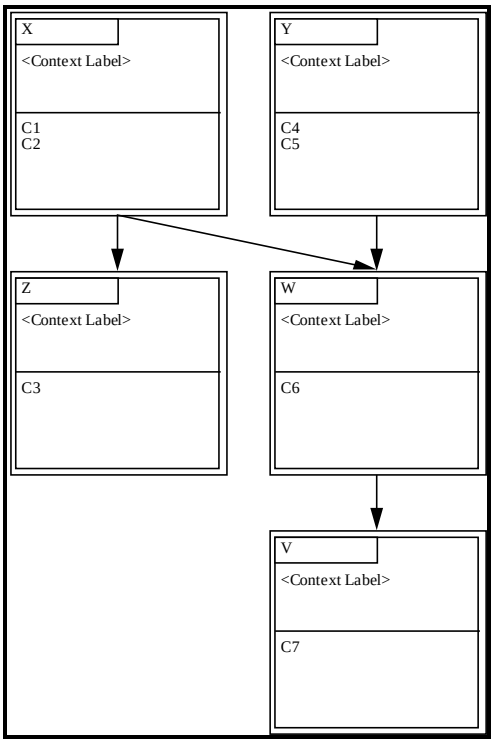


Figure 9
Example context schematic

Candidate semantics:

- A double-line box denotes a context. The box is divided into top and bottom halves by a separator line with the upper half containing a smaller rectangle anchored to the top left inside corner of the box.

- A single-headed arrow denotes that the context attached to the tail of the arrow subsumes the context attached to the head of the arrow. That is, all constraints that hold in the context attached to the tail of the arrow also hold in the context attached to the head of the arrow.

Discussion: The example (Figure 9) illustrates five distinguished contexts (V, W, X, Y, and Z). Each context is uniquely distinguished by a context identifier and provided with a descriptive label. A more detailed description of the context would be included in an elaboration form supporting the schematic. Although the central focus of this schematic is the context, graphical depiction of the anatomy of the context (i.e., the facts, objects, and relations that collectively define the context) was found to be unnecessary and possibly burdensome. The primary purpose for the schematic is to organize constraints in terms of the situations in which they hold. Thus, the only information deemed necessary to display about the context was that needed to distinguish one context from another. As indicated by the example above, constraints C1 and C2 are listed in the context box identified as X. The arrow leading from context X to context Y is to show that context Y is a subcontext of context X. That is, all that was true in X is also true in Y plus possibly more constraints. Contexts can be merged to form new contexts as is shown by context W, a combination of context X and context Z. Note that this process is strictly additive. That is, new constraints are always added to the contexts as you move down the schematic. You can not remove constraints as you move down the schematic.

Issues:

- Is there one way of expressing the label for a context that is somehow better than others? Adverbial phrases (e.g., [being] at the construction site, working on government contracts?) "When...?"

- When the list of constraint identifiers grows long, is this an indication that one needs to further refine the context and thus break things up into more manageable pieces?

- What pieces of information do we want to record (and/or display) about the contexts and their relationships? What are the types of relationships that we can expect between the concepts? What are the ones that we want to display in the diagram?

Constraint Resource Schematic

Purpose: The purpose of this schematic is to display (1) the system(s) or object(s) that maintain or enforce the constraint.

Viewpoint: Constraint-centered

Procedure component(s) supported:

- Mode Two, Collect and analyze evidence

- Mode Three, Hypothesize candidate constraints

- Mode Four, Validate and refine constraints

Graphical metaphor: Hierarchical

Candidate syntax:

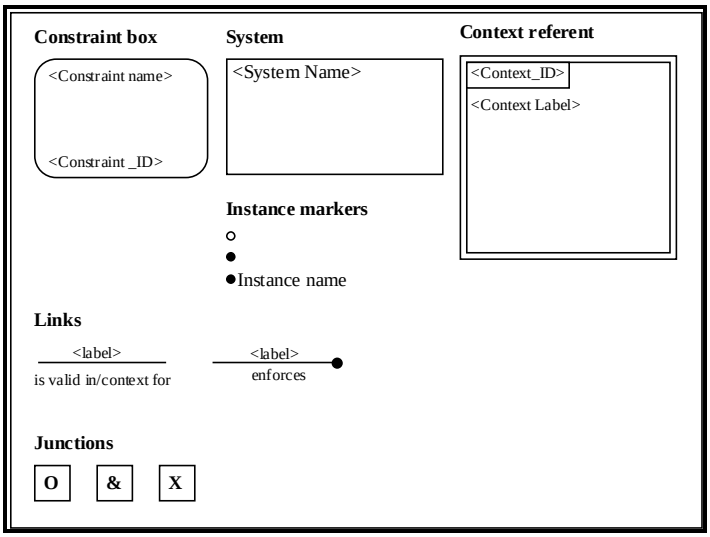


Figure 10
Candidate syntax for the constraint resource schematic

Example: Constraint C15: The Board of Directors is responsible for representing the shareholders in ensuring that the allocation of end-of-year profits toward stock value and dividends maintain an acceptable level of Return on Investment (ROI). Acting on a majority vote of the Board of Directors, the Comptroller invests profits and/or distributes dividends to stockholders.

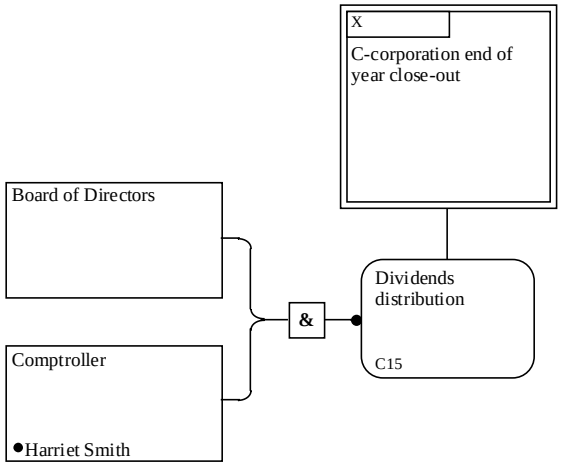


Figure 11
Example constraint resource schematic

Candidate semantics:

- A double-lined box denotes a reference to a context.
- A round-cornered box denotes a particular constraint whose unique identifier is located in the lower left-hand corner and whose label is in the upper left hand corner. The label is a meaningful name given to the constraint.
- A rectangle denotes a system (object or collection of objects). Instance markers may be included in the lower left-hand corner of the system rectangle. When no instance marker is present, the system rectangle represents the object kind indicated by the system label. An open instance marker indicates that *all* instances of the object kind are involved. A filled instance marker without a name indicates that *any* one instance of the object kind is involved. A filled instance marker with a name represents the *specific* named instance that is involved.
- As a convenience, the logic symbols could be omitted when the semantics is equivalent to an &-junction.

Note: All links have a label and should have an associated elaboration form that describes, for example, how a constraint is enforced by a system, object, or instance.

Constraint-relationship schematic

Purpose: The purpose of this diagram is to display existential dependency, part-of, and user-defined relationships among constraints.

Viewpoint: Relationship-centered

Procedure component(s) supported:

- Mode Two, Collect and analyze evidence
- Mode Four, Validate and refine constraints

Graphical metaphor: Hierarchical

Candidate syntax:

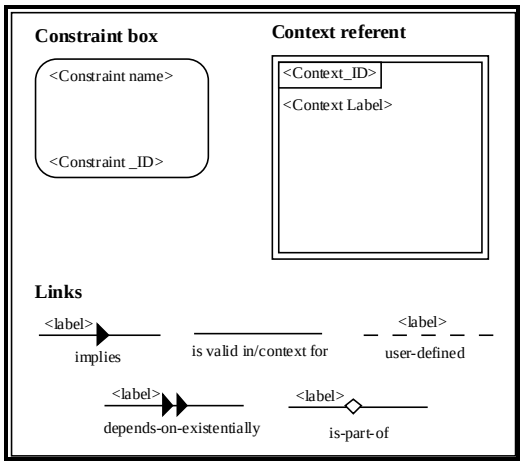


Figure 12
Candidate syntax for the constraint relationship schematic

Example:

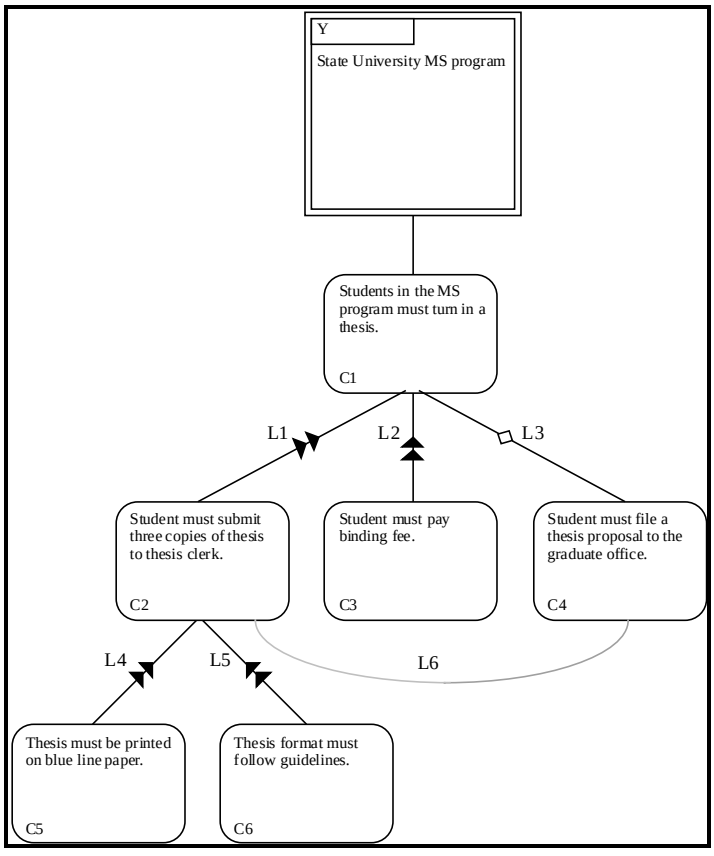


Figure 13
Example constraint relationship schematic

Candidate semantics:

- A double-lined box denotes a reference to a context.

- A round-cornered box denotes a particular constraint whose unique identifier is located in the lower left-hand corner and whose label is in the upper left hand corner. The label is a meaningful name given to the constraint.

Discussion: The idea for the diagram is to use the high level relation (depends-on) as the main focus. The analyst can specialize these relations using different symbols. Finally, other relations (user-defined) are marked with another symbol.

Issues: All links should have labels and elaboration forms associated with them. The elaboration forms define the nature of the relationship in detail. An important characterization of the relation may be the essential/non-essential property.

Constraint Effects Schematic

Purpose: To display the object(s) and system(s) affected by a constraint.

Viewpoint: Constraint-centered

Procedure component(s) supported:

- Mode Two, Collect and analyze evidence

- Mode Three, Hypothesize candidate constraints

- Mode Four, Validate and refine constraints

Graphical metaphor: Spider web

Candidate syntax:

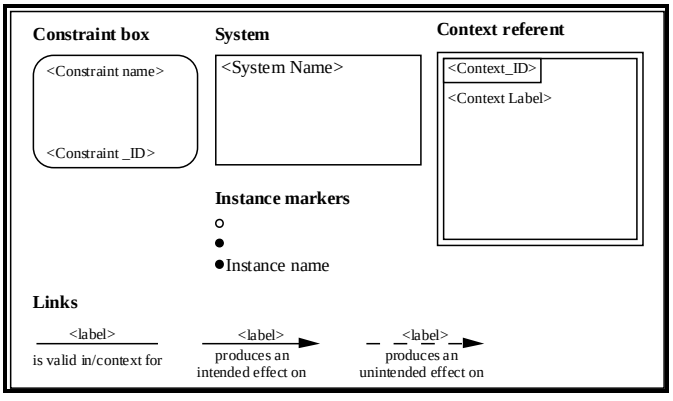
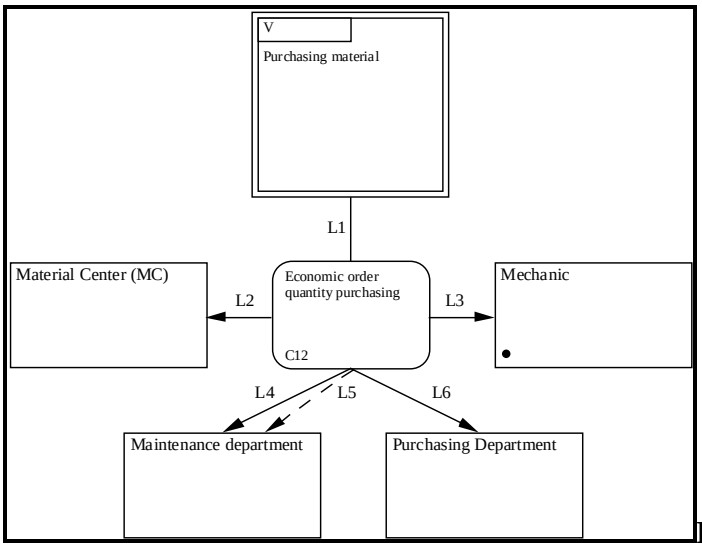


Figure 14
Candidate syntax for the constraint effects schematic

Example:



FigFigure 15

Example constraint effects schematic

Candidate semantics:

- A double-lined box denotes a reference to a context.
- A round-cornered box denotes a particular constraint whose unique identifier is located in the lower left-hand corner and whose label is in the upper left hand corner. The label is a meaningful name given to the constraint.
- A rectangle denotes a system (object or collection of objects). Instance markers may be included in the lower left-hand corner of the system rectangle. When no instance marker is present, the system rectangle represents the object kind indicated by the system label. An open instance marker indicates that *all* instances of the object kind are involved. A filled instance marker without a name indicates that *any* one instance of the object kind is involved. A filled instance marker with a name represents the *specific* named instance that is involved.

Goal schematic

Purpose: The purpose of this diagram is to show the relationship between constraints and individual goals. It may also be used to display the relative impact of each constraint with respect to the goal of interest and in particular the "prioritization" of constraints with respect to the system's goals. The constraints are ordered from top to bottom in "priority" (the highest priority being at the top). The constraints having a negative effect on the goal are displayed on the left of the goal, while the constraints having a positive effect on the goal are displayed on the right.

Viewpoint: Goal-centered

Procedure component(s) supported:

- Mode Two, Collect and analyze evidence
- Mode Three, Hypothesize candidate constraints
- Mode Four, Validate and refine constraints

Graphical metaphor: Spider/Hierarchical

Candidate syntax:

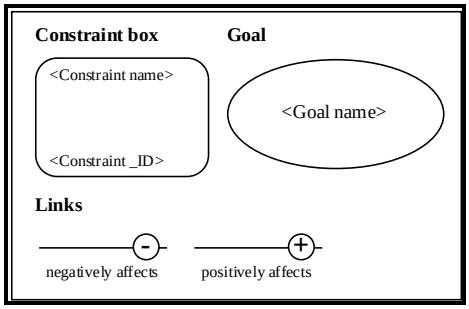
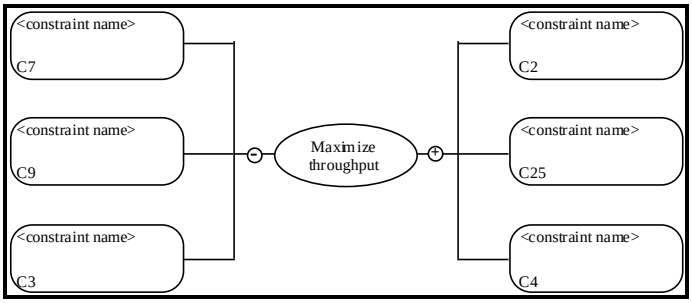


Figure 16
Candidate syntax for the goal schematic

Example:



or

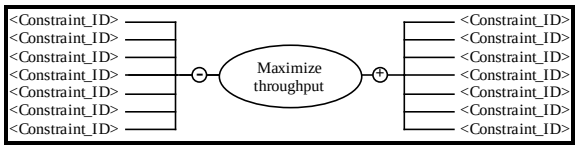


Figure 17
Example goal schematic

Candidate semantics:

- An oval denotes a goal where the goal is identified by the goal name.

- A rectangle denotes a system (object or collection of objects). Instance markers may be included in the lower left-hand corner of the system rectangle. When no instance marker is present, the system rectangle represents the object kind indicated by the system label. An open instance marker indicates that *all* instances of the object kind are involved. A filled instance marker without a name indicates that *any* one instance of the object kind is involved. A filled instance marker with a name represents the *specific* named instance that is involved.

- For any given pair of constraints on a link, the constraint positioned above the other has a stronger affect.

Discussion: An alternative mechanism for correlating constraints with goals is through the use of a matrix-based metaphor, such as used in Quality Function Deployment diagrams. Both strong and weak positive and negative correlations could be displayed between all goals and all constraints rather than centered on a single goal, as with the schematic above.

Goal relationship schematic

Purpose: The purpose of this diagram is to show the relationships (subsumption, conflicts, etc.) between goals.

Viewpoint: Goal-centered

Procedure component(s) supported:

- Mode Two, Collect and analyze evidence

- Mode Four, Validate and refine constraints

Graphical metaphor: Hierarchical

Candidate syntax: To be developed.

Example: None.

Candidate semantics: Not applicable.

Discussion: A number of relationships between goals can be considered. Among these are *is valid in/context for*, *depends-on-existentially*, *implies*, *is-part-of*, and so forth. The concept of a performance measure may also be needed in the schematic (perhaps as a list of metric names in the goal object).

Issues: To be determined.

Symptom schematic

Purpose: Assist with tracing the underlying cause of symptoms to the lack of an enabling constraint or to the existence of a limiting constraint.

Viewpoint: Symptom-centered

Procedure component(s) supported:

- Mode Two, Collect and analyze evidence
- Mode Three, Hypothesize candidate constraints
- Mode Four, Validate and refine constraints

Graphical metaphor: Spider

Candidate syntax:

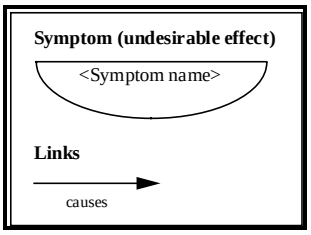


Figure 18
Candidate syntax for the symptom schematic

Example:

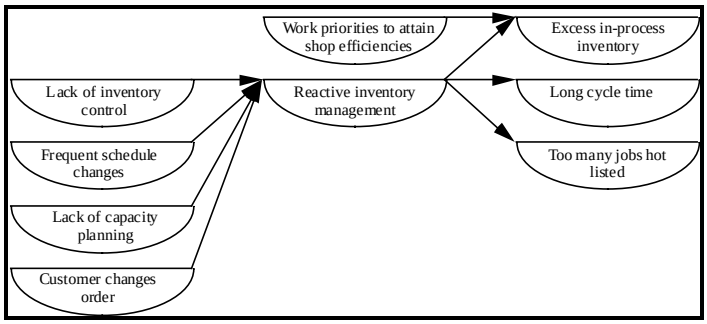


Figure 19
Example symptom schematic

Candidate semantics:

- A half-oval denotes a symptom or negative effect that is distinguished by its label.

- An "inclusive or" condition is implied at any point where arrow heads or tails converge. In other words, either one, a combination, or all of the symptoms stand in the relation "causes/is caused by" when multiple arrows originate from or terminate at a symptom symbol.

Discussion: A symptom is defined as an indication or sign (e.g., change in normal function, sensation, or appearance) indicating a problem. A problem is defined as a source of trouble or annoyance. To one domain expert, another's problem is the cause of another problem. To another individual, the same problem will be recognized as merely a symptom of another problem. Thus, the way negative effects are described (i.e., as the cause, problem, or effect) depends on the viewpoint taken. The actual root cause(s) of negative effects (or positive effects) can be very elusive. Symptoms and causality relationships among symptoms, on the other hand, are relatively easy for domain experts to recognize. For this reason, the schematic above has been named the *symptom schematic*.

Issues:

- Given an *inclusive-or* semantics on the relation *causes/is caused by* in the above schematic, additional information will need to be captured on a supporting elaboration form. This would help make the relations explicit, provided that information was available. An alternative might be to explore the introduction of new graphical symbols to display additional logic on the schematic.

- It may be useful to provide a mechanism to classify sets of like symptoms.

- An alternative schematic that merits further investigation is the Ishikawa diagram, also called a fishbone or cause-effect diagram, to assist in discovering the relationship between undesirable effects. Ishikawa diagrams are single problem-centered (i.e., one problem, symptom, or effect is isolated in the diagram as the center of focus). The central problem displayed on Ishikawa diagrams is represented by a "backbone" arrow from which one or more branches and sub-branches are created to illustrate cause-effect chains terminating at the backbone. The schematic above allows for either single effect-centered analysis and data collection or multi-effect analysis.

SIGNIFICANCE OF THE EFFORT

Among the most significant contributions of the Armstrong Laboratory's IICE program is a characterization of the nature and role of constraints in governing and predicting behavior among objects and agents in a system. This contribution is underscored by the observation that most domain experts need to "unlearn" an association between *constraint* and something negative. The majority of constraints, in fact, are necessary and enabling. The IDEF9 method helps to promote awareness of both the enabling and limiting aspects of constraints in an organizational context.

Building upon this foundation, the need for a method promoting a systematic approach to business constraint discovery was established. The work that followed has produced a prototype procedure and graphical language that is uniquely suited to the task. Together, these components establish an explicit process for recognizing, collecting, documenting, and validating business constraints.

Among the central features of the procedure is the provision for identifying *type problems* typical of business systems (See Table 1). The procedure itself was also designed to help avoid analogous problems in IDEF9 method application. A unique validation procedure was developed wherein analysts first hypothesize candidate constraints and then attempt to substantiate them by generating or collecting supporting data. This is followed by a step wherein domain experts challenge the conclusions of the analyst. The challenge step helps to ensure that candidate constraints promoted to the status of constraint arrive there with the full participation and consensus of domain experts.

Explorations into alternative language designs uncovered a number of promising schematics, each with a unique focus supporting both constraint discovery and downstream reuse of constraint information. One of the most interesting of these is the Context Schematic. The overall simplicity of this schematic in representing the context in which a constraint holds underscores the observation that a language is best judged not by what can be expressed but by what is explicitly not expressed. The structure of a context in terms of the objects, facts, and relations comprising the context could have easily overwhelmed the design of this schematic. In its design, the context schematic permits successive levels of detailing that in turn allow the analyst or domain expert to describe the context at an appropriate level of granularity. This design also permits focus on the relationship between the constraints and the context(s) in which it holds rather than placing an inordinate emphasis on the context itself. Additional schematics provide visualizations for other relationships of interest. For example, the Constraint Effects Schematic illustrates the relationship between constraints and affected objects in terms of intended and unintended effects. Existential dependency relations among constraints are displayed in the Constraint Relationship Schematic permitting rapid identification of outdated or unnecessary constraints. The Goal Schematic provides a mechanism to establish the relationship between constraints and organization goals and for prioritizing their relative impact. The Symptom Schematic assists in identifying cause-effect chains among problematic symptoms in the organization that can ultimately lead to the discovery of constraints among objects and agents within the system.

Collectively, these developments provide a sound foundation for future development. Additional development, testing, and refinement will be needed, however. The following section outlines some potential areas for future development, both within the IDEF9 method and in terms of spin-off work that can provide additional capabilities to leverage constraint information.

POTENTIAL AREAS FOR FUTURE WORK

A number of promising areas for additional work were identified during the development of the IDEF9 method. Several of these are listed below together with a brief description of the benefits to be gained by pursuing further development along these lines.

1. Method refinement. A number of areas within the prototype method merit further development and testing. Among these are expansions to the techniques supporting the IDEF9 method procedure, further development of graphical language elements, the incorporation of elaboration forms and data collection sheets, the development of a computational language of expression (elaboration language) and so forth. Specific provisions were made to ensure ease of integration with other methods (particularly with the IDEF5 method), however, a more in-depth treatment of this aspect of IDEF9 merits consideration. Each of IDEF9's design elements also needs to be tested, both in a highly controlled setting and in a production environment where unanticipated misuse or other problems can be detected and resolved. A wide range of test situations is recommended to maximize the robustness of the method.

2. Application framework development. Methods are, by design, highly task or skill oriented. Their true value is demonstrated mainly through their application in the context of projects involving the use of many skills and tasks coordinated to satisfy some goal or objective. Constraint discovery is an implicit first step for strategic planning, BPR, TQM, and a number of similar improvement strategies. There is a need, however, to establish exactly how constraint discovery activity and the constraint information collected can be integrated into the process.

3. Argument validity checking. Most decision-making is based not on quantitative data (e.g., bottom-line cost) but on qualitative judgments. Arguments for or against some proposed decision establish a case for action, one of whose supporting elements may be a business case. The relative merits of one decision over another, however, is largely determined by the validity of the argument posed to support the decision. Arguments are themselves composed of a chain of premises and conclusions that eventually lead to the final conclusion. Further investigation could be made to provide the IDEF9 method with techniques specifically enabling decision-makers to assess the validity of a proposed argument.

4. Mechanisms to allocate costs for enforcing constraints. Activity Based Costing (ABC) has been successfully applied as a mechanism for allocating costs to organization functions and for determining the appropriate cost of doing business [Kaplan 88]. The basic mechanism underlying the ABC approach is to identify how resources are used by activities to accumulate costs. The activity costs are then traced to the products and services generated by the activities. In a similar fashion, it would be useful to explore using a similar approach to

allocate costs to constraints since they involve the use of resources in their maintenance or enforcement. Such mechanisms would provide additional decision support for determining when the cost of a constraint exceeds its value and when it merits additional emphasis.

5. Methods and tools to design constraints. The design or redesign of constraints is a potential research topic in which a number of promising areas can be explored. The purpose of such efforts would be to assist with change management through more effective, proactive constraint management. Methods and tools are needed to support the design of constraints such that intended effects are maximized and unintended effects are minimized, systems designed to enforce constraints are appropriately configured, constraints are "flagged" whenever changes in the environment precipitate the need to reevaluate the need for a constraint, and so on.

6. Constraint categorization schemes. The mental exercise of classifying constraints has been shown to assist both in constraint discovery and downstream reuse of constraint information. Further research is needed, however, to explore alternative constraint taxonomies and their use in analyzing candidate constraints, checking for appropriate coverage in constraint discovery effort, and in identifying opportunities for improvement. The product of such effort would be a set of constraint taxonomies and techniques for using the taxonomies to support varied analyses.

7. Tools to capture, display, and maintain constraint information. The success of any method depends heavily on the existence of automated tools. This fact has always been true and is likely to continue to be so in the forseeable future. Automated tools not only assist practitioners in the application of a method but provide a rapid and reliable means for sharing, storing, and reusing information.

8. Libraries of constraints. On-line repositories of business constraints made available through the information superhighway would provide business owners, strategic and tactical planners, and systems developers with a powerful mechanism for exploring new ways of doing business and for expanding into new areas of business. The goals of dual-use conversion, agility, and similar initiatives depend on the ability to expose the current constraints under which the business operates and the constraints under which world-class systems in a given industry sector operate. With this visibility, a clear path to leveraging new areas of opportunity can be established.

9. Tools and environments for constraint-driven change management. One of the key problems facing organizations today is not having the ability to understand the effect of local change on the global enterprise. Many times a small change in policy, procedure, or product in one area has adverse effects, not initially understood, in other areas. The resulting effect is then either having to 'undo' the initial change or to have the change ripple through the enterprise in a domino effect. Either of these effects results in an inefficient use of the systems and

human resources within the organization. In organizations today, significant effort is expended on managing the effects of change, not the management of change itself. To reverse this situation, there is a need for more visibility of constraining relationships giving rise to organization behaviors, predictive tools enabling assessment of the impacts of change, and design tools to assist in the development of the system of constraints needed to proactively manage change.

CONCLUSION

Effective change management is greatly facilitated through the discovery and documentation of business constraints. The change management process begins with identifying business constraints. The knowledge of what constraints exists and how those constraints interact is often at best incomplete, disjoint, distributed, or completely unknown. Yet business constraints initiate, enable, govern, and limit the behavior of objects and agents within the business to accomplish the goals or purposes of the business. The IDEF9 method was designed to assist in the discovery and analysis of these constraints.

Considerable progress has been made toward the development of a complete and mature IDEF9. In its current form, IDEF9 provides a systematic and reliable approach for business owners, strategic and tactical planners, systems developers, project leaders, and decision-makers to identify and document business constraints. These developments provide the foundation for future endeavors both to refine and mature the IDEF9 method and to more effectively leverage the products of IDEF9 application.

BIBLIOGRAPHY

Demmy, Steven W., and Petrini, Arthur B., "The Theory of Constraints: A New Weapon for Depot Maintenance Planning and Control", *Air Force Journal of Logistics*.

Goldratt, E. M. and Cox, J., The Goal, North River Press, Croton-On-Hudson, NY, 1986.

Goldratt, E. M. and Fox, Robert E., The Race, North River Press, Croton-On-Hudson, NY, 1986.

Goldratt, E. M., Theory of Constraints, North River Press, Croton-On-Hudson, NY, 1990.

Goldratt, E. M., The Haystack Syndrome, North River Press, Croton-On-Hudson, NY, 1991.

Goldratt, E. M., An Introduction to Theory of Constraints: The Production Approach--Workshop Description.

Goldratt, E. M., An Introduction to Theory of Constraints: The Production Approach--Two-Day Workshop Course Materials, 28 July 1992.

Coleman, S., *Information Engineering Practitioner's Guide Volume IV Business Area Analysis*, Pacific Information Management, Inc. Culver City, CA, 1991.

Coleman, S., *Corporate Information Management Process Improvement Methodology for DoD Functional Managers*, Unpublished report by D. Appleton Company, Inc.,1992.

Gross, M., Ervin, S., Anderson, J. and Fleisher, A. "Designing with constraints," Computability of Design, Y.E. Kalay, Ed. Wiley, New York 1987.

Martin, James, *An Information Systems Manifesto*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

Maher, M. L. "Synthesis and evaluation of preliminary designs," Artificial Intelligence in Design, J. S. Gero, Ed. Springer-Verlag, New York 1989.

Mayer, R. J., et al. Knowledge-based integrated information systems development methodologies plan (Vol. 2), (DTIC-A195851).

Olle, T. William, Hagelstein, Jacques, Macdonald, Ian G., Rolland, Colette, Sol, Henk G., Assche, Frans J.M., Verrijn-Stuart, Alexander A., *Information Systems Methodologies: A Framework for Understanding*, Addison-Wesley Publishing company, Reading, MA, 1988.

Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, New York, 1987.

Putnam, Bill, *Emergency Base and MOB Interaction Model.*

Ramey, Timothy L., *Guidebook to System Development*, Hughes Aircraft Company, 1983.

Silver, Gerald A., and Silver, Myrna L., *Systems Analysis and Design*, Addison-Wesley, Reading, MA, 1989.

Smith, Gerald F., and Browne, Glenn J., "Conceptual Foundations of Design Problem Solving", IEEE Transactions on Systems, Man, and Cybernatics, Vol .23, No. 5, September/October 1993.

Smith, Gerald F., "Defining Real World Problems: A Conceptual Language", IEEE Transactions on Systems, Man, and Cybernatics, Vol. 23, No. 5, September/October 1993.

Wallace, Robert H., Stockenberg, John E., and Charette, Robert N., *A Unified Methodology for Developing Systems*, McGraw-Hill, Inc., New York, 1987.

Zachman, J., "A framework for information systems architecture", IBM Systems Journal, Vol. 26, No. 3, 276-292, 1987.

GLOSSARY OF TERMS

Constraint	A relationship that is maintained or enforced in a given context.
Context	A distinguished set of conditions.
Project scope	A delineated set of boundaries for constraint discovery effort documented in the form of scope statements.
Project purpose	The aim, object, or desired ends for the constraint discovery effort. The purpose is usually established by 1) prioritized objective statements for the effort, 2) statements of needs that the constraint discovery effort must satisfy, and 3) questions or findings that the client wants answered.
Evidence	An indication, sign, or manifestation that supports or proves the existance of a constraint in a given context.
Relation, relationship	An abstract, general association or connection that holds between two or more conceptual or physical objects.
Effect(s) of a constraint	Something that inevitably follows an antecedent (as a cause or agent). Effects are classified as direct or indirect, intended or unintended, and desirable or undesirable.
Symptom	Something characteristic or indicitave of a condition impairing normal functioning.
Process	An ordered sequence of events. In human-designed systems, the events that constitute a process are designed and ordered to achieve some desired outcome. A business process, in particular, is an ordered sequence of events involving people, materials, energy, and equipment, that is designed to achieve a defined business outcome
System	A *system* is defined as a collection of objects standing in particular relations and exhibiting particular behavior prescribed by a collection of constraints.
System, business	A collection of objects behaving to perform one or more business functions under the influence of constraints to accomplish a particular goal.

Goal	An object or end that the business or system strives to achieve.
Constraint, rationale of the	The set of beliefs motivating the establishment and maintenance of a constraining relationship.
Constraint statement	A textual description of the constraining relationship.