

System Architecture of Offshore Oil Production Systems

by

James Keller

B.S. Civil Engineering
Kansas State University, 2002

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[June 2008]
MAY 2008


© 2008 Massachusetts Institute of Technology. All rights reserved.

The author hereby grants to MIT permission to reproduce
and to distribute publicly paper and electronic
copies of this thesis document in whole or in part.

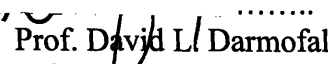
Signature of Author.....

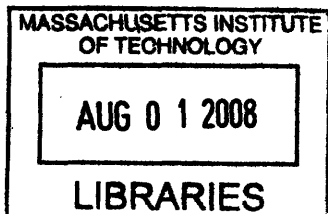

James Keller
Department of Aeronautics and Astronautics
May 2008

Certified by.....


Ed Crawley
Professor
Aeronautics and Astronautics
Thesis Supervisor

Accepted by.....


Prof. David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students



ARCHIVES

System Architecture of Offshore Oil Production Systems

by
James Keller

Submitted to the Department of Aeronautics and Astronautics
on May 16, 2008 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Aeronautics and Astronautics
At the Massachusetts Institute of Technology

ABSTRACT

This thesis presents an approach to applying Systems Architecture methods to the development of large, complex, commercial systems, particularly offshore oil and gas production systems. The aim of this research was to assist BP in the development of concepts for a multi-billion dollar oil production system, particularly in the unprecedented deep water arctic locations prone to seismic activity, as well as in existing fields that must be extended. The thesis demonstrates that these systems can be decomposed and analyzed using rigorous, methodical system architecture thinking that archives and represents tacit knowledge in several graphical frameworks. The thesis breaks the architecture of oil and gas production systems into two problems. The first problem is the architecture of one facility and one reservoir; a classic problem of assigning function to form. The second problem is the architecture of multiple facilities and multiple reservoirs; a classic problem of connection and routing. For the first problem, the production process is decomposed using Object Process Methodology (OPM). The decompositions provide a methodology to capture industry knowledge that is not always explicitly stated and provides a framework to explore the entire architectural design space. The thesis then describes how these decompositions of general and specific oil systems can be used to develop software models, using the meta-language tool OPN (Object Process Network), that successfully generate thousands of architecture concepts. This set of feasible architectures can be prioritized and better understood using metrics in an effort to down-select to a handful of preferred concepts to be carried forward for more detailed study and eventual development. The approach to the second problem demonstrates that even a modest set of facilities and reservoirs have a huge number of connection possibilities. This space of connection possibilities is large and daunting, and typically is not fully explored. To solve the second problem the thesis presents two models that generate all the possible connection schemes between elements in a system, in this case oil facilities and reservoirs. It is then demonstrated that these possibilities can be prioritized through the use of metrics. The thesis presents a method that can identify new concepts, highlight preferred sets of concepts, and underline patterns common to those concepts. This method increases the architects' overall knowledge and understanding of the entire space of possibilities, and ensures that all options are considered in the development of complex systems.

Thesis Supervisor: Ed Crawley

Title: Professor, Aeronautics and Astronautics

ACKNOWLEDGEMENTS

My thesis and work at MIT would not have been possible without the support and contribution of many individuals.

My thesis advisor, Professor Ed Crawley, took a chance by hiring a civil engineer from Kansas, for which I'm extremely grateful. Ed's guidance and wisdom were the driving force for my transition to become a systems engineer. I always particularly enjoyed Ed's digressions, which always amazed me with their breadth of subject matter and frequency of occurrence. Ed sets a high bar for striving to have a well-rounded mind.

I would also like to thank my research partners, Ziv Rozenblum and Wen Feng. They made the journey very enjoyable and I learned a great deal from them--some of it even had to do with systems engineering.

The cooperation and patience shown by our sponsors, Bob Robinson, Alastair Barr, and Tomas Flanagan from BP, for 4 guys as naive about the oil industry as we were, is remarkable. It was a pleasure working with them. They were also great teachers.

Two other MIT faculty members, who contributed in a large way to my success at MIT with their advice and concern, were Professor Oli de Weck and Col. John Keesee.

The classes and the research wouldn't have been much fun had it not been for my lab mates. I'd like to thank Ryan Odegard, Phillip Cunio, Zahra Khan, Tim Sutherland, Bruce Cameron, and Theo Seher for injecting every day with spectacular (and sometimes odd ball) discussions and interactions. I would also like to thank the three students I think of as my PHD student mentors Wilfried Hofstetter, Ryan Boas, and Bill Simmons; thanks for the advice and mentorship.

I'd like to thank my good friends here at MIT: Andy, Bastien, Adeline, Michelle, and Jim, who have made the past two years here in Boston a blast.

Finally, I'd like to thank my family for their support and encouragement (and the visits). In particular, I thank my uncle, Rick Fornelli, for encouraging me to apply to MIT and believing I could make it; my sister, Katie, for making my application and resume bleed on so many occasions in order to make it to MIT; and my brother, Grant, for talking to me on so many trips home from campus and making me feel I was still connected to K-State. All of this would certainly not be possible without the support (including financially) and love from my wife (and sugar-momma) Melissa. This thesis is as much her doing as mine. And last but not least, I want to thank my parents and in-laws, Tom and Lea and Mac and Sharon. Mom and Dad, thanks for instilling in me compassion, curiosity, a thirst for knowledge, and a strong work ethic - they help me every day.

TABLE OF CONTENTS

CHAPTER 1. Introduction	11
CHAPTER 2. Architecture Model of Oil and Gas Production Process	15
2.1 Introduction.....	15
2.2 Decomposing the Oil and Gas Production Process.....	16
2.2.1 First Level Decomposition of the Oil and Gas Production Process.....	17
2.2.2 Second Level Decomposition of the Oil and Gas Production Process.....	19
2.2.3 Third Level Decomposition of the Oil and Gas Production Process	21
2.2.4 Fourth and Fifth Level Decomposition of the Oil and Gas Production Process	23
2.2.5 Describing Location.....	24
2.3 Developing the Morphological Matrix for the Oil and Gas Production Process.....	26
2.4 Constraints within the morphological matrix	30
2.5 Metrics and Outcomes.....	31
2.5.1 Possible Metrics	32
2.6 Implementation of the Model.....	35
2.7 Alternate Specific Oil and Gas Production System Model	39
2.7.1 TRL Metric and Outcomes	43
2.8 Summary	48
CHAPTER 3. Multi-Reservoir Multi-Facility Connection Generator.....	50
3.1 Introduction.....	50
3.2 Motivation	50
3.2.1 The Problem.....	50
3.2.2 Requirements.....	51
3.2.3 Possible Solution Concepts	52
3.3 The Solution	56
3.3.1 One Block without Constraints	56
3.3.2 Multiple Blocks.....	59
3.3.3 The Addition of Constraints	60
3.4 Output	64
3.4.1 Exporting to Excel	65
3.5 Expansion Possibilities.....	66
3.5.1 Other Potential Constraints	66
3.6 Use of Metrics with the Generator	67
3.7 Summary	70
CHAPTER 4. Expanded Multi-Reservoir, Multi-Facility Connection Generator (N plus M Connection Generator).....	71
4.1 Introduction.....	71
4.2 Motivation	71
4.2.1 The Problem.....	71
4.3 The Concept	73
4.4 The Implementation.....	75
4.4.1 Each Block	75

4.4.2	Blocks in Series.....	76
4.4.3	Other Elements in the Model	77
4.4.4	Combinations of Combinations	78
4.4.5	Constraints	78
4.4.6	Other Potential Constraint.....	82
4.4.7	Output.....	83
4.5	Expansion Possibilities.....	84
4.6	Limitations.....	85
4.7	Summary	85
CHAPTER 5. Conclusion and Future Work		87
5.1	Conclusions.....	87
5.2	Recommended Future Work.....	89
APPENDIX 1. OPM Symbols		92
APPENDIX 2. Oil Decomposition Levels.....		93
APPENDIX 3. Constraints and Explanations		99
APPENDIX 4. Abbreviations and encoding used in Oil OPN		102
APPENDIX 5. TRL Scale (From sponsor presentation).....		103
APPENDIX 6. Data for the Alternate Specific Architecture Model.....		104
APPENDIX 7. Operation of the N by M connection generator model		106
APPENDIX 8. Global Script code for N plus M generator		110
APPENDIX 9. Operation of the N Plus M connection generator model.....		112
BIBLIOGRAPHY		117

LIST OF FIGURES

Figure 2-1. Definitions for Oil and Gas Production Process Decomposition	17
Figure 2-2. Level 1 Decomposition of Oil and Gas Production Process.....	19
Figure 2-3. Level 2 Decomposition of the Oil and Gas Production Process	21
Figure 2-4. Level 3 Decomposition of the Oil and Gas Production Process	22
Figure 2-5. One of the 11 Elements of the Level 4 and 5 Decomposition of the Oil and Gas Production Process (for the others see Appendix 2).....	24
Figure 2-6. Locations for a Concept	26
Figure 2-7. Attributes for Locations.....	26
Figure 2-8. Example of Decisions in Decomposition.....	27
Figure 2-9. Hierarchical Morphological Matrix of the Oil and Gas Production Process.....	29
Figure 2-10. Table of the Constraints in the Oil and Gas Production Process Model	31
Figure 2-11. Example Plot of Metrics for Oil and Gas Production Process Architectures.....	34
Figure 2-12. OPN Model Implementation Layout.....	36
Figure 2-13. Screenshot of Oil and Gas Production Process OPN Model.....	38
Figure 2-14. Decision Path for Alternate Specific Architecture Model (Full View)	41
Figure 2-15. Decision Path for Alternate Specific Architecture Model (Part 1 of 3)	42
Figure 2-16. Decision Path for Alternate Specific Architecture Model (Part 2 of 3)	42
Figure 2-17. Decision Path for Alternate Specific Architecture Model (Part 3 of 3)	43
Figure 2-18. Table of Concept Families for the Alternate Specific Model.....	45
Figure 2-19. TRL Penalty (Method 1) vs Cost (1700m)	45
Figure 2-20. TRL Penalty (Method 1) vs Cost (2500m)	46
Figure 2-21. TRL Penalty (Method 2) vs Cost (1700m)	46
Figure 2-22. TRL Penalty (Method 2) vs Cost (2500m)	47
Figure 2-23. Table of Preferred Concepts from TRL Model.....	48
Figure 3-1. Example Connection Matrix	51
Figure 3-2. Example Connection Illustration	51
Figure 3-3. Connection Matrix Output Example.....	52
Figure 3-4. Looping-Indexing Concept.....	53
Figure 3-5. Example Block.....	54
Figure 3-6. Blocked-Cascading Concept.....	55
Figure 3-7. Constraint Entry Code in Global Script	61
Figure 3-8. Screenshot of the Actual OPN N by M Connection Generator	64
Figure 3-9. Example Output Spreadsheet.....	65

Figure 3-10. Complete Output for an Example 3-by-2 Connection Matrix	66
Figure 3-11. Example 3X4 Connection Layout	67
Figure 3-12. Metrics Plot of 2401 Connection Schemes for 3X4 Example	69
Figure 3-13. Metrics Plot of 81 Connection Schemes for 3X4 Example	69
Figure 3-14. Four Connection Schemes for 3X4 Example	70
Figure 4-1. Example N+M Connection Matrix	72
Figure 4-2. Example N+M Connection Illustration.....	73
Figure 4-3. N+M Model Concept	74
Figure 4-4. Screenshot of OPN N+M Connection Model.....	77
Figure 4-5. Connections for a Single Element.....	80
Figure 4-6. N+M Model Excel Front-end.....	82
Figure 4-7. Example N+M Connection Matrix Output.....	84

CHAPTER 1. Introduction

The successful development of complex systems is a difficult problem and it's becoming more difficult: as technology advances, systems' complexity increases. The increasing complexity, coupled with the fact that our largest and most complex systems are decades-long endeavors, requiring many billions of dollars in resources, make it imperative that the development of these systems is performed in the most efficient way possible. The earliest development stages of complex systems are an area where considerable planning must be done to ensure selection of the best possible system. In the beginning stages of the development of a system the designer has the greatest leverage on the final performance and cost of the system. The further the system architect moves into the system's development, the more decisions are made which lock in costs and performance measures that will influence the lifecycle of the system.

The most important task in the early stages of development is selecting the concept for the system. Traditionally this is done by a committee of engineers who collectively draw on their experience to generate a handful of feasible concepts, which, through further study, are down-selected to the specific concept that will be developed. However, given the enormous complexity of modern systems, can the system architect ever be certain all options were considered? Furthermore, given the enormous quantity of resources and time these systems take to develop, isn't it important that every effort be made to ensure all options are explored before selecting a system's concept? This thesis will address these questions.

The objective of this thesis is to show that systems architecture methods and thinking can successfully be applied to the development of large, complex, commercial systems, in particular to offshore oil and gas production systems. Successful application means that these large commercial systems can be decomposed and analyzed using rigorous, methodical system architecture thinking that explicitly archives and represents tacit knowledge in several graphical frameworks. Successful application also implies that these system architecture methods give interesting and useful results when applied to complex systems. Further success is demonstrated if the use of these methods increases the productivity of the engineers working on the development of these systems.

The sponsor of this research, BP, was dealing with some significantly difficult systems architecting problems. They were interested in developing offshore oil and gas production systems in arctic locations with water depths in excess of 100m and in regions prone to heavy seismic activity. The development of fixed or floating platforms that could handle crushing ice

loads or heavy seismic events in such deep water has never been achieved. BP was also faced with the challenges of extending existing fields by connecting existing facilities to potentially many untapped reservoirs. However, they weren't able to explore all the connection scheme possibilities because the number of possibilities was just too great using current practices. To deal with these complex problems BP has many systems engineering tools and practices that enable them to analyze concepts and architectures with enormous depth and repetition.(1) In particular they use concept catalogs as a way to document all the possible components that might be used in an architectural concept. These catalogs include information that details which situations each component is most applicable, as well as, specification for preliminary concept building. These catalogs are produced for various regions and functions, such as Gulf of Mexico production systems, production systems in arctic environments, and sub-sea production elements. For a new project, they generate about a dozen concepts using graphic building blocks and then analyze them in great detail. This analysis is enabled, in-part, by powerful cost and performance software like Oil and Gas Manager (OGM). The analysis allows them to weed out the infeasible and poorer performing concepts. At the same time, they adjust the surviving concepts based on the knowledge gained about the problem through the analysis. This continues until they reach the final concept which goes on for development.

This research is based on a system modeling approach that describes a system through two basic elements, functions and forms. The idea of thinking of two basic system types was first proposed by Harel and Jacobson.(2)(3) This idea developed into two lines of thinking. One was that systems could be represented by processes in data flow diagrams (DFD), developed by De Marco(4). The other line of thinking was that systems could be represented as a set of objects. The object-oriented thinking was widely used and expanded [Rumbaugh et al., Booch, Coad and Yourdon, Embley et al., and Shlaer and Mellor(5)(6)(7)(8)(9)] in development of complex software systems. The idea of using both types of elements, objects and processes, when representing any system was beginning to be seen in "action" DFD (ADFD), developed by Shlaer and Mellor(9). However, describing the functions and forms of all systems using processes and objects fully came together in the approach called Object-Process Methodology (OPM), developed by Professor Dov Dori.(10) The OPM approach uses a graphical representation called Object-Process Diagrams (OPD) to describe and archive the behavior and structure of systems. Crawley introduced the idea of rigorously representing forms as instrument objects within an OPM and functions as coupled operands and processes within OPM.(11) A meta-language, called OPN (Object-Process Network), developed by Professors Ed Crawley and Ben Koo, has been used to model and explore the architectural design space of complex systems.(12) This led to the recognition that the nature of the design space for architecture

problems could be represented in an Architecture Decision Graph (ADG) framework, developed by Simmons.(13) The ADG representation of a design space is made up of a set of interrelated decision variables. Up to this point, OPN and ADG's have been used to assist NASA in the development of space systems for exploration of the Moon and Mars.(14)(15) Prior to this thesis, these methods and tools had not been applied to commercial systems, or even outside the aerospace engineering field.

The specific objective of this thesis is to apply OPM thinking to the development of oil and gas production systems. This is to assist BP in the development of multi-year, multi-billion dollar projects around the world. In particular, this thesis aims to demonstrate that, using OPM and OPD's, the general oil and gas production process can be decomposed and analyzed. This should enable BP's engineers to explicitly archive the intricacies of the production process, so that innovative solutions can be developed. Once this is possible for the general process, this method should be demonstrated on more specific "real world" problems.

The thesis should also show that OPN can be used as a tool to transform decompositions of oil and gas production processes into a software engine that generates all the feasible architecture concepts for a given situation. Our method should also be shown to assist in the down-selection of concepts to a handful of preferred concepts using metrics including cost, schedule, and technology readiness. The preferred set of concepts can then be carried forward for more detailed study. The overall goal is to provide BP with systems engineering tools and procedures that will help them develop new and better production systems.

The problem of architecting very complex offshore oil and gas production systems will be broken into two parts. The first part is the problem of architecting an offshore oil and gas production system which consists of one facility producing from one oil reservoir. This will be approached as a classical system architecture problem of assigning function to form. The second part is architecting a field of multiple facilities and multiple reservoirs. This problem will use the solutions to the first problem for the individual facilities and reservoirs, but will approach the core of the second problem as another classical system architecture problem, that of connection and routing.

This thesis is organized into three main chapters. Chapter 2 addresses the first part of the problem, one facility and one reservoir. In particular, Chapter 2 discusses the use of our systems architecture methods on the general oil and gas production process. It starts by decomposing the process into a graphical representation of the functions and forms contained in the overall process. Then, searches that decomposition for decision points and then gathers and organizes them in a matrix framework. This chapter then goes on to discuss the

development of the OPN model that generates all the feasible concepts in that design space. It finishes with a discussion of the development of an alternate, more specific, oil and gas production system model, and a discussion of how the architectures it generates can be prioritized using metrics. The second part of the problem, multiple facilities and multiple reservoirs, is addressed by Chapters 3 and 4. Chapter 3 discusses the development of an OPN model that generates all the possible ways to connect a set of oil platforms and oil reservoirs, a general connection assignment problem. It also lays out the usefulness of applying this connection generator in other disciplines. Chapter 4 presents the development of an enhanced connection generator that captures a more general set of connections between different elements in a system. This thesis finishes by presenting its conclusions and recommendations for future work.

CHAPTER 2. Architecture Model of Oil and Gas Production Process

2.1 Introduction

The aim of this research was to assist BP in the development of concepts for a multi-billion dollar oil production system, particularly in the arctic. The intent was to build an OPN model that would explore the entire envelope of the design space for the oil and gas production process and generate many viable concepts. These concepts would have one or more metric values associated with each one, so that after the set of concepts was generated for a certain situation, one could focus in on the top handful of concepts for further study.

This idea of generating all the concepts in the design space comes from the following notion. In standard engineering practice today, during the concept development stage for a new system, especially for a new or unprecedented system, the first step is to gather a team of experienced engineers. They develop five to ten concepts for more detailed study and eventual down-selection. Because these engineers are human, they draw on their collective and individual past experience to develop these concepts. This method has served the engineering community well thus far, but it is possible that because we are developing concepts based on the past experiences of individuals, there may be innovative concepts that are missed in this sweep of the design space. This method of enumerating all the concepts in a design space takes advantage of current computing power combined with the experience and knowledge that an engineering community possesses. Enumerating all possible concepts with some metrics associated with each concept allows us to thoroughly sweep the whole design space and ensure the handful of concepts sent forward in the development process represent those preferred from the entire space.(15)

In this chapter the motivation for this work is discussed first. Next, there will be a discussion of the first steps of decomposing the oil and gas production process into its more basic objects and processes. This decomposition will be used to find the decision points with the architecture design space. The next sections discuss the development of the morphological matrix and constraints that would act as a framework and starting point for the making of an Object Process Network (OPN)(12)(15)(14)(16)(10) model that would generate all the possible architecture concepts. This is followed by a discussion of the metrics that would be used within the OPN model to evaluate and rank the generated architectures. The chapter concludes with a discussion of the implementation of the OPN model and the outcome from that model.

2.2 Decomposing the Oil and Gas Production Process

The first step for this research project was to develop an understanding of the oil exploration and production process. This was particularly important and necessary because the author knew very little about the oil business. This required a steep learning curve in order to grasp the oil and gas production process and the technical language of the oil industry with enough detail to decompose it. This initial lack of knowledge had one advantage in that it allowed us to look at the oil exploration and production process from a fresh perspective as we learned and decomposed the process. This was important, as it freed us from having preconceived connections between particular functions and their forms. It allowed us to think about generic, solution-neutral processes and forms. This solution-neutral thought process is important in allowing this method to possibly develop “out-of-the-box” concepts, as thinking about problems in an abstract way helps to open up more possibilities for solutions, and solution-neutral thinking allows for a high degree of abstraction in the description of a problem, as it deliberately avoids prescribing any particular solution.(11)

Our learning of the oil and gas production process was enabled by our readings(17)(18). But most of our understanding was developed through conversations with our BP sponsors, which occurred at least once per month. The following decomposition was completed by Ziv Rozenblum(19) and the author over the period of the first year of this project. We used Object Process Diagrams (OPD’s) which are graphs using Object Process Methodology(10)(11), to describe the decomposition of the system. A tutorial on OPN and OPD’s can be found in Appendix 1. Through the course of the work we had to develop a specific dictionary of terms to describe processes in the whole oil and gas production system. Where possible, an effort was made to keep these words solution-neutral in the sense that they did not have the specific connotation that accompanies similarly defined words in the oil industry’s technical language. Otherwise we kept the terms that were used by the oil industry for a particular process or object. The following table of definitions , Figure 2-1, will be useful in reading the coming decompositions.

Changing Composition	Altering the composition of the components of the reservoir fluid (e.g. separating)
Changing Properties	Altering the property attributes of something (e.g. temperature, pressure)
Constructing	Causing a form to come into being
Custody Transfer	Change ownership of an operand to another entity
Disposing	Getting rid of something
Disturbing	Something from outside the system causing a negative effect on the system's ability to perform its functions
Energizing	Any process which assists the transportation of the reservoir fluid within the process of extracting
Extracting	Creating a means to get the reservoir fluid to a wellhead and then moving it to the wellhead
Maintaining	Scheduled and unscheduled fixing, replacing, and repairing
Moving	Changing the position of something between facilities but not within a facility
Operating	The actions involved with causing all processes to run and/or to continue to run
Powering	Providing energy in a useful form
Provisioning	Providing the necessary materials, supplies and logistics
Reaching	The process within "Extracting" by which one gets at the reservoir fluids (i.e. drilling)
Reusing	Using a reservoir fluid to deliver value other than making a sale
Storing	Holding something stationary at a facility at which this is the only function (other storing taking place at a facility with other functions will be assumed to be taking place as required, but will not fall under this term)
Supporting	Mechanically supporting
Supporting Horizontal	Counteract forces acting at right angles to gravity
Supporting Vertical	Counteract gravity
Sweeping	Changing the properties of the fluid in the reservoir and/or the reservoir such that it moves with more ease to the well head (i.e. change viscosity, reduce attachment to rock particles...)
Transporting	The process taking place within "Extracting" that changes the position of the fluid from in the reservoir, to at the well head
Treating	Changing the fluid to make it more useful

Figure 2-1. Definitions for Oil and Gas Production Process Decomposition

2.2.1 First Level Decomposition of the Oil and Gas Production Process

At the highest level, we define the oil production process as, "taking the fluids from an underground reservoir and outputting sellable oil and gas to a purchaser." We decomposed the oil production process into four main processes: Extracting, Treating, Moving, and Storing. The exact definitions of these four processes can be seen in Figure 2-1. It is around these four main processes that we built our decomposition. We decomposed the oil and gas production

process using the hierarchy of levels. Level 1, for instance, is the highest level in the hierarchy. It represents the system as simply as possible and contains information about the context around the system. Subsequent levels zoom-in on the process, breaking the system down into component systems (objects) and component processes. Note that objects can be any physical things, including the oil itself, a platform, or other important items like storage tanks, ships, and pipelines. Processes act on objects, and may affect them, change them, or produce new objects. Mapping the basic abstract principles of object and process onto everyday language, an object can be almost any noun and a process can be almost any verb.

The OPD for oil and gas production process Level 1 can be seen in Figure 2-2. The main process, Producing, interacts with reservoir fluids and leads to the custody transfer of exported oil and gas. Producing is done by the main instrument object, the Producing Facility. Producing also takes place in context with the local ocean, land, ocean floor, and sub-sea geography. Producing is made of six sub-processes: Extracting, Treating, Moving, Storing, Disposing, and Reusing. The producing facility is affected by a process we call disturbing. Disturbing is done by meteorological and oceanic conditions (MetOcean), seismic activity, arctic ice, and gravity.

In our OPD's there are two features in addition to object and processes. The first is the box made from a dashed line. This represents the system boundary. That means that the architect of this system is responsible for designing the objects and processes within the box and dealing with the interface of anything that crosses the boundary. Additionally the architect must be knowledgeable about whatever lies just outside the boundary.

In these diagrams, arrows or lines that cross the boundary indicate that something from outside the system is passing or being passed some operand to/from the system or that something outside the boundary is effecting or being affected by the system. Any of these cases indicate to the architect that there is an interface that needs to be considered. There are also instances of processes that lie on the boundary. These are inherently processes that interface between the system and the wider world around the system.

The second feature in the OPD's is the color code. As you can see in the lower right hand corner of Figure 2-2 there is a legend for the coloring of the various elements in the OPD. Green elements are elements that will be included in the OPN model. Yellow elements may be included in the OPN model. Red elements will not be included in the OPN model, but are represented in the OPD's to provide more detail and generality. This color coding was based on two criteria. First, does that element have a major role in differentiating different concepts? An element that does have a major role may be involved as a major cost driver or it may involve

creating the major branches in the architecture's design space. The second criterion is whether or not the element contributes greatly to the complexity of the OPN model. The intent was to make the first model simple and then increase the complexity as the model was refined.

This modeling of the oil and gas production process had to be able to represent the various situations in which this system might be built. Each situation could have a different location, context, and/or environment. The set of feasible architecture concepts then would also be different for each situation. For example, you may want to build an oil and gas production system in the arctic or you may want to build it in the Gulf of Mexico. Each situation has a different location, context and environment. They also have a very different set of feasible architecture concepts. However, what is important and represented in this modeling of the oil and gas production process is that they both have the same oil and gas production process. They are both represented by this decompositional model.

In the decompositional model, these situations are represented by the different MetOcean, Seismic, Ice, and geography that lie outside the system boundary.

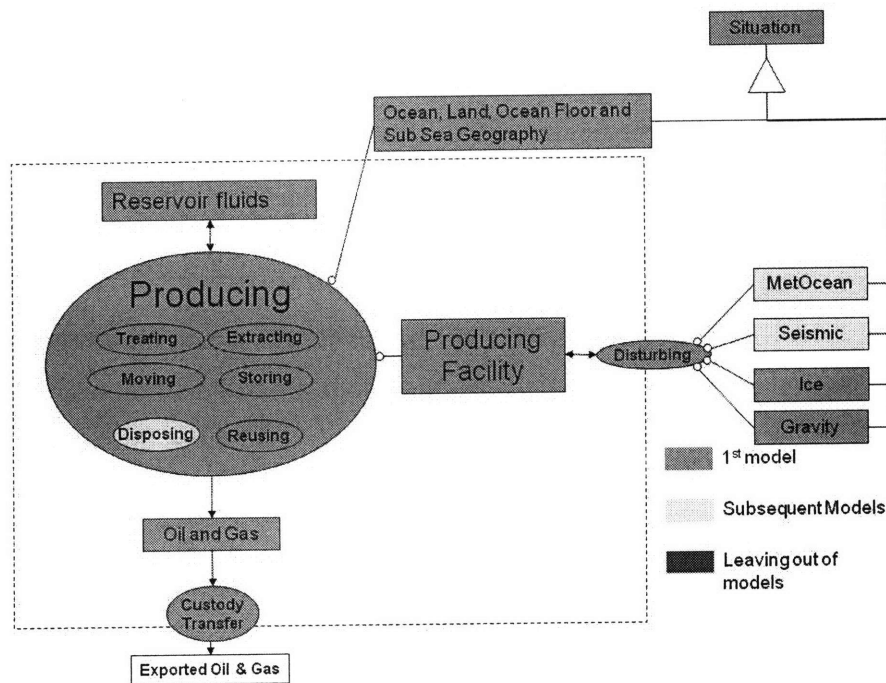


Figure 2-2. Level 1 Decomposition of Oil and Gas Production Process

2.2.2 Second Level Decomposition of the Oil and Gas Production Process

Figure 2-3 represents our understanding of the Level 2 decompositional model of the oil and gas production process. In this OPD the process called Production is expanded to its sub-

process. The instrument object (a special class of object, by means of which a process occurs) Production Facility is also expanded to its component objects.

The production process is best decomposed as four main sub-processes and two more minor sub-processes. The four main processes are Extracting, Treating, Storing, and Moving. In Figure 2-3 you can see that there is some notion of the relative timing of these processes. Extracting and Treating must occur. They must be the first processes to occur and they must be in that order. Extracting interacts with the reservoir fluids. Treating has an input of reservoir fluids and outputs the operands: oil, gas, water and other. Storing and Moving may or may not occur as processes in oil and gas production, and sometimes occur more than once. These two processes can also occur in various orders before and after one another. Moving can even occur before the treating process. Storing and Moving both interact with the operands oil and gas. Storing also potentially interacts with the other two operands. Each of these four processes is done by generic instrument objects: Extractors, Treaters, Depositories, and Movers. It is upon these objects that the process of disturbing interacts.

The two more minor sub-processes, Reusing and Disposing, each usually occur in the oil and gas production process. Both can have the operands of gas, water and other as their inputs. Reusing interacts with the reservoir fluids when operands are returned to the reservoir. The Disposing process lies on the system boundary because it takes its operands outside the defined system.

A process called Custody Transfer also sits on the boundary of the system. It has inputs of oil and gas. It can also be thought of as the interface between the system and the outside, at which value is delivered by the system. Because the geographical location and physical manner in which this process takes place both have large variance, it was quite difficult to arrive at the current representation. Further discussion on this issue can be found in a thesis by Ziv Rozenblum(19).

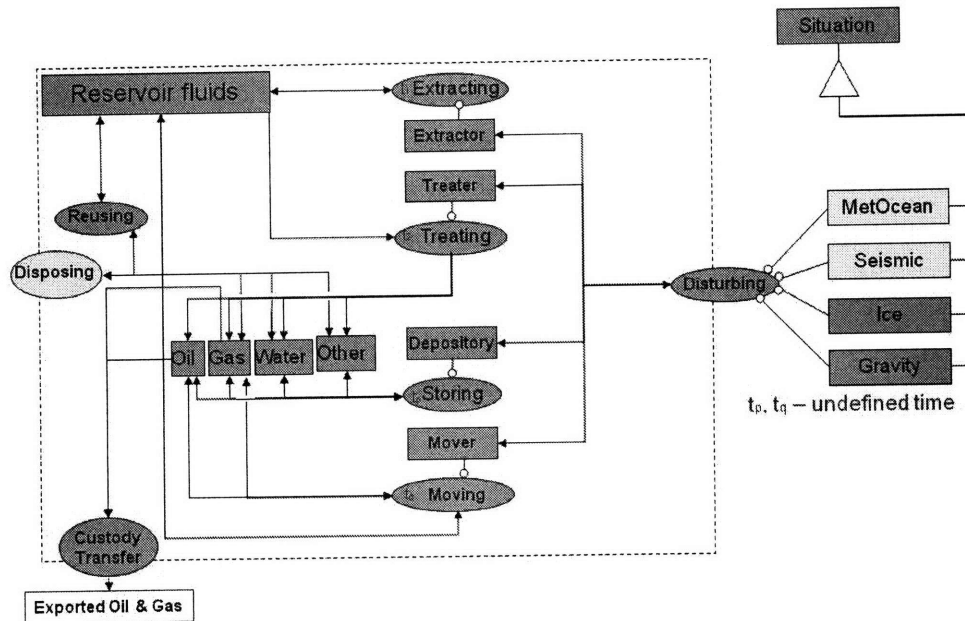


Figure 2-3. Level 2 Decomposition of the Oil and Gas Production Process

2.2.3 Third Level Decomposition of the Oil and Gas Production Process

The Level 3 decomposition of the oil and gas production process, Figure 2-4, expands Extracting and Treating to two sub-processes each and adds supporting processes. Extracting is broken up into Reaching and Transfer, which must occur in that order. Treating decomposes to Changing Composition and Changing Properties. (Once again for the formal definitions of all the elements in the decomposition see Figure 2-1 on page 17.) Both of the Treating processes usually occur but not in a particular order. All four of these sub-processes are executed by generic instrument objects: Reachers, Transferers, Composition Changers, and Property Changers.

The Interfacing process interacts with the Moving process and represents specific interface functions that several objects in the oil industry do in nearly every oil and gas production process. This occurs in two places in the production process. First, when moving of reservoir fluids occurs just after extraction there is a need for specific interfaces to the moving process. The wellhead is one of these interfaces. The second occurrence of these interfaces is between Treating and Moving. When the Mover is a discrete mover like a tanker, an interface called an offloading facility is required. The OPD's that show the decomposition of these two interfacing processes are shown in Appendix 2.

On the right side of Figure 2-4, we can see the supporting processes in the level 3 model. These so-called supporting processes each interact with most of the instrument objects of the value delivering processes, but because they aren't directly involved with the delivery of value (that

is, turning reservoir fluids into exported oil and gas) they fall into the second layer processes—the supporting processes(11). It was important to highlight these supporting processes because the decisions on how to do some of these processes are major architectural decisions. For instance, choosing which type of mechanically supporting substructure is to be used turns out to be one of, if not the, main decisions for an offshore facility. Other supporting processes such as maintaining, powering and provisioning, are operationally important but less significant from an architecture standpoint. The level 3 decomposition also shows a few objects outside the system boundary: people who operate and maintain, and provisions for the system which must be accounted for in the interfaces.

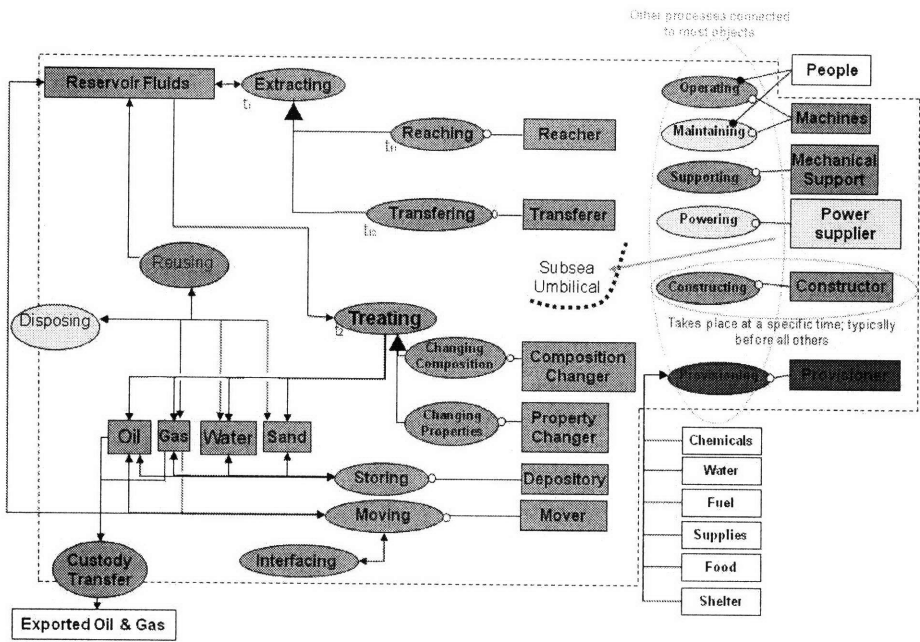


Figure 2-4. Level 3 Decomposition of the Oil and Gas Production Process

The first icon also shows up in level 3. The subsea umbilical icon is an example of a power supplier. We placed icons throughout the decomposition in an effort to make it easier for our sponsors, who are extremely familiar with all the elements that are involved in the oil and gas production process, to understand our different way of decomposing their processes. Each icon shows a concrete example of an element or process that we have abstracted to a more generic representation. We found that the use of these icons helped those in the oil business be more comfortable thinking about the oil and gas production process in this decomposed way. Much as the OPM methodology seeks to discover new solutions by abstracting objects and processes, concretizing an object or process serve to provide an example of one particular solution.

2.2.4 Fourth and Fifth Level Decomposition of the Oil and Gas Production Process

Our lowest level decompositions are the level 4 and 5 decompositions. There are 11 of these level 4 and 5 OPD's all of which can be seen in Appendix 2. Each zooms in on a particular component process in the oil and gas production process in the level 3 decomposition (Figure 2-4). Figure 2-5 shows one example of these level 4 and 5 decompositions. It zooms in on the sub-process Reaching.

Figure 2-5 shows that Extracting is decomposed (in part) to Reaching, which is done by a Reacher. A specialized form of a Reacher is a drill. Reaching has the attributes of: direction, operation time, location, and placement. Each of those attributes has some specialized variant. For example, operation time could be year-round or seasonal. Location is an attribute that describes the geographic location at which the process will take place. Placement is an attribute that describes, in a sense, the vertical "plane" on which the process is supported (mechanically). More discussion of these different attributes of location is found in the next section.

As discussed in the previous section, this and most of the level 4 and 5 OPD's contain icons that link elements in the decomposition to concrete instances in the real world which that element represents as used by BP.

The level 4 and 5 decompositions represent the full extent to which we have decomposed the oil and gas production process. The remainder of the chapter will focus on how this decomposition leads to an architecture generating model in OPN.

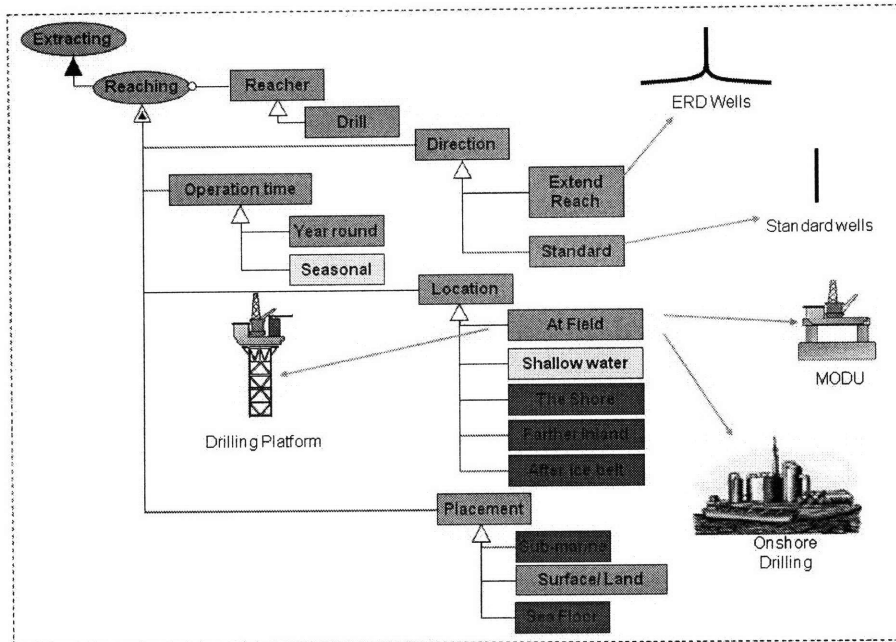


Figure 2-5. One of the 11 Elements of the Level 4 and 5 Decomposition of the Oil and Gas Production Process (for the others see Appendix 2)

2.2.5 Describing Location

The motivation for this work was to generate architectural concepts for a given situation in which an architecture was desired. A particular situation is described and differentiated by the context that surrounds it and the particular parameters for that situation. These parameters are a set of variables that are “given” from outside the system and are understood to be the same for all possible architectures. It could be said that they outline the architectural design space for each situation. These parameters include information about the reservoir(s), the geologic, MetOcean, and other information about the environment surrounding a potential oil project. They also include information about the financial markets and political environment that the oil project must exist in.

We developed a framework we called Location that describes a set of those parameters concerned with the geography of the area near the potential oil project. We needed a framework that could be used to describe the various situations that might require an architecture concept for an oil production process. This framework had to include all the information that would need to be filled-in to completely describe the situation. It would have to include enough detail such that the different concepts could have metrics calculated about each of them. The framework had to be general enough that it could be used for a wide range of situations.

Figure 2-6 is an illustration of the location framework that we developed. A particular situation for an architectural concept would have all of these possible locations: At the Field, in Shallow Water, on The Shore, Farther Inland, and in Ice Free Water (if there is ice present). We think these locations are the minimum set needed to describe a situation in enough detail to be able to generate architecture concepts and calculate meaningful metrics about each of those concepts. Different situations would have different attribute values at each of these locations, but each situation, in general, would have these locations and the same attribute decisions.

Figure 2-6 illustrates the five locations, while Figure 2-7 shows what attributes need to be defined for each location for a given situation. Location 1, At Field, describes the location offshore that is directly above the reservoir(s). Before you could generate concepts you would need to fully describe this location, including its water depth, geographic location (as in latitude and longitude coordinates or distance from the other locations), the area of the reservoir or field in question (i.e. km²), whether or not that location was prone to seismic activity and the characterization of that seismic threat, the ability of the sea floor and surrounding environment to be built upon, whether or not there are concerns of arctic ice and the characterization of that ice disturbance, the particulars of the MetOcean conditions at that location, and finally the nature of the environmental sensitivities of that location.

Location 2, Shallow Water, represents the location closest to the field with a shallow enough water depth that shallow water structures could be built and/or from which extended reach drilling to the field could take place. The water depth of this location would be in the up to 50 m. The attributes required to be defined for this location is shown in column 2 of Figure 2-7. This includes the one attribute decision that has not been described yet, Infrastructure. The attribute infrastructure requires a characterization of any existing infrastructure at a location that could be leveraged either in construction or during production. Examples might be existing transportation lines or nodes at a location or other oil facilities or pipelines that already exist at a location.

Location 3, The Shore, describes the closest point on shore to the field or shallow water location that could be used in an architecture for logistics or as a location for a facility in the production process. Location 4, Farther Inland, describes a place farther inland that either has some existing useful infrastructure or is more conducive for the construction of some useful infrastructure in an architecture. Location 5, Ice Free Water, is only used in arctic situations and describes a location that is the closest spot to the field with year-round ice free water. This is

where one might locate a logistics node or production facility in an architecture. Each of the required attributes for these locations can be found in Figure 2-7.

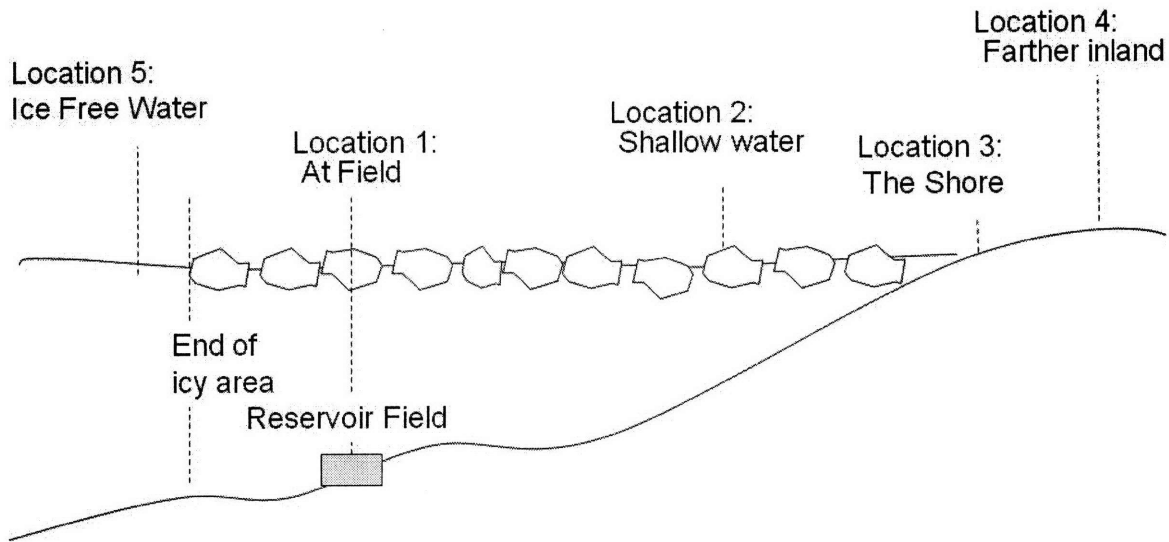


Figure 2-6. Locations for a Concept

Important Attribute	Location				
	1	2	3	4	5
Depth	X	X			X
Location	X	X	X	X	X
Area of Field	X				
Seismic	X	X	X	X	
Buildability	X	X	X	X	
Ice Concerns	X	X			
MetOcean	X	X			X
Infrastructure		X	X	X	X
Env Sensitive	X	X	X	X	X

Figure 2-7. Attributes for Locations

2.3 Developing the Morphological Matrix for the Oil and Gas Production Process

The first step toward making an OPN model to generate architectures for the oil and gas production process was to construct a morphological matrix that outlines the design space from which all the architecture concepts are generated. The morphological matrix is basically a collection of all the major decisions that must be made in order to define a concept for the architecture. In order to collect all the major decisions we referred to the decomposition of the oil and gas production process outlined in the preceding sections. Specifically, we looked at all the level 4 and 5 OPD's for places where decisions must be made.(14) An example of this is shown in Figure 2-8.

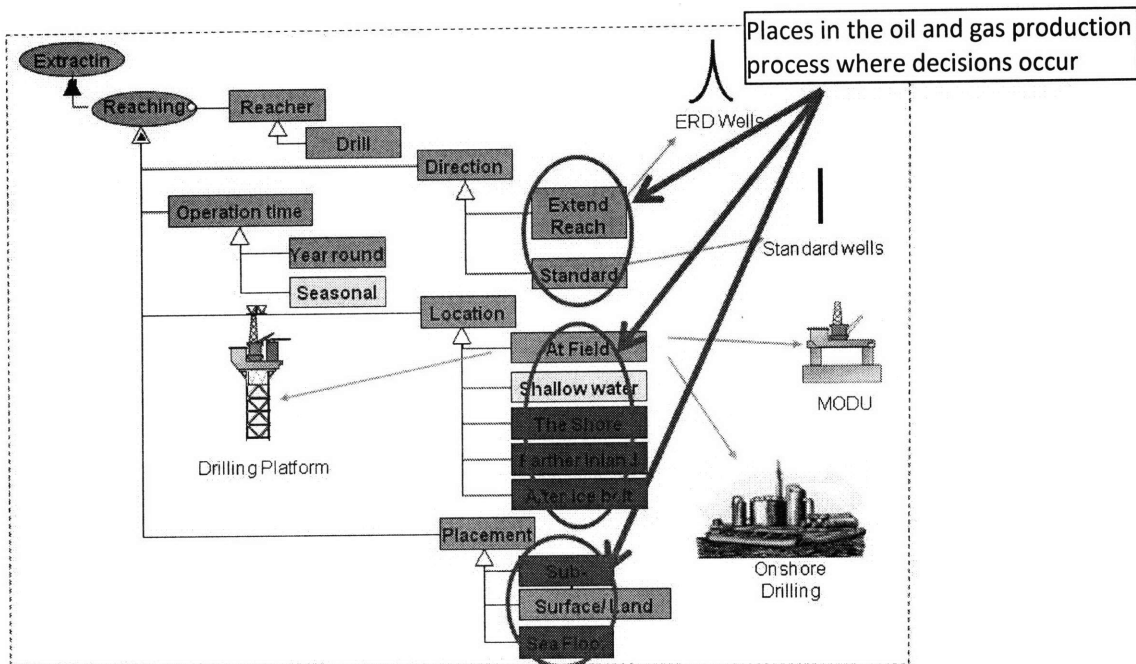


Figure 2-8. Example of Decisions in Decomposition

The choices that come from the OPD's turn out to be choices of processes, choices of form, and choices of attributes. The traditional morphological matrix has rows that each represent one decision with the name of that decision and the individual choices that could be made. However, this style of morphological matrix doesn't convey any sense of the hierarchy of process, then form, and then attribute. So we developed a hierarchical morphological matrix, shown in Figure 2-9, that includes extra information for each row where the name of the decision used to appear. In this new style of morphological matrix one lists the process decision and any individual choices about that decision. Next, any decisions about the attributes of that process are listed. Then, one lists form decisions that are related to that process and individual choices about those forms. Finally, one lists attribute decisions about those forms and individual choices about those attributes. These decisions are listed in an outline format to show the structure of the hierarchy.

One could go further in this idea of hierarchy in the morphological matrix and show a cascading set of lower decisions to be made based on higher level decisions. For instance in a hierarchical morphological matrix, if there was a decision about process x with two choice possibilities A and B, choice A would have a whole different set of form and attribute decisions below it than choice B. Each of those form or attribute decisions could have different sets of related decisions below them. We have not produced such a matrix and producing a visualization of it would be difficult but the style of morphological matrix has the potential to be useful.

For the oil and gas production process morphological matrix, after locating all of these different places in the decompositional model where a decision was necessary, the decisions are gathered in the matrix shown in Figure 2-9, using the hierarchical method of listing the decisions described above. In this matrix, the decisions in the somewhat outline format are on the left while the individual choices for each decision or listed on the same row on the right. Shaded/yellow boxes indicate decisions or individual choices that were left out of the eventual OPN model in order to simplify the model.

Every possible concept is represented in this matrix. A concept would be an instance of choosing one individual choice from each row. This combinatorial space is very large. Ignoring the shaded/yellow squares and any constraints about things that can and can't go together there are about 224 million concepts. This is way too many concepts to consider or generate. Luckily there are a great number of constraints that limit different individual choices being paired with other individual choices.

Function	Form	Attribute	Possible Choices					
Extracting-Reaching	Wells	Drilling	Standard			Extended Reach		
		Well Head Location	At field			Shallow Water		
		Drilling Season	Year Round			Seasonal		
	Driller		Platform		MODU		Both	
Extracting-Transport	Reservoir Driver -push		Gas and Water Inj		Gas Inj		Water Inj	
Extracting-Transport	Reservoir Driver- pull		ESP Pumps		Gas Lift		None	
Interface Extracting	Tree Type		Dry		Wet		Mixed	
	Facility Between Wellhead and Treating		Satellite Platform		Sub-sea Manifold		None	
Treating	Treating Facility	Geographic Location	At field		Shallow Water	The Shore	After Ice Belt	
		Vertical Placement	On land		On Sea Surface		On Sea Floor	
		# of Sites	One				Multi	
		Operational Season	Year Round			Seasonal		
		Substructure	GBS	SPJ	Semi	FPSO	Artificial Island	None
Storing	Storing Facility		Yes				No	
		Storing Geographic Location	At field	Shallow Water	The Shore	Farther Inland	After Ice belt	None
		Storing Vertical Placement	On Land		On Sea Surface		On Sea Floor	
		Substructure	GBS	SPJ	Shipshape	Artificial Island	None	
Moving	Mover from Wellhead to Treating Facility		Pipeline			None		
	Mover From Treating Facility to Storage		Pipeline		Tanker		None	
	Mover From Treating Facility to Treating Facility		Pipeline			None		
	Mover To Custody Transfer		Pipeline		Tanker		None	
Interface Moving	Interface to Tanker from Treating Facility		Off-loading System			None		
Interface Moving	Interface to Tanker to Custody Transfer		Off-loading System			None		
Custody Transfer	Custody Transferer	Location	At field	Shallow Water	The Shore	Farther Inland	Distant Shore	

Figure 2-9. Hierarchical Morphological Matrix of the Oil and Gas Production Process

2.4 Constraints within the morphological matrix

This section will discuss the constraints that correspond to the decisions and choices in the morphological matrix that are necessary to prune the infeasible concepts from the design space. Figure 2-10 shows the constraints within the morphological matrix. (See Appendix 3 for further explanation about each constraint.) These constraints in general contain a great deal of the industry knowledge that is required to construct an architecture generator. Some of the constraints are buried in simple physical understandings. For instance, if you choose to have a treating facility on the shore it makes no sense to later choose a ship-shaped substructure (FPSO-Floating Production, Storage and Offloading). Others constraints are only clear to those in the oil industry. For instance, you wouldn't have a pipeline mover from a Treater with a FPSO substructure. It took more than a year for the author to gain the industry knowledge to develop these constraint relationships, but for those already familiar with the oil industry these relationships are better understood and almost second nature. One of the great advantages of this method is that it takes this tacit information about these relationships, which is well known by experts in an industry, and formally collects it, makes it explicit, and leverages it to "automatically" generate concepts. One key lesson was that this method is much easier and quicker if done by those already knowledgeable about the industry in question. This is particularly true if the person is knowledgeable about these constraint relationships, major architecture decisions and rules of thumb concerning different elements of an architecture or system.

About 20% of the constraint relationships shown in Figure 2-10 were "discovered" through trial and error in the development of the OPN model. That is, after test runs, some nonsensical concepts would be generated and the OPN model would have to be adjusted to eliminate them from being generated in the future by adding some additional constraints to the model. These "discovered" relationship constraints are buried in the logic of the OPN model and now are difficult to express explicitly.

#	Constraint
1	no standard drill with shallow water well location
2	no platform driller with wet or mixed trees, no MODU with dry or mixed trees
3	no dry trees with sub sea manifolds; wet or mixed trees must have sub sea manifold
4	no standard drill with shallow water, the shore or after ice belt treater location
5	no dry trees on sea surface treater placement, no wet trees with on land treater placement,
6	no mixed trees with on land or sea surface treater placement
7	no on sea floor or on land treater placement or dry or mixed trees or the shore treater location with Semi or FPSO substructure,
8	no at field or shallow water or after ice belt treater location or platform driller or on sea floor or on sea surface treater placement with no substructure
9	no the shore or after ice belt treater location or on land or on sea surface treater placement with GBS or SPJ substructure,
10	must be no storage facility for FPSO treater
11	no on land storage placement with at field or shallow water or after ice belt storage location,
12	no on sea surface storage placement with the shore or farther inland storage location,
13	no on the sea floor storage placement with the shore or farther inland or after ice belt storage location
14	no the shore or after ice belt or on land or on sea surface with GBS or SPJ,
15	no on sea surface or on land or the shore with Semi or FPSO,
16	no at field or shallow water or after ice belt or on sea floor or on sea surface or FPSO with none
17	no dry trees with pipeline from wellhead to treater
18	must be no mover from treater to storage for FPSO treater and if there is no storage facility,
19	no pipeline mover from to storage for Semi Treater
20	no pipeline mover to custody transfer with FPSO treater or shipshape storer or semi treater
21	no offloading facility with pipeline mover or none for treating to storage mover, must be none if no storage facility
22	no offloading facility with pipeline mover or none for mover to custody transfer
23	if no storage facility then 'none' for 'to custody transfer mover if match with treater location,
	if the treating location is ...
24	...at the field then the wellhead location must be at the field
25	... in shallow water then the drill type must be extended reach
26	... in shallow water and the wellhead is in shallow water then mover from wellhead must be none
27	... in shallow water and the wellhead is at the field then mover from wellhead must be pipeline
28	...the shore then the drill type must be extended reach and the mover from wellhead must be pipeline
29	...after ice belt then drill type must be extended and the wellhead must be at field and mover from wellhead must be a pipeline
	if storage location is...
30	...in shallow water then treating location must be at the field if there is a mover to storage or after ice belt if the mover to storage is tanker
31	...at the shore then the treating location can't be at the shore
32	...at the shore then there must be a mover to storage if treating location is at field or in shallow water
33	...at the shore and the treating location is after ice belt then mover to storage must be a tanker
34	...after ice belt and the treating location is after ice belt then mover to storage must be a none
35	...after ice belt and the treating location is at field then mover to storage must be a tanker
36	... farther inland then the treating location must be at the shore and the mover to storage must be pipeline

Figure 2-10. Table of the Constraints in the Oil and Gas Production Process Model

2.5 Metrics and Outcomes

Enumerating all the architectures in a design space is an important and informative method in the development of complex systems. The method discussed in this chapter helps to ensure the full envelope of the design space is explored. It will hopefully lead to new concepts that might not have been developed using more traditional concept development methods. However, enumerating hundreds or thousands of concepts is not helpful unless there is a way

to make sense of them all. That is, the ultimate aim is to so develop a single system, so there must be some down selection. This is enabled by thoughtfully chosen metrics, applied in a well planned strategy. One advantage to this method is that it can employ very rough, easy-to-calculate, metrics to cull 90% of the concepts, leaving the preferred concepts for further study or for further down selection by more sophisticated metrics. This method also enables the use of multiple metrics, allowing for a multi-disciplinary search for the best concepts and allowing human decision-makers the ability to see how the application of different measures of merit affect the set of feasible concepts for a system. This section will discuss how metrics can be used in the generation of oil production system architectures.

2.5.1 Possible Metrics

Ultimately a decision maker want to choose the best concept based on which concept delivers the most value. Value is benefit at cost. (11) For the oil and gas production problem, benefit is measured in the systems performance. This would include throughput of exportable product and treatment efficiency. Cost in a broad sense would include resources expended (cost in a more narrow sense) and length of construction or development schedule. Both benefits and costs could be metrics used to measure concepts to find the ones with the best value. In this case, we assumed that the performance or benefit delivered from each concept would be equal. Thus we were only interested in measuring the cost side of value. These metrics are then iso-performance metrics, because they are used with the understanding that the performance of each concept is equal. Therefore the concepts with the least cost have the best value.

Cost

Cost is the first obvious metric that could be used when enumerating thousands of oil and gas production system architectures. A cost metric would be implemented by attaching some cost value to each decision choice and then summing all those values for each concept. This could be done in two ways. The first would be to have the cost values on each choice process in the OPN model and as a token passes, the corresponding cost value from a choice would be added to some accumulating cost variable on that token. The second way would be to associate the cost for each choice and sum them up in post processing. This could be in a simple spreadsheet or in a more complex cost estimating software.

In this project we were not able to extract all the need cost data to be able to apply this cost metric to the architectures generated by the model discussed in the next section. However, using a limited knowledge of the nature of the cost estimation relationships in this design space a pseudo cost metric was applied to the 878 architectures that were generated by the model

discussed in section 2.6. Figure 2-11 shows the pseudo cost for each of the generated architectures in the darker (blue) bars. In this chart a couple of trends are shown that are commonly seen in similar metric charts for architectures of different systems. First, a series of step-like patterns in the cost metrics can be seen. These steps indicate that some significant architectural choices are being made at each step. To better understand the design space it may be a good idea to investigate what major decisions and choices correspond to these steps. Additionally, on the left there is clearly a set of concepts (the top 10) that are distinctively lower from a cost perspective. That makes it easy to just take those 10 concepts on further in the development and analysis and throw away the remaining concepts. It is interesting to know what common features all of these preferred concepts share and how they are functionally different.

Figure 2-11 also shows an example of a very simple metric that may be useful. The white bars at the bottom show the number of major elements in the architecture as indicated by the scale on the right. This number ranges between 3 and 8 and counts elements like platforms, pipelines, tankers, and subsea wells. One interesting trend in this graph is that the pseudo cost does not strongly correlate with the number of elements in a concept as one would expect. It can be seen that the number of elements in a concept varies widely across the entire spectrum of cost. This would at least indicate in a real case that the number of elements is not a strong influence in the cost of a concept, but that other architectural decisions have a stronger impact on cost.

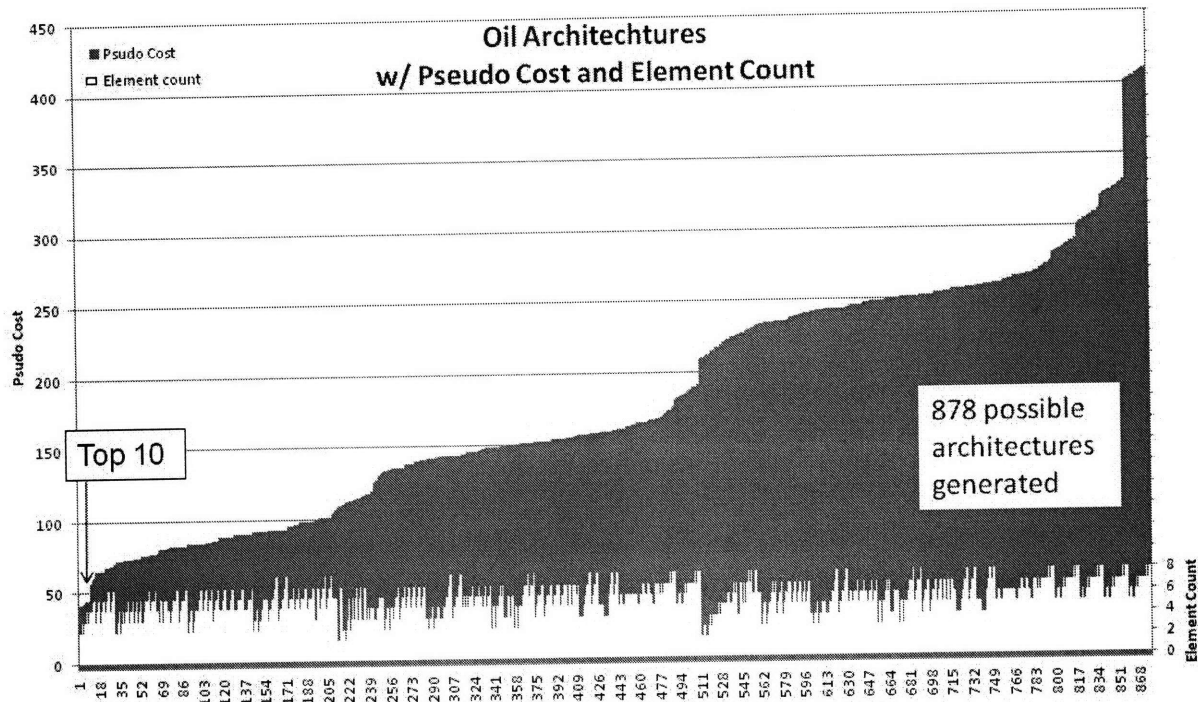


Figure 2-11. Example Plot of Metrics for Oil and Gas Production Process Architectures

Schedule

Another metric that was of interest to our sponsors and that would be as easy to implement as the cost metric is construction/development time. Schedule is a major factor in the development of many systems in the commercial world. To implement this metric one could estimate the construction time by assigning some time-to-build value to each choice and cleverly summing them up to a total development/construction time. Because construction of elements in large systems is nearly always done in a somewhat parallel manner, the total development time wouldn't be a pure sum, but one could easily enact some rough time estimation procedure. Once again the aim would be to roughly estimate some time metric to be able to tell the possible desirable concepts from the undesirable concepts.

Net Present Value

With knowledge of construction schedule and cost, and estimates of production schedules, one could also use Net Present Value (NPV) as a metric to sort the generated concepts. The idea of using NPV to sort architectures generated by OPN models is discussed in more detail in Rozenblum's thesis.(19)

Technology Readiness Level

The final metric we considered using to sort the 1000's of possible architectures that may be generated by this method was to assume their development risk as measured by Technology

Readiness Level (TRL). TRL is a score given to a technology or component within a system to gauge how far it has matured in the development cycle. A technology with a TRL value of 7 represents a technology or component that is mature and fielded. A TRL value of 0 represents a technology or component that is only an idea. The integers between 0 and 7 fill in the spectrum of development between those two extremes. (See Appendix 5 for the full scale.) In order to use TRL as a metric for sorting architectures we needed data that matched oil industry components and their TRL scores. Our sponsor was able to provide such data, but it was for a more specific problem and design space which was different enough from our general model (discussed in the next section) that we had to make an alternate model. This alternate specific model and its further use of TRL as a metric are discussed in section 2.7.

2.6 Implementation of the Model

With the idea of metrics now in place, this section will discuss the implementation of the oil and gas production process as an OPN model that generates all the feasible architectures. The implementation of the model is facilitated by OPN.(12) It begins by making the various objects and processes to be used in the OPN model and connecting them in a logical way. Next, code is written to be performed by each process. Then constraint coding is written for the connections as pre- or post-conditions to prune infeasible concepts before they are created. Finally, the model is run several times, with the resulting concepts being checked manually each time for infeasible instances. If such infeasible concepts are found, the model is adjusted to eliminate those infeasible concepts from being generated again. This “bug checking” process is iterated until there are no infeasible designs in the generated set of concepts.

Figure 2-12 shows the layout of how the OPN model is populated with processes and objects. The objects (rectangles) are the decisions from the morphological matrix shown in Figure 2-9. Initially objects were placed in the OPN model labeled and in the same order as the decisions in the morphological matrix. After each of these objects, OPN processes (ovals) were placed in-line, abreast, and below their respective decisions. This is shown in two examples in Figure 2-12, with the OPN processes (ovals) replaced by icons to illustrate what each OPN process represents. Arrows are added as shown in Figure 2-12, branching from the decision OPN object (rectangle) to each of the individual choices for that decision and then collecting again at the next decision object. In this way, the diagram in Figure 2-12 illustrates the flow of data tokens through the OPN model: each process generates a set of tokens such that one token propagates through each of the possible choices associated with that decision (each choice being represented by an oval in the OPN model and an icon in the figure), and the tokens congregate at the next decision (object) in the chain to repeat the procedure. This continues

until the entire tree of decisions and choices has been traversed by tokens; the first pass at the model simply proceeds along the tree in the manner laid out as top to bottom of the morphological matrix seen in Figure 2-9.

On each OPN process which represents an individual choice, code is written that would identify on a token that passes through each process which choice was made in each decision. This was done by assigning a variable to represent each decision, then assigning a different value to that variable based on which individual choice process a token passed through (chose) in each decision. For instance, in the example in Figure 2-12, there would be two variables; the first could be *Sub_typ* and would represent which choice was made in the substructure type decision. *Sub_typ* would equal 1, 2, 3, 4, or 5 based on which of the five individual choices (from left to right respectively) through which the token passed. The second variable could be *MT* and would represent which choice was made in the “Mover from Treating” decision. *MT* would equal 1 or 2 based on which of the two individual choices (from left to right respectively) the token passed through. A token passing through the Floating Production process would have the variable *Sub_typ* set to equal 2. If that same token were to then pass through the Oil Tanker in the next decision it would then also have the variable *MT* set to equal 2. Thus, when a token reached the end of the OPN model and had *Sub_typ*=2 and *MT*=2, this token would represent one architecture that has a Floating Production substructure and an Oil Tanker Mover from Treating. Appendix 4 shows the different variable abbreviations used for each decision in the OPN model, as well as the values used for each of the individual choices in those decisions.

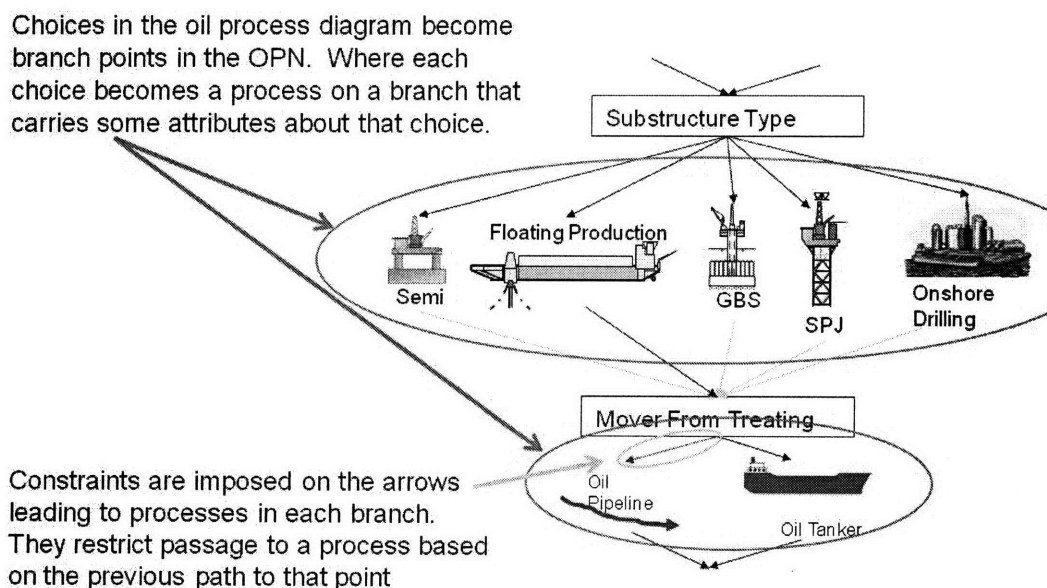


Figure 2-12. OPN Model Implementation Layout

The model, if run in its current configuration, would generate more than 200 million architectures. And if the model were able to finish, the tokens that reached the end would be encoded with all the different individual choice elements it passed through (chose), and after some decoding this could be turned into a description of that instance of an architectural concept. Those millions of architectures would be mostly infeasible and that many concepts being generated would exceed the computational resources available, so constraints are necessary to prune the infeasible concepts in the model.

Constraints (shown in Figure 2-10) are implemented by placing Boolean statements (preconditions) on the arrows leading to each OPN process (these are all the individual choices in each of the decisions). These Boolean statements don't allow the passage of tokens that don't meet the inequality constraints of the statements. Each of the constraints in Figure 2-10 is transformed into Boolean statements on one or more arrows in the model. For instance, a constraint statement might be "no oil pipelines with Floating Production systems" (constraint number 21 in Figure 2-10). Practically this is because it is not possible to hook a pipeline up to facilities with these types of substructures. In the model this constraint statement would be implemented by writing a Boolean statement on the arrow leading to the Oil Pipeline in the Mover from Treating decision that says that the variable *Sub_typ* must not equal 1 or 2. This stops any token from passing to Oil Pipeline if it has already passed through Floating Production in the previous decision, but allows any other token to pass. Constraints can also be implemented that are much more complicated. Two general examples of complicated constraints would be constraints that say "D may not go with C=3 if A=1" or "D may not go with any C but C=2 or any B but B=2 and B=3". All the constraints in Figure 2-10 were implemented in the model in this manner. After many iterations of debugging, 20% of those constraints were added because they were overlooked in the initial thinking.

In OPN, the order of the decisions and constraints in a model, such as this one, affect the model's ability to run to completion. In order to help the model run more quickly and ensure completion with available memory, the model should be structured following two rules of thumb. First all the decisions with more branches (more individual choices in that decision) should be towards the end of the model. The model should generally be arranged with decisions with the smallest number of branches near the beginning and decisions with the largest number of branches near the end. The second rule of thumb is that the model should be arranged such that the constraints are concentrated closer to the beginning of the model. Unfortunately these two rules of thumb often conflict. In order to improve the speed of this model, the decisions were rearranged to follow both rules of thumb as best as possible. An N^2

diagram was made that showed the number of branches for each decision and which decisions had how many constraints related to which other decisions. After calculating a weighted priority for each decision based on its adherence to the two rules of thumb and some trial and error, the current arrangement of decisions in the model was completed. This current arrangement is a good compromise between conflicts of the two rules of thumb. A screen shot of the final oil and gas production process OPN model is shown in Figure 2-13. This model generates 878 architectures in 156 seconds.

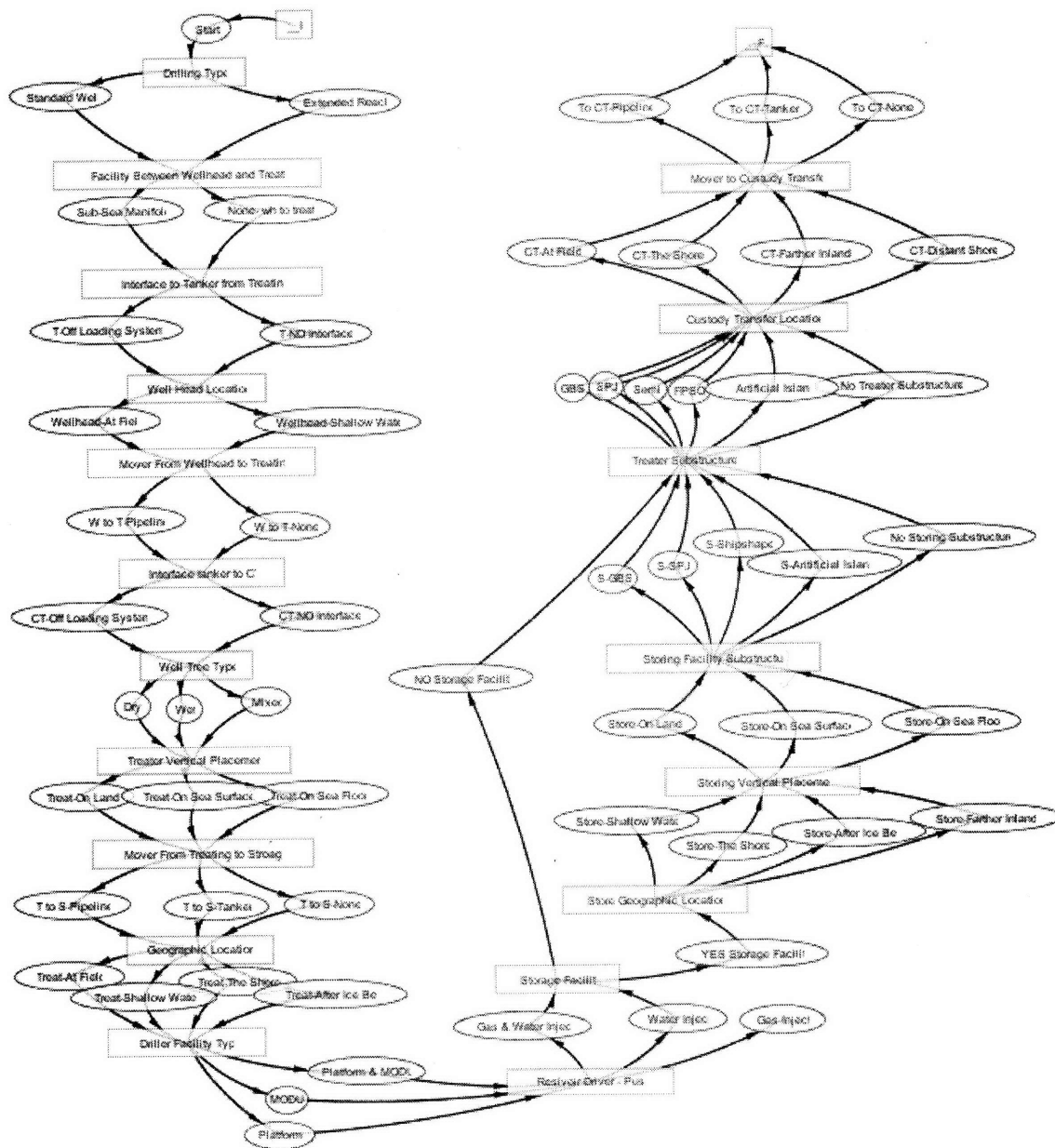


Figure 2-13. Screenshot of Oil and Gas Production Process OPN Model

At this point the model is generating all the feasible concepts. All the tokens that reach the end of the model are instances of valid architecture concepts. Each of these tokens has an encoded string of variables that represent all the choices made in the design space. These concepts can now have metrics applied to them in post-processing to rank them from best to worst or help in selecting the best set to be carried forward in development. One could also implement metrics in the OPN model. This would involve adding code to each of the processes throughout the model that calculates some running metrics as the tokens flow down through the model. This method allows the pruning of concepts in the middle of the model that will clearly not meet some threshold, thus speeding up the model overall. One could also add processes at the end of the model that calculate metrics for each token based on the variable string each is carrying. This allows the user to export the concepts from the model with metric values already attached.

2.7 Alternate Specific Oil and Gas Production System Model

This project also considered the use of Technology Readiness Levels (TRL) as a metric to sort the many generated concepts. The use of TRL as a metric was prompted by our sponsor during work on architecture concepts for a specific problem for which they were able to provide specific TRL's for all the major components of all the major concept branches in the decision space. To use this specific TRL data, an alternate specific OPN model was made to generate all the possible concepts in this specific design space. This model was based on discussion and information from our sponsors and followed a line of thinking similar to a decision tree or decision path. This decision path representing the Alternate Specific Model is shown in full view in Figure 2-14 and in closer views in Figure 2-15, Figure 2-16, and Figure 2-17.

This alternate specific model was made in a very short time span (about 2 days from initial discussion with our sponsor to understand the design space to completely finished and generating architectures). Of course, this was after almost 2 years of learning about the oil and gas industry. In the decision path individual decisions fall into decision categories, which are represented by the corresponding bands across the decision path. Each of those bands corresponds to one of the major processes in the oil and gas production process shown in Figure 2-4. Comparing it to the oil and gas production process decomposition and the subsequent OPN model, both of which were discussed in the majority of this chapter, we found similarities and differences.

The two models are similar in that the more specific model still captures all the major processes previously discussed in the more general model. The more specific model also contains some

of the same alternatives (choices) as the general model. Finally, they both have an overall branching structure that comes from ADG(13) type models.

There are several noticeable differences between the general model and the specific model. The specific model has a much more tree-like structure as opposed to the branching in and out structure of the general model. The tree-like structure handles many of the constraints in the specific model explicitly in the structure of the model. In the general model nearly all of the constraints are handled as pre or post conditions and thus the structure has a more general ADG appearance. The general model does have one instance of this tree-like structure which is seen in the “No storage facility” branch in Figure 2-13. The specific model has many more of these tree branch structures. For example, the wet tree branch, the TLP (Tension-Legged Platform) branch and the no hull branch avoid the use of pre and post condition constraints by having their own train of decisions. There are more than 40 pre and post condition constraints implemented in the general model while the specific model, through the use of the tree-like structure, only has 11 pre and post condition constraints. One example is that coming from the deep draft caissons, only those with wet trees may follow the branch to a MODU (Mobile Oil Drilling Unit). It is therefore much easier to visualize the nature of the decision space on the more specific model. Another difference is that the specific model does not address many of the location and placement related forms and attributes of the general model. It also does not address other more specific forms and attributes of the general model. This more specific model concentrates on floating hull types and leaves out any platform-like structures which are both represented in the more general model. This is because in the situation that this more specific model addresses, the water is too deep for fixed structures and much more conducive to floating facilities. The more specific model also addresses several forms, attributes and processes that we thought were too specific and not major architectural decisions in the general model. The main reason of the inclusion of these more specific decisions is their relationship to this specific situation or, more importantly because these are system components that have specific technology development concerns for this particular situation. Therefore, due to the developmental concerns for these more specific decisions they must be included in a model that uses TRL and cost as a metric.

Major Process - Decision

Extracting- Tree Type

Interfacing/Operating- Subsea Elements

Powering- Subsea Umbilicals

Moving- Subsea pipes

Treating- Subsea Treating

Supporting- Hull Type

Extracting- Drilling Facility

Extracting- Electrical Submersible Pumps (ESP) with Mobile Oil Drilling Units (MODU)

Extracting- Drilling Riser Type

Supporting- Mooring Type

Supporting- Seafloor Fixture Type

Extracting/Moving- Riser Type

Constructing- Tow Type

Moving- Long Tieback Subsea Driver

Treating- Processing Facility

Storing- Storage Type

Moving- Offloading Type

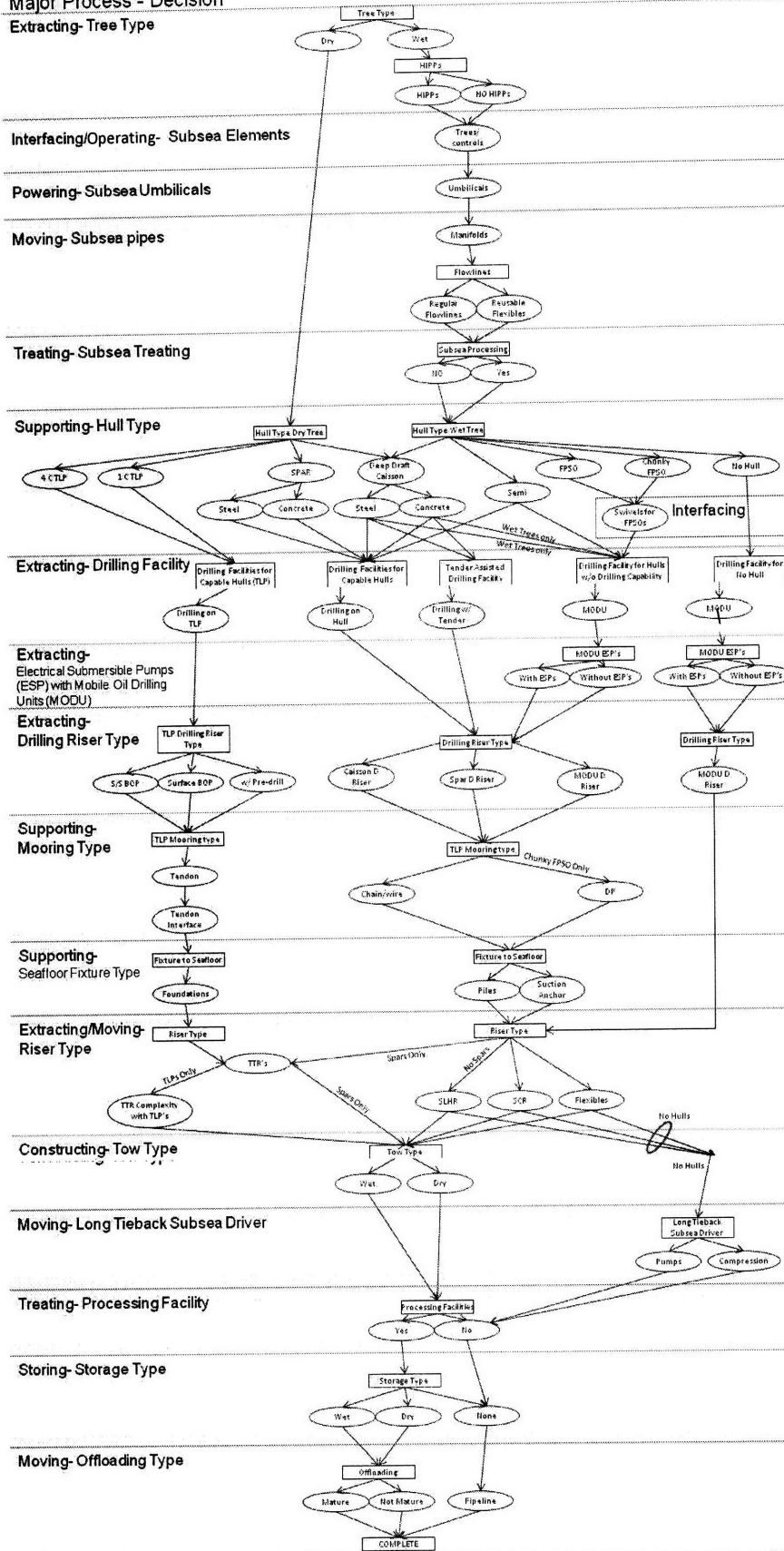


Figure 2-14. Decision Path for Alternate Specific Architecture Model (Full View)

Major Process - Decision

Extracting-Tree Type

Interfacing/Supporting- Subsea Elements

Powering- Subsea Umbilicals

Moving- Subsea pipes

Treating- Subsea Treating

Supporting- Hull Type

Extracting- Drilling Facility

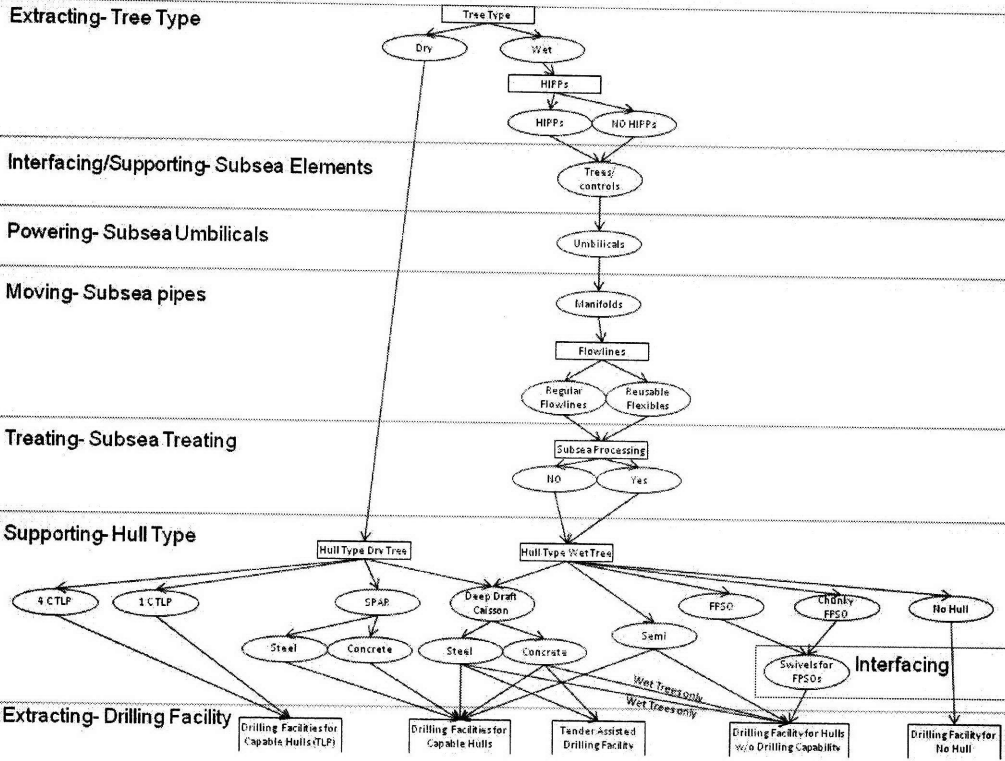


Figure 2-15. Decision Path for Alternate Specific Architecture Model (Part 1 of 3)

Extracting- Drilling Facility

Extracting- Electrical Submersible Pumps (ESP) with Mobile Oil Drilling Units (MODU)

Extracting- Drilling Riser Type

Supporting- Mooring Type

Supporting- Seafloor Fixture Type

Extracting/Moving- Riser Type

Constructing- Tow Type

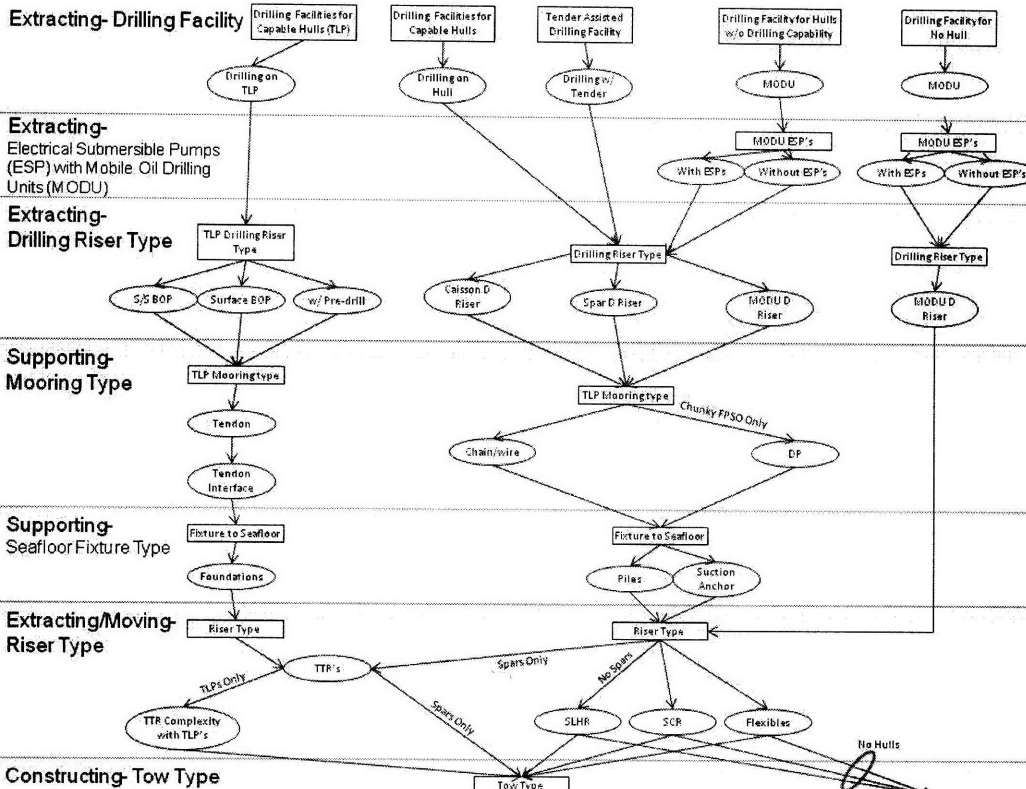


Figure 2-16. Decision Path for Alternate Specific Architecture Model (Part 2 of 3)

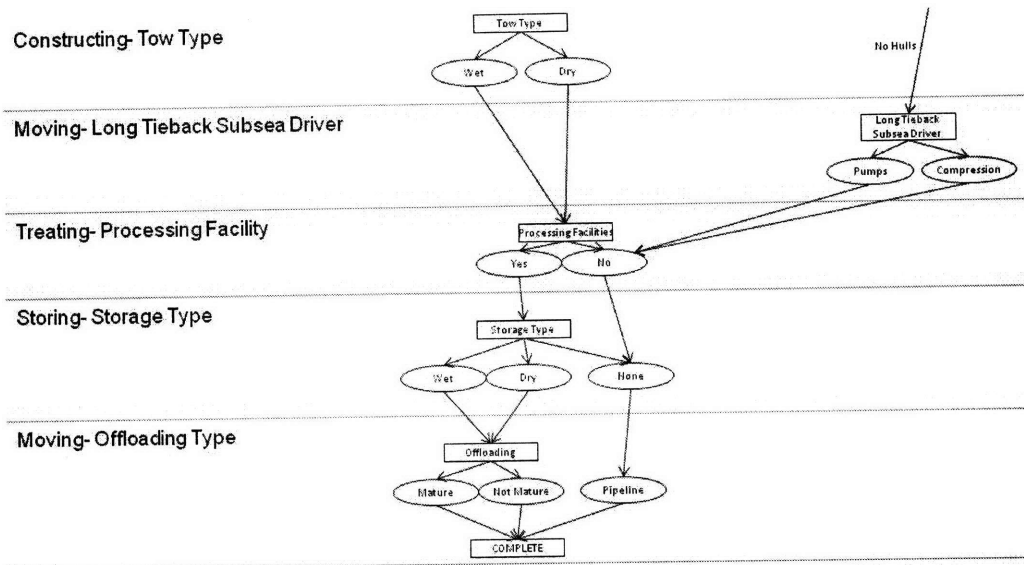


Figure 2-17. Decision Path for Alternate Specific Architecture Model (Part 3 of 3)

2.7.1 TRL Metric and Outcomes

The alternate specific OPN model that was made based on the model in Figure 2-14 , generated 7,536 architectures. The components of each of these architectures had corresponding TRL values. In post processing, each of these architectures had a metric calculated based on the TRL values for the components in each architecture. The TRL values varied between 7, which represents a technology that is proven, fielded and mature, and 0, which represents a technology that is only an idea. Each architecture had a varying number of components based on its particular route through the decision path. (See Appendix 6 for TRL and cost data for each component.) For each component in a concept a penalty was calculated based on its TRL. The following relation describes the penalty:

$$\text{Component Penalty} = 7 - \text{TRL}$$

The penalty, therefore, varied from 0 (no penalty) for mature components to 7 for components that must be completely developed.

We looked at two methods for calculating the TRL penalty for a concept made up of its various components. The first uses only the TRL values for the various components and the second uses these TRL values and the cost for each component. The two methods for finding the total TRL penalty for a concept are:

Method 1-

$$\text{Concept Penalty} = \sum \text{Component Penalties for that concept}$$

Method 2-

$$\text{Concept Penalty} = \sum \text{Component Penalty} * \text{Component Cost}$$

for each component in that concept

Because we did not have any cost data for the components or concepts in this specific problem, each component was categorized into three cost groups: high cost components (near \$100 million), medium cost components (near \$25 million), and low cost components (near \$1 million) based on our limited understanding of the nature of the costs in this design space. These costs groups were used to assign some estimated cost value to each component and then added up for each architecture to give an estimated cost for a concept. These costs were also used to weight the TRL penalty in the second method under the assumption that larger cost items will require more development resources than a lower cost item at the same TRL level. The first method is useful when it is more important to simply gauge the relative maturity of different concepts by using their component TRL values.

The TRL penalty and cost for each concept was then plotted on a scatter plot chart as shown in Figure 2-19, Figure 2-20, Figure 2-21 and Figure 2-22. Figure 2-19 and Figure 2-20 use method 1 to calculate the TRL penalty for a concept, while Figure 2-21 and Figure 2-22 use method 2. Figure 2-19 and Figure 2-21 represent TRL values for components if they are used in water depths of 1700m, while Figure 2-20 and Figure 2-22 represent TRL values for the same components used in 2500m water. Each point on the graphs is a single concept. The legend shows the 12 different data point shapes that correspond to the 12 major concept families originally provided by our sponsor. The names and a breakdown for the total number of architectures generated for each of the 12 concept families is shown in Figure 2-18.

Concept Family		Count
1	Caisson Tender Assist Drilling	864
2	Caisson w/o Drilling	1536
3	Caisson w/ Drilling	1728
4	Spar w/ Drilling	192
5	TLP 4col w/ Drilling	24
6	TLP 1 col w/ Drilling	24
7	Semi w/o Drilling	768
8	Semi w/ Drilling	768
9	Small FPSO w/o Drilling	768
10	Long SS Tieback w/o ss Process	48
11	Long SS Tieback w/ ss Process	48
12	Chunky FPSO w/o Drilling	768
	Total	7536

Figure 2-18. Table of Concept Families for the Alternate Specific Model (with the number of concepts generated for each family)

In the plots the data points (complete concepts) closest to the lower left correspond to the concepts that have low cost and a low TRL penalty (the least amount of technology development required).

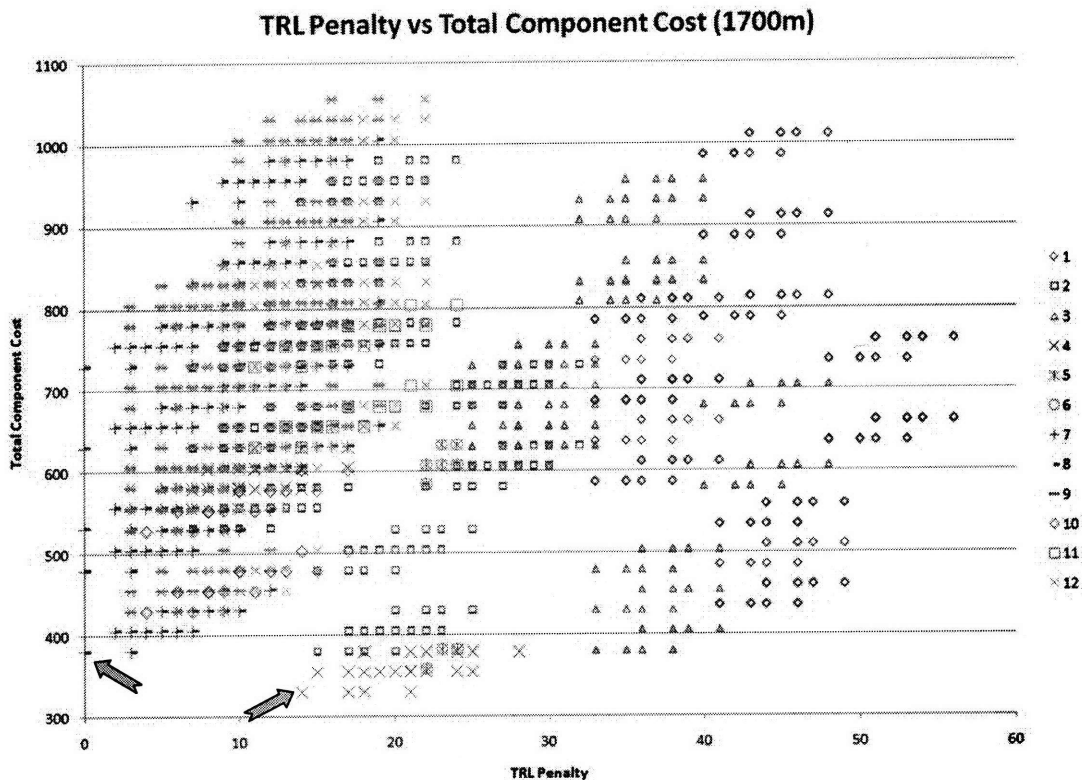


Figure 2-19. TRL Penalty (Method 1) vs Cost (1700m)

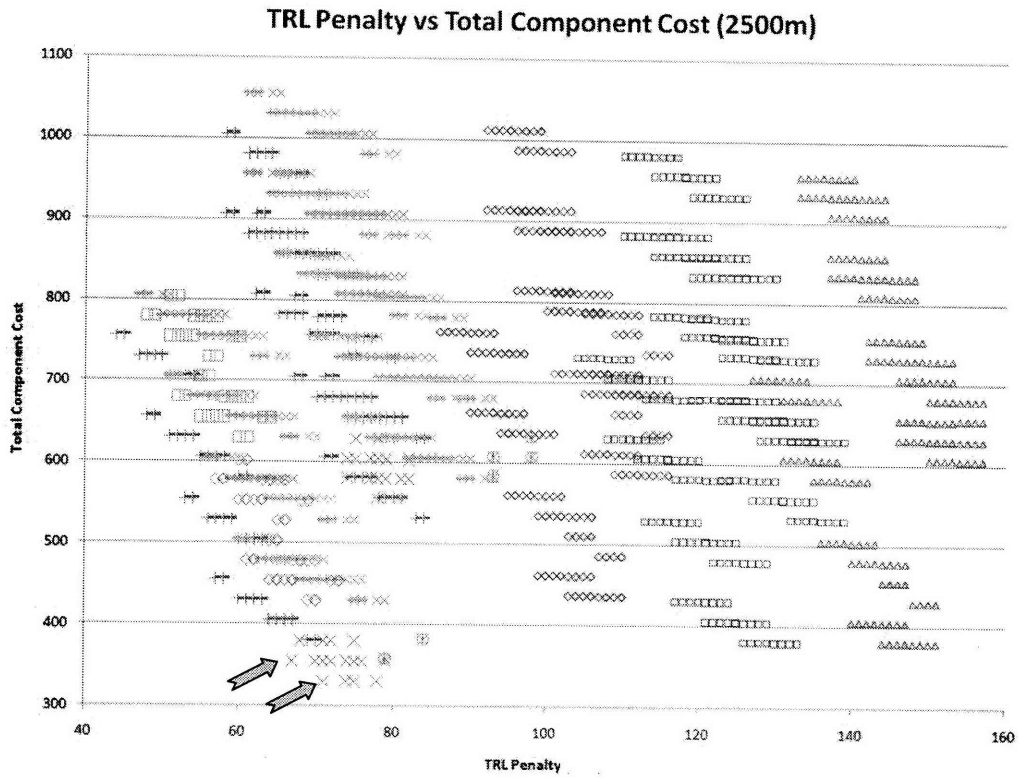


Figure 2-20. TRL Penalty (Method 1) vs Cost (2500m)

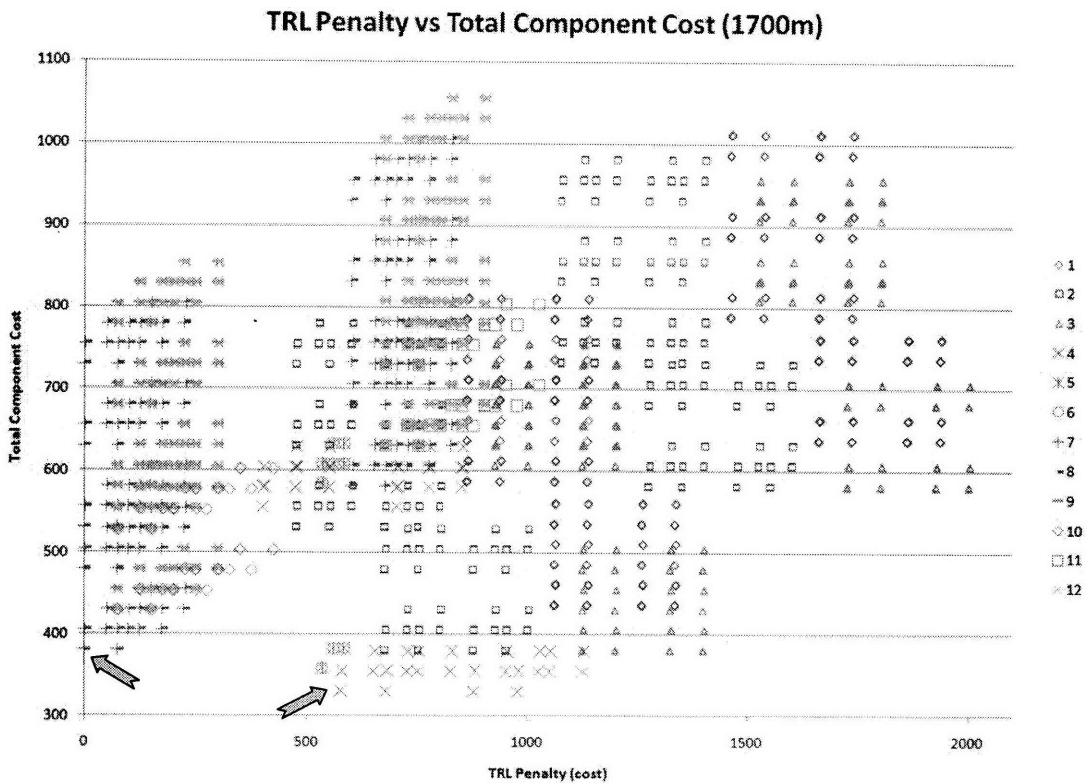


Figure 2-21. TRL Penalty (Method 2) vs Cost (1700m)

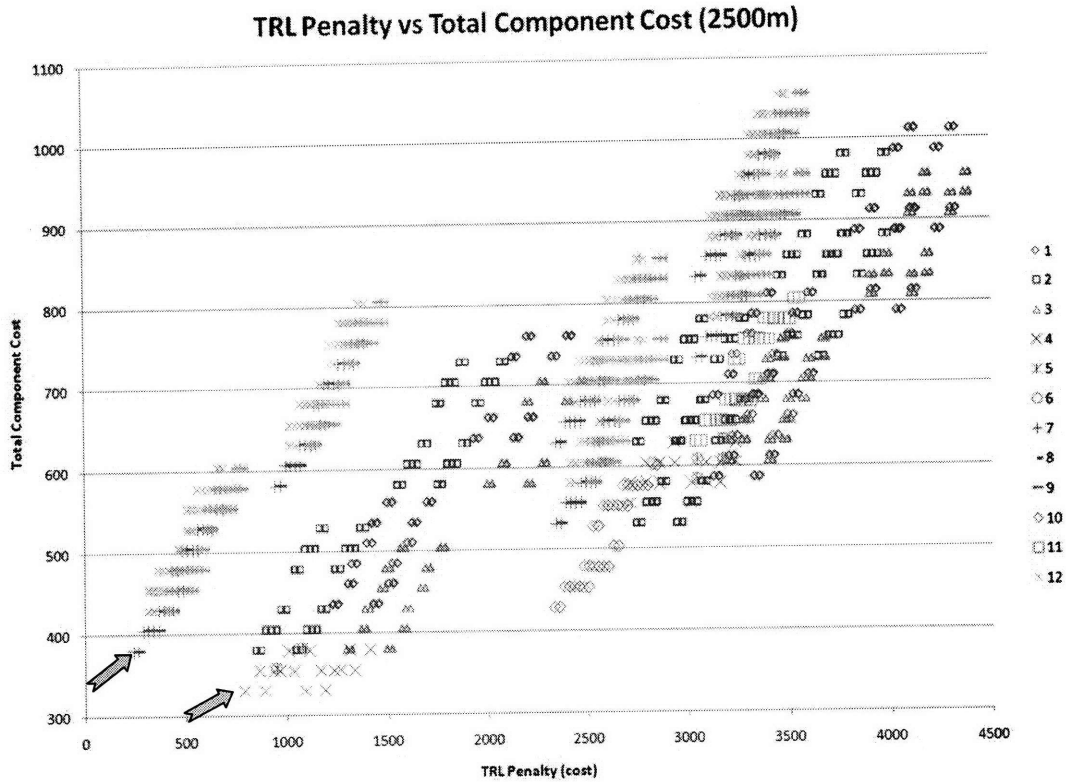


Figure 2-22. TRL Penalty (Method 2) vs Cost (2500m)

Examining Figure 2-19 thru Figure 2-22, several interesting patterns can be seen. For example, how groups of concepts fall on the graph and how families of concepts perform compared to others all based on the TRL's of various component building blocks in a design space. We can find the preferred concepts in each plot. They are the points that lie closest to the lower left corner. Since the author is not the decision maker in this problem, it is not clear whether lower cost or lower TRL penalty is more important, so we have highlighted (with arrows) the two points in each plot that represent the two ends of the spectrum of this idea. Figure 2-23 shows a table of the concepts that lie at these points on the plot. Two sets of concepts pop up as preferred concepts in all four plots. These are shown shaded in blue and red and are highlighted in the second to last column. One interesting note is that according to our sponsor the concept family that seemed the most favorable from their analysis was the spar concepts (concept family number 4). Our analysis appears to have a similar indication.

Figure #	16	16	17	17	18	18	19	19
Concept Family	7	4	4	4	7	4	7	4
TRL Penalty Method	1	1	1	1	2	2	2	2
TRL Penalty	0	578	71	67	0	578	243	578
Total Component Costs	380	330	330	335	380	330	380	330
Concept #	4417,4422-4424	4143, 4144	4143, 4144	4134, 4135	4417,4422-4424	4143, 4144	4417,4422-4424,4431,4434,4437,4438	4143, 4144
Tree Type	Wet	Dry	Dry	Dry	Wet	Dry	Wet	Dry
HIPP's	No	---	---	---	No	---	No	---
Processing Facility	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hull Type	Semi-Submersibl	Spar	Spar	Spar	Semi-Submersibl	Spar	Semi-Submersible	Spar
Hull Material	---	Steel	Steel	Steel	---	Steel	---	Steel
Tow Type	Wet or Dry Tow	Wet or Dry Tow	Wet or Dry Tow	Wet or Dry Tow	Wet or Dry Tow	Wet or Dry Tow	Wet or Dry Tow	Wet or Dry Tow
Storage Type	Wet or Dry Storage	Wet Storage	Wet Storage	Wet Storage	Wet or Dry Storage	Wet Storage	Wet or Dry Storage	Wet Storage
Offloading Type	Tanker	Tanker	Tanker	Tanker	Tanker	Tanker	Tanker	Tanker
Mooring Type	Chain/wire/poly	Chain/wire/poly	Chain/wire/poly	Chain/wire/poly	Chain/wire/poly	Chain/wire/poly	Chain/wire/poly	Chain/wire/poly
Seafloor Fixture	Piles	Piles	Piles	Suction Anchor	Piles	Piles	Piles	Piles
Riser Type	SCR	TTR	TTR	TTR	SCR	TTR	SCR or SLHR	TTR
Subsea Manifold	Yes	---	---	---	Yes	---	Yes	---
Subsea Umbilical	Yes	---	---	---	Yes	---	Yes	---
Subsea Flowline	Reuseable	---	---	---	Reuseable	---	Reuseable	---
Subsea Processing	No	---	---	---	No	---	No	---
Drilling Facility	MODU	Drilling from Hull	Drilling from Hull	Drilling from Hull	MODU	Drilling from Hull	MODU	Drilling from Hull
MODU ESP's	No	---	---	---	No	---	No	---
Drilling Riser Type	MODU Riser	Spar Type Drilling Riser	Spar Type Drilling Riser	Spar Type Drilling Riser	MODU Riser	Spar Type Drilling Riser	MODU Riser	Spar Type Drilling Riser

Figure 2-23. Table of Preferred Concepts from TRL Model

2.8 Summary

This chapter discussed the method used by this research effort to develop a model that generates many possible architecture concepts for an offshore oil production system. It began by decomposing the oil and gas production process into its basic objects and processes. This was done in order for the researchers, which were new to the oil business, to better understand the process, and to find the decision points within the development of oil production system architectures.

Next the development of a hierarchical morphological matrix and corresponding set of constraints was discussed. This morphological matrix was a tool that explicitly outlines the entire architecture design space and serves as a framework for the creation of an OPN model to generate concepts. The corresponding set of constraints operate as a set of rules that prune the infeasible concepts from the full set of possible architectures. These constraints were

developed from implicit rules of thumb known by most in the oil business, but are rarely expressed explicitly.

Then the metrics that were to be used within the model to rank and evaluate the possible concepts were discussed. This was followed by the steps to implement the general OPN model itself, including the outcome of that model.

Finally, in order to explore the use of TRL as a metric, an alternate more specific model was made to be able to use accompanying TRL data for a more specific oil and gas production problem. This model and the use of TRL as a metric both showed further usefulness for the overall method.

CHAPTER 3. Multi-Reservoir Multi-Facility Connection Generator

3.1 Introduction

In order to address the second part of the oil and gas production problem, the architecture of multiple facilities and multiple reservoirs, this chapter will discuss the development of an Object Process Network (OPN)(12)(10)(14)(15) model that can be used generically to generate all possible links between any two sets of elements. The model will generate all the possible connection matrices, a sub-class of Multi-Domain Matrices (MDM) (20), given two sets of elements to be linked and any constraints on their linkage. The generator discussed was developed for a specific purpose and project, but it is quite conceivable that this model could be used as one of several general tools for building system architecture models with OPN.

This chapter will first provide the motivation for the development of this connection generator. Next, it will explain two possible concepts for this OPN model and why concept that will be implemented was chosen. The chapter then steps through the implementation of the generator as an OPN model. Next, it discusses the addition of constraints to the generator. This is followed by a discussion on the output format and procedures for the generator. This chapter then finishes with a discussion of the limitations and expansion possibilities of the model.

3.2 Motivation

3.2.1 The Problem

The problem that motivated the development of this generator became evident during research conducted at MIT on behalf of BP (21)(1). This multi-reservoir, multi-facility problem centered around developing an oil reservoir model that captured uncertainties in reservoir characteristics as well as uncertainties in the economic characteristics that might surround a project intent on obtaining oil and gas from a reservoir. This model aims to enable BP to better understand the potential profit and loss distribution for a given reservoir. This model was expanded to do the same assessment for a set of reservoirs that were connected to a set of oil production facilities. This would enable BP to better understand the profit and loss distributions for different schemes of facilities connected to reservoirs, hopefully aiding in the selection of the best scheme. This model could take a connection scheme and evaluate it, but each connection scheme had to be entered manually. It was evident that an engine that generated all possible connection schemes, each to be evaluated by the main model, would expedite the search for the best connection scheme and ensure that no possible schemes were missed during the search.

3.2.2 Requirements

The design space for these connection schemes was derived from the following hypothetical situations. There is a set of oil production facilities; for this example we will say there can be up to three facilities. There is a set of reservoirs. For this example, there are three reservoirs. A facility could be connected to one or more reservoirs, meaning that facility would produce oil (or gas) from the reservoirs it is connected to. Finally, there is a set of rules about how these facilities could be connected to the reservoirs. These rules, also called constraints, might be that Facility 1 can only be connected to up to two reservoirs or that Reservoir 2 may only be connected to one facility. With these sets of facilities, reservoirs, and rules there is a finite set of possible connection schemes. An instance of one connection scheme can be represented by a connection matrix. In a connection matrix, the rows represent the different facilities and the columns represent the different reservoirs. Marks in the matrix indicate which facilities are connected to which reservoirs. For example, in Figure 3-1 the “X” in the Facility 1 row and Reservoir 2 column indicates that Reservoir 2 is connected to Facility 1. Figure 3-2 is an illustrated description of what the connection matrix in Figure 3-1 represents.

	Reservoir 1	Reservoir 2	Reservoir 3
Facility 1		X	
Facility 2	X		
Facility 3		X	X

Figure 3-1. Example Connection Matrix

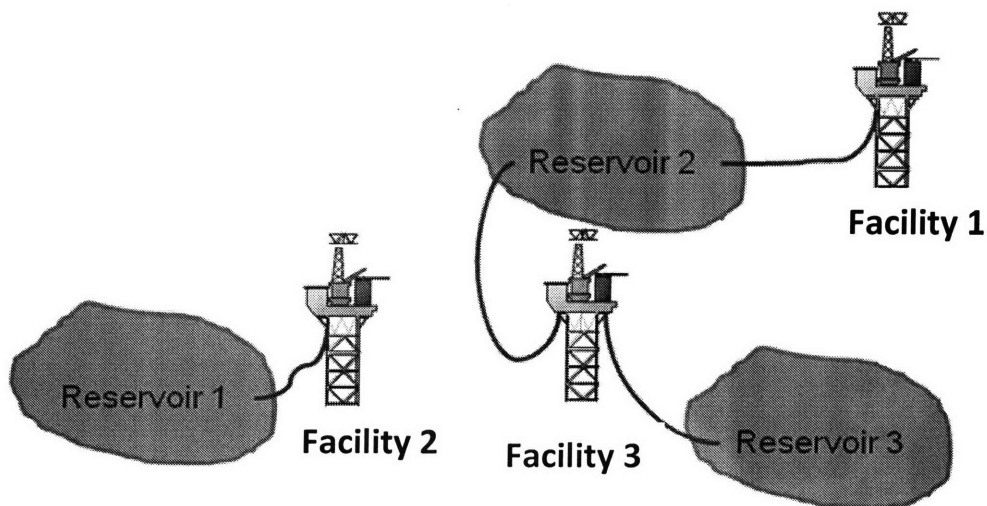


Figure 3-2. Example Connection Illustration

The connection matrix is the form by which a connection scheme is input in to the multi-reservoir, multi-facility problem model. The requirement for the connection generator discussed here created up to a 5-by-5 connection matrix for facilities and reservoirs. This generator had to enable the user to specify:

- The total number of facilities to be connected
- The total number of reservoirs to be connected
- The maximum number of reservoirs that could be connected to each facility
- The maximum number of facilities that could be connected to each reservoir
- Any specific reservoirs that were not allowed to be connected to the same facility as other specific reservoirs

The model would be required to generate all the possible connection schemes for a given set of constraints that fell within the above framework and produce each instance of a connection scheme in a connection matrix or a form easily transformed into a connection matrix. It was also understood that the software language to be used for this generator would be OPN.

3.2.3 Possible Solution Concepts

Two possible OPN model concepts were considered to generate all possible connection schemes for a given set of constraints. The first was a looping-indexing concept. The second was a blocked-cascading concept. Both concepts would generate a connection matrix made of ones and zeros. The ones would correspond to “X’s” in the matrix in Figure 3-1 and zeros would correspond to the blank spaces. The general idea for both concepts was to start with a matrix of all zeros. OPN can be thought of as tokens that flow through the network; recording their path as they go and picking up attributes as they pass through the various processes in the network. These processes can also change each token’s attributes. In this model, as an OPN token passed a process oval (corresponding to a specific facility and reservoir connection), the value of that connection in the connection matrix would be changed to “one”. An example of this type of connection matrix is shown in Figure 3-3.

	Reservoir 1	Reservoir 2	Reservoir 3	Reservoir 4	Reservoir 5
Field 1	0	0	1	0	1
Field 2	0	1	0	0	0
Field 3	0	0	1	0	0
Field 4	0	1	0	0	0
Field 5	1	0	0	0	1

Figure 3-3. Connection Matrix Output Example

The looping-indexing concept

The looping-indexing concept is represented Figure 3-4. The process flow in this concept would work in a similar fashion to how one might structure code in MatLab or C+ to generate all possible connection matrices. This concept uses indexing inner and outer loops to generate all the possible connection instances. As the generator starts, tokens begin by passing through "Facility Indexing", where the facility is set to Facility 1. Then the tokens go to the "Reservoir Connection Loop Start" point, where they branch to the reservoirs. After this, they go to the "Reservoir Connection Loop End", where they branch to go back to the reservoir loop to be connected to more reservoirs, or to the "Facility Indexing" loop to move on to connections in other facilities. Otherwise, they branch to the "Generator End" where instances of the connection matrix are recorded. The "Facility Indexing" would advance the reservoir loop to generate connections for the next facility, much like moving to the next block in the following blocked-cascading concept. Rules would be written to avoid return trips to the same reservoir while in the same facility index. Coding would have to take care of advancing the reservoirs to be "writing" on the next row down in the connection matrices. Some care would have to be taken so that repeats were not produced. For example, in this concept a token path through Reservoirs 3, then 2, then 1 and a token path through Reservoirs 1, then 2, then 3 could be produced, but these would be redundant as both paths represent a facility being connected to all three reservoirs.

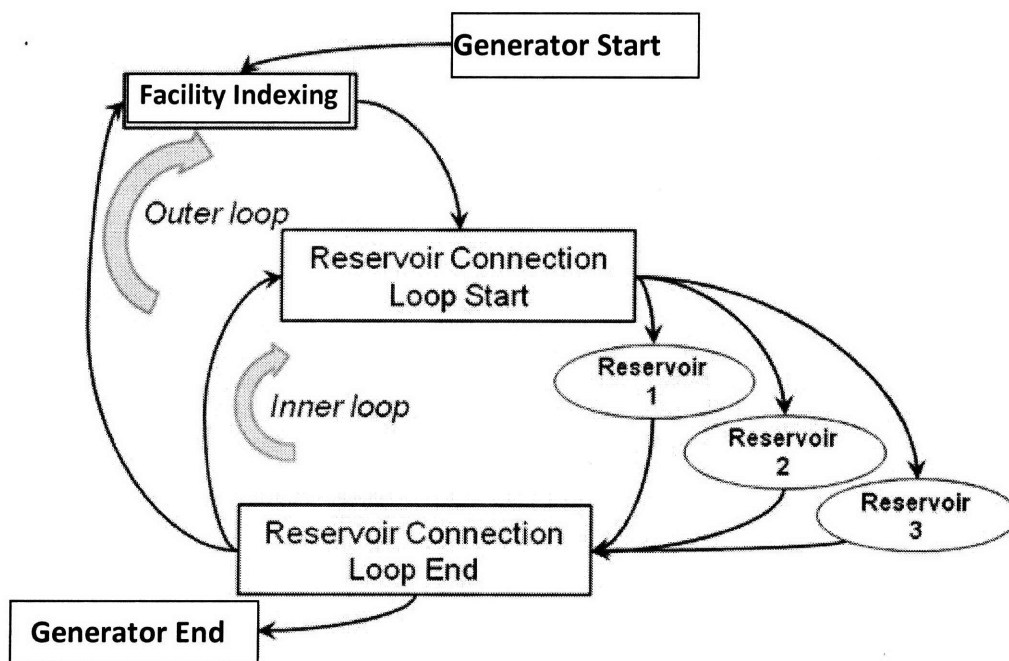


Figure 3-4. Looping-Indexing Concept

The blocked-cascading concept

The blocked-cascading concept would look like Figure 3-6 (which is not an OPN). This concept works by forming blocks as in Figure 3-5 (which is also not an OPN). A block would generate all the connection possibilities for one row of the connection matrix. Each row in the connection matrix would have one specific block that generates all its possible instances. These blocks would be connected in series to form the whole matrix.

Each block would be structured such that all possible combinations of token paths would exist. For the connection instance where Facility 1, as represented by Block 1 (Figure 3-5), is connected to reservoirs 1, 2 and 3, the token would start in the "Block 1 Start" object, then go to Reservoir 1 and cascade down to Reservoirs 2 and 3 before finishing at the "Block 1 End" object. Likewise, for the instance where Facility 1 is only connected to Reservoir 2 the token path would be: Block 1 Start, Reservoir 2, Block 1 End.

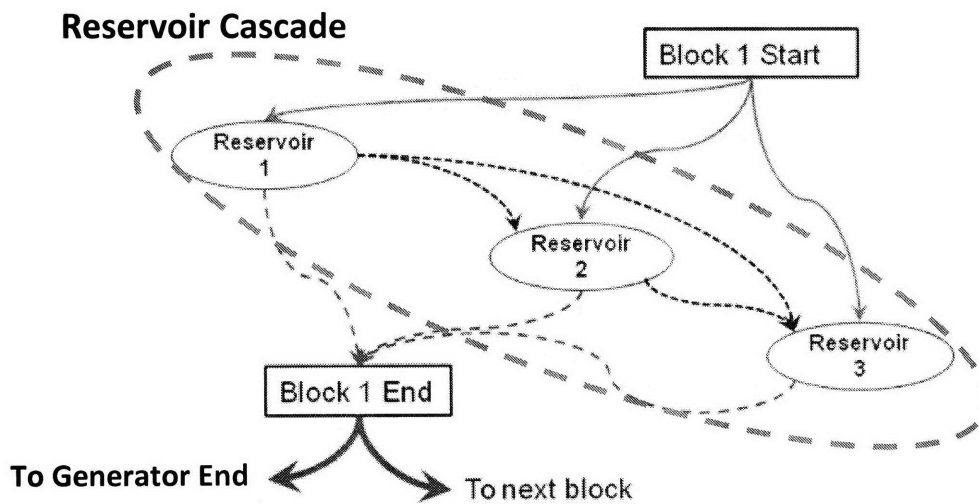


Figure 3-5. Example Block

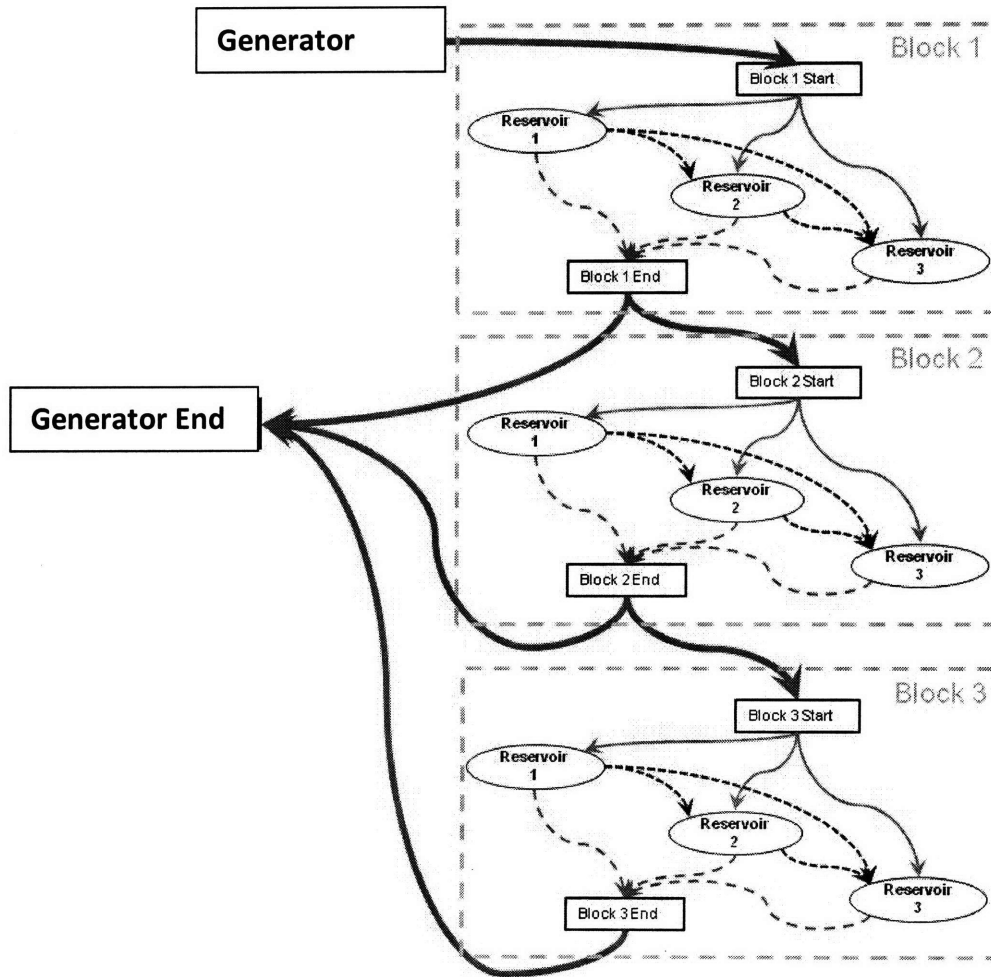


Figure 3-6. Blocked-Cascading Concept

Concept selection

The two concepts each have their advantages and disadvantages. The looping-indexing concept is advantageous because it can be very easily expanded in both reservoirs and facilities and should be able to be quickly made into an OPN model (with very few objects and processes to create). The disadvantages of the looping concept are that it relies heavily on coding, is prone to redundancy, and is harder to visualize in OPN since a good deal of its functionality is in its coding. The blocked-cascading concept has the advantage of eliminating redundancy of connections by the nature of its structure, not relying heavily on intricate coding, and being easy to visualize its functionality, which is largely in its structure. A disadvantage of the blocked concept are the large number of objects, processes and connections to be made in an OPN model which will slow the creation of the model and make expansion tedious. In addition, because one cannot copy and paste in current OPN versions, creating and coding each of many connections will be tedious and redundant work making the model prone to bugs.

After several weeks of effort failed to develop a looping-indexing generator (the effort was plagued with many problems programming OPN to do the indexing from one row to the next), the blocked-cascading concept was selected. This selection was largely due to the difficulties in coding OPN to perform the indexing process, which made the looping concept unattainable in a short span of time. The blocked-cascading concept will be described in detail in the next section.

3.3 The Solution

The general concept of the blocked-cascading structure for the OPN model to generate all possible connection schemes for the oil reservoir problem has been presented in the previous section. Thus, this section will discuss in more detail how this concept was implemented and how the implemented model works. It will start by examining how one block works, then describe how adding blocks completes the structure of the model, and conclude by explaining how the various constraints are handled in the model.

3.3.1 One Block without Constraints

In order to understand how the model works it is best to examine a single block closely. As a reminder, the aim of a block is to generate connection scheme possibilities for one row in the connection matrix. In this case a block generates all the possible connection schemes for a set of reservoirs to one facility. In Figure 3-5, we see that a block has three sections and three types of connections between items within the block. The three sections of the block are: the block start, the block end, and the reservoir “cascade”.

Block sections


The block start initializes the block by setting all the values in the row of the connection matrix, with which this block is concerned, to “zero”. It also sets all the internal block counters (these will be explained further in the constraints section 3.3.3) to “zero”. Finally, it provides a branch point for tokens to be entered into the reservoir cascade. In OPN a branch point describes any place where more than one arrow emerges from an object or process. A token that arrives at one of these branch points makes copies of itself so that a copy of that token follows each arrow leading away from the branch point.


The block end collects all the completed connection instances for that block and passes them to other parts of the model. Each token that reaches the block end is one instance of possible connections between one facility and the set of reservoirs.


The reservoir cascade is made up of the OPN processes that represent a connection to each reservoir in the problem to one facility. In other words, each reservoir in the cascade is responsible for assigning a “connected” value (in this case a “one”) to the various columns in the connection matrix for the row being modeled by that block. The number of reservoirs in the cascade will equal the number of columns in the connection matrix. When a token passes through a reservoir in the cascade, the reservoir assigns a value of “one” in the corresponding position in the connection matrix on that token. For example, in Figure 3-5, which we will call Block 1 or Facility 1, we are only dealing with the first row in the connection matrix. As a token passes through Reservoir 2 in this block, a value of “1” is assigned to the position in the matrix that is in the first row and second column. This corresponds to Facility 1 being connected to Reservoir 2. Conversely, if a token does not pass through Reservoir 2 in Block 1, then Reservoir 2 is not connected to Facility 1, and a zero would remain in the position in the matrix that is in the first row and second column. The reservoir processes in the OPN model also advance counting variables on each token as they pass, which will be explained later in the constraints section (3.3.3).

Block connection types

Again in Figure 3-5, we see that a block has three types of connections between items within the block. The first type is the connections between the block start and the reservoirs in the cascade. The second type is the connections between the reservoirs within the cascade, and the third is the connections between the reservoirs and the block end.

The connections between the block start and the reservoirs are the solid green lines  in Figure 3-5. They feed tokens into the reservoir cascade and allow tokens to enter from any position along the cascade of reservoirs.

The connections between the reservoirs within the cascade are the small-dashed black lines  in Figure 3-5. These connections are responsible for making the group of reservoir processes a cascade. The structure of these connections only allows a token to move down the cascade or out of it, but never up the cascade. Reservoir 1 is connected to each of the reservoirs after it and the block end. Reservoir 2 is connected to each of the reservoirs after it and the block end but not Reservoir 1. Reservoir 3 is connected to each of the reservoirs that may be after it and the block end but not Reservoirs 1 and 2. This pattern is continued for additional reservoirs while the last reservoir in the cascade is only connected to the block end.

The connections between the reservoirs and the block end are the large-dashed red lines  in Figure 3-5. They allow tokens to exit the cascade from any position. They carry completed connection instances to the block end.

Proof that a block generates all possible combinations and no more

The structure of the connection in the block ensures that all the possible connection instances for a row in the connection matrix will be generated, but it also eliminates the possibility of redundant connection instances. The elimination of the redundant instances is taken care of by the fact that tokens can only move “downward” within the cascade. If tokens could move both up and down in the reservoir cascade then there could be a token path that goes from Reservoir 1, to 2, to 3 and a token path that goes from Reservoir 3, to 2, to 1. Both of these data strings correspond to a facility being connected to all three reservoirs. The structure of the cascade forces the former token path to be only one of six possible combinations to represent this connection scheme. The connection structure within the block, which allows tokens to enter and exit the cascade from any position in the cascade, and the fact that tokens may pass through one or more reservoirs within the cascade, contribute to the block's ability to generate all possible connection instances.

The total number of connection instances for one facility to a set of reservoirs with no constraints is easy to calculate. It is $2^n - 1$, where ‘n’ is the number of reservoirs. The ‘minus one’ term is included because the connection instance where the facility is not connected to any reservoir is ignored and not generated by the block. An example would be that for one facility and five reservoirs, the total number of combination possibilities is $2^5 - 1 = 31$. With OPN, it is quick and simple to count the number of instances generated by a one-block model, and implementing a model for the above example generates 31 instances. Tests of this nature were performed throughout the development of this generator to check for bugs and to ensure all possible combinations were being generated.

Expanding a Block

Expanding a block to have more reservoirs requires an additional OPN process representing the additional reservoir. It also requires the addition of n+1 connections. Considering the fact that each connection has some Boolean coding attached to it (for constraint purposes explained later in section 3.3.3) and the fact that OPN currently does not have any “copy and paste” function, the addition of more reservoirs becomes increasingly tedious. This issue compounded by the fact that it must be done all over again for the addition of reservoirs in other blocks in the model.

3.3.2 Multiple Blocks

Since one block only generates connection possibilities for one row in the connection matrix, additional blocks must be added in order to complete the matrix. Essentially all the blocks in the model are the same. The one small difference is the block or facility number indices, which describe for which row in the connection matrix the block is responsible. In the model shown in Figure 3-6, there are three blocks. Block 1 would generate all the combination possibilities for Row 1 of the connection matrix. Block 2 and Block 3 would do the same for Rows 2 and 3, respectively.

Combination of combinations

Now that we have a model that generates each of the possible connection combinations for each row of the connection matrix individually, these blocks must be connected so that all the possible connection combinations for the whole system are generated. For example, in the model shown in Figure 3-6, there are three facilities to be connected to a set of three reservoirs. With no constraints, each block will generate $2^3 - 1 = 7$ possible connection schemes for each facility. For the whole system of three facilities and three reservoirs then there are $7^3 = 343$ total possible connection schemes. This is because each instance of a connection scheme for one row in the matrix could occur in combination with each of the possibilities in the other two rows.

This generation of the set of possible row combinations is handled in this OPN model by putting the blocks in series. For example, in the model in Figure 3-6, as the tokens reach the Block 1 End object, there are seven tokens, each with a different set of row-one values in their connection matrices representing the seven different connection schemes for Facility 1 and a set of three reservoirs. Each of these seven tokens has “blank” rows 2 and 3 in their matrices. Now, as each token enters Block 2 to have its second row assigned, each token entering Block 2 generates seven tokens exiting the block. Therefore, a token with one particular combination instance of Row 1 enters Block 2. It will generate seven tokens at the end of Block 2, each with that one particular instance of Row 1. Additionally, each of the seven tokens will have one of the seven possible combination schemes for Row 2. Thus, seven different instances entering Block 2 from Block 1 will generate $7 * 7 = 49$ instances leaving Block 2. Furthermore, 49 instances entering Block 3 from Block 2 will generate $49 * 7 = 343$ instances leaving Block 3, and so on for more blocks. This series arrangement of blocks ensures all possible combinations for the whole system are generated.

The general relation for the total number of connection schemes generated by this N by M generator, where N equals the number of columns in the matrix and M equals the number of rows in the matrix is:

$$(2^N - 1)^M$$

Additional model items

The model has two additional items beyond the set of blocks in series. The first is the “Generator Start”. It initializes the model and sends the initial token into the first block. The last item is the “Generator End”, which collects all the tokens that have made it through the whole model.

Expanding the model with more blocks

Expanding the model to add more facilities requires the addition of more blocks to the series. Each nth block added would be a copy of the existing blocks, with the index for the facility or row it is responsible for changed to “n” in all the new block’s elements. This block will then be connected in series at the end of the chain of existing blocks and to the “Generator End”. With the current version of OPN without copy and paste, this is a tedious but simple task.

3.3.3 The Addition of Constraints

With a model that now generates possibly millions of connection schemes, it is essential that constraints are added to limit the number of solutions to large problems. The addition of constraints into this model is important for two reasons. First, since the number of combination schemes generated by the model is exponential with the number of facilities and reservoirs, there must be constraints in larger problems in order to ensure that OPN does not run out of memory resources while trying to generate 10,000 or more instances, and so that the model runs to completion in a reasonable amount of time. For example, a 5x5 connection matrix with no constraints has more than 28 million $((2^5-1)^5)$ possible connection schemes, which is many times more than OPN can handle currently. More importantly, constraints allow the user to tailor the model to generate connection schemes that are possible in the real problem. This model currently allows the user to set five types of constraints:

- Total number of facilities
- Maximum number of reservoirs
- Maximum number of reservoirs per facility
- Maximum number of facilities per reservoir
- Specific non-compatible reservoirs

All constraints are set before running the model in the "Global Script" section of OPN. The code for the entry of constraint settings in the model is shown in Figure 3-7.

```
#Sets total number of facilities 1-5
Fmax=3
#Set max number of Reservoirs 1-5
Rmax=5
#sets max number of reservoirs for @ facility
1-5
F1max=2
F2max=2
F3max=2
F4max=1
F5max=1
#sets max number of facilities for @ reservoir
1-5
R1max=1
R2max=1
R3max=1
R4max=1
R5max=1
#sets reservoirs that can't go together
# example r1r2=1 (res1 and res2 can't go
together)
# example r2r3=0 (res2 and res3 can go
together)
r1r2=0
r1r3=0
r1r4=0
r1r5=1
r2r3=0
r2r4=0
r2r5=0
r3r4=0
r3r5=0
r4r5=0
```

Figure 3-7. Constraint Entry Code in Global Script

Total number of facilities

For the first constraint, the user may set the total number of the facilities to be connected. This essentially sets the number of blocks to be used by the generator. This can also be viewed as setting the number of rows in the connection matrix. The current generator has a total of five blocks, so the user can select from 1 to 5 total facilities.

This constraint is implemented by creating a variable called $Fmax$ and assigning a value to it in the Global Script, which can be changed by the user to set the total number of facilities. This variable is then used within the model to restrict tokens passing from one block to the next and from blocks to the generator end. Tokens may only move from one block to the next if the value of $Fmax$ is greater than the index of the block it is leaving. For example, if $Fmax=2$ then tokens are allowed to move from Block 1 to Block 2, but not from Block 2 to Block 3. Also, tokens are only allowed to pass from a block to the "Generator End" if the index of the block they are leaving is equal to the value of $Fmax$. Tokens are restricted or allowed to pass using Boolean statements on the connection arrows leading to elements in OPN.

Maximum number of reservoirs

The user can set the maximum number of reservoirs to be considered by the model. This constraint essentially sets the number of columns in the connection matrix. The current generator has a total of five reservoirs in each block, so the user can select from 1 to 5 as the maximum number of reservoirs.

In the model, this constraint is implemented by creating a variable called $Rmax$ and assigning a value to it in the Global Script, which can be changed by the user to set the maximum number of reservoirs. This variable is then used within the model for restricting tokens from passing to reservoirs with an index higher than $Rmax$. This is done identically in each block by writing a Boolean expression on all the connections within the reservoir cascade that connect one reservoir to another. In the current model, by setting $Rmax=4$ all the Reservoir 5's in each block are not used.

Maximum number of reservoirs per facility

The oil problem described above required that a constraint be created, where the user could set the maximum number of reservoirs that each facility could be connected to. Physically this constraint could be required because a facility only has enough oil production capacity for a given number of reservoirs. For the connection matrix, this sets the maximum number of "x's" allowed in each row. This model allows the user to set each facility's maximum number of reservoirs individually.

The implementation of this constraint is done by creating counting variables called: $F1cnt$ for Facility 1, $F2cnt$ for Facility 2, etc. These counters are zeroed at the beginning of their respective blocks. Then within each block its specific counter counts the number of reservoirs a token has passed through while in that block. Boolean statements on the connections leading into each reservoir restrict tokens that have already connected to the maximum number of

reservoirs allowed for that block. This essentially sends the token to the block end instead. The maximum number of connections for each facility is set by a variable in the Global Script called: *F1max* for Facility 1, *F2max* for Facility 2, etc.

Maximum number of facilities per reservoir

In the multi-reservoir, multi-facility problem one of the constraints that needed to be able to be set was the maximum number of facilities to which a reservoir could be connected. Physically this could be because a reservoir may only contain enough oil to be processed by a certain number of facilities. In terms of the connection matrix, this constraint sets the maximum number of "x's" in each column. In this model, this constraint is implemented such that the maximum number of facilities can be set for each individual reservoir.

This constraint is implemented by creating counting variables on each token. There is one counter for each reservoir. These are called *R1cnt* for Reservoir 1, *R2cnt* for Reservoir 2, etc. All of the counting variables are set to zero as the generator initializes. Then, as a token passed through a reservoir, its *Rxcnt* variable is advanced by one. (*Rxcnt* is the counting variable for Reservoir "X".) For example, as a token is passed through Reservoir 4, in any of the blocks, it adds one to the existing *R4cnt* value of the token. Each counter keeps track of how many facilities its corresponding reservoir has been connected to thus far in the model. The connecting arrow leading into each reservoir has a Boolean statement that restricts any token from passing that has already had the maximum number of connections with the reservoir the token is trying to enter. This essentially sends that token to a different reservoir or to the end of the block. The maximum number of connections for each reservoir is set by variables in the Global Script called: *R1max* for Reservoir 1, *R2max* for Reservoir 2, etc.

Specification of non-compatible reservoirs

In the oil problem, there may be a case where two reservoirs are at such a distance from each other that one would never want to connect them with the same facility; or the two reservoirs may have very different types of oil or gas in them, such that one could not process them at the same facility. In the connection matrix, this constraint states that there can never be an "x" in both the Reservoir A column and Reservoir B column for any row. For a system with "n" reservoirs (or columns) there are $2n-1$ individual pairs of reservoirs that may not be allowed to connect to the same facility. Thus, for this model with five reservoirs there are nine pairs of reservoirs that may not be allowed to go together.

The implementation of this constraint is done by creating flag variables in the Global Script. They can be seen at the bottom of the code in the Global Script in Figure 3-7. There are nine

flag variables, each corresponding to the nine different pairs of reservoirs. Flag variable *r1r2* corresponds to the pair Reservoir 1 and Reservoir 2, and *r3r5* corresponds to the pair Reservoir 3 and Reservoir 5, for example. By setting the flag variable *r1r2*=1, Reservoir 1 and Reservoir 2 cannot be connected to the same facility. Leaving a flag variable equal to zero means the corresponding two reservoirs can be connected to the same facility. The connections between the reservoirs in the cascade use these flag variables within the model. There are Boolean statements on these connections that restrict tokens entering reservoirs if the token has already passed through a non-compatible reservoir within that block. A screenshot of the complete N by M connection generator in OPN is shown in Figure 3-8.

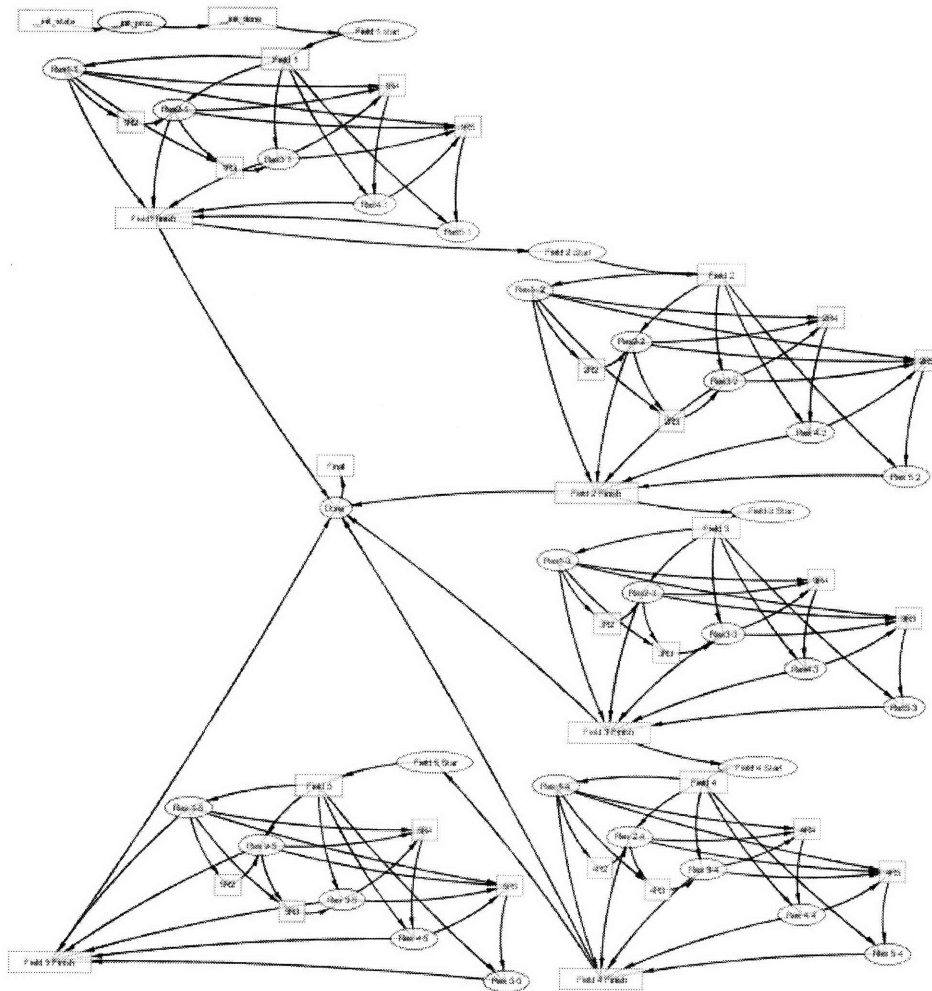


Figure 3-8. Screenshot of the Actual OPN N by M Connection Generator

3.4 Output

The model will direct, to the generator end, all the tokens that have a complete instance of a connection scheme that satisfies the constraints. Each of those tokens contains the variables

that represent all the intersections of the rows and columns of the connection matrix. Each will equal either zero or one. A “Zero” indicates that there is no connection at the intersection of that column and row (reservoir and facility), and “one” shows that there is a connection. These “ones” will be assigned by passing through the particular reservoir in the particular block corresponding to that column and row. The labeling convention for these variables is as follows: R31 corresponds to the connection of Reservoir 3 and Facility 1 (or in the matrix column 3 and row 1) and R25 corresponds to the connection of Reservoir 2 and Facility 5, and so forth. In addition, each token that reaches the end will have the counting variables discussed above. Essentially, these tokens now have a vector containing all of these variables and data about the connections.

In this OPN model, the first quick check is to view the list of tokens that have reached the “Generator End” object. It is there that the user can read how many tokens have arrived. This should equal the total number of possible connection schemes. The user can also graphically view the path each token took and read the output vector described above for each token.

3.4.1 Exporting to Excel

From the OPN object marking the “Generator End”, one click will export the tokens to an Excel spreadsheet. (This is convenient for sorting and filtering the outputs.) Within the exported spreadsheet, there will be a header row that names all of the columns, each of which represents a variable in the output vector on each token. The following rows each represent one instance of a connection possibility generated by the model. Instructions for running the model and retrieving these outputs are found in Appendix 7. To create a connection matrix from one row in the spreadsheet, one would delete or ignore the counting variables (all of which have a header label ending in “cnt”). One would then move (probably via MATLAB or an Excel macro) the elements of the vector into their corresponding positions in the connection matrix so that it looks like Figure 3-3. An example output spreadsheet for a 3-facility by 2-reservoir matrix is shown in Figure 3-9. The complete set of connection schemes output from the model for an unconstrained 3-by-2 model is shown in Figure 3-10.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	R11	R21	R31	R41	R51	R12	R22	R32	R42	R52	R13	R23	R33	R43	R53	R14	R24	R34	R44	R54	R15	R25	R35	R45	R55
2	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
3	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0
4	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1
5	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
6	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0
7	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0
8	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1

Figure 3-9. Example Output Spreadsheet

1	R1	R2	2	R1	R2	3	R1	R2	4	R1	R2	5	R1	R2	6	R1	R2	7	R1	R2	8	R1	R2	9	R1	R2
F1	1	0	F1	1	0	F1	1	0	F1	1	1	F1	1	0	F1	1	1	F1	1	0	F1	1	0	F1	1	1
F2	1	0	F2	1	0	F2	1	1	F2	1	0	F2	1	1	F2	1	0	F2	0	1	F2	1	0	F2	1	1
F3	1	0	F3	0	1	F3	1	0	F3	1	0	F3	0	1	F3	0	1	F3	1	0	F3	1	1	F3	1	0
10	R1	R2	11	R1	R2	12	R1	R2	13	R1	R2	14	R1	R2	15	R1	R2	16	R1	R2	17	R1	R2	18	R1	R2
F1	1	1	F1	0	1	F1	1	0	F1	1	1	F1	1	1	F1	1	1	F1	0	1	F1	0	1	F1	1	0
F2	0	1	F2	1	0	F2	1	1	F2	1	0	F2	0	1	F2	1	1	F2	0	1	F2	1	1	F2	0	1
F3	1	0	F3	1	0	F3	1	1	F3	1	1	F3	0	1	F3	0	1	F3	1	0	F3	1	0	F3	1	0
19	R1	R2	20	R1	R2	21	R1	R2	22	R1	R2	23	R1	R2	24	R1	R2	25	R1	R2	26	R1	R2	27	R1	R2
F1	1	0	F1	0	1	F1	0	1	F1	0	1	F1	0	1	F1	1	1	F1	0	1	F1	0	1	F1	1	1
F2	0	1	F2	1	1	F2	1	0	F2	1	0	F2	0	1	F2	0	1	F2	1	1	F2	0	1	F2	1	1
F3	1	1	F3	0	1	F3	0	1	F3	1	1	F3	0	1	F3	1	1	F3	1	1	F3	1	1	F3	1	1

Figure 3-10. Complete Output for an Example 3-by-2 Connection Matrix

3.5 Expansion Possibilities

There are several foreseeable possibilities for future work on this model that would expand its capabilities and utility. They include changes to the model itself and updates to the OPN program. This model could also be made to do larger than a 5-by-5 connection matrices. Finally, there could be additional constraints added to the model, like minimum number of facilities and reservoirs to be connected.

3.5.1 Other Potential Constraints

There are a few conceivable constraints that could be added to this model that were not needed for the initial oil problem, but that could be needed for other problems.

Maximum number of facilities used

The first would be a constraint that sets the maximum number of facilities used, as opposed to the current model, which strictly sets the total number of facilities connected. For the connection matrix, this would allow generation of connection schemes that have one, two or more rows up to a set maximum. This could be implemented by simply changing the Boolean statements on the connections between the block ends and the generator end. Currently these statements only allow the passage of tokens that have been through a specific number of facilities. By changing these Boolean statements to an inequality, the model would generate additional connection schemes that have used fewer facilities for connection. Essentially, this would allow completed tokens to reach the generator end from several block ends instead of just one.

Minimum number of facilities used

The addition of a constraint that sets the minimum number of facilities used would only be a slight variation of the previous constraint. The addition of this constraint would require the

creation of one more variable in the Global Script to set this value and slight changes to the inequalities on the connections between the block ends and the generator end.

Minimum number of facilities per reservoir

A user may want to specify a minimum number of “x’s” allowed in a column (the current version allows the user to specify a maximum). This would translate into stating a minimum number of facilities to which a reservoir could be connected. This could be done on an individual reservoir basis. The constraint could be implemented by creating minimum variables for each reservoir in the Global Script. This minimum variable and the *Rxcnt* variables, which count the number of times a token has passed through a reservoir “x”, could then be used in the Boolean statements on the connections between the block ends and the generator end to restrict tokens that have not passed through each reservoir enough times.

Minimum number of reservoirs per facility

Finally, a user may want to set a minimum number of reservoirs for each facility. This would be a minimum number of “x’s” in each row of the matrix (the current version allows the user to set a maximum). This constraint could be implemented by creating a minimum variable for each facility in the Global Script and then using this minimum variable and the *Fxcnt* variables on the connections between the reservoir cascades and the block ends to restrict tokens that have not passed through enough reservoirs within each block.

3.6 Use of Metrics with the Generator

This generator could also be used separately to screen connection schemes with the addition of iso-performance metrics. An example problem was developed to illustrate this point. In this problem there are three facilities and four reservoirs that are to be connected by some to-be-determined scheme. The facilities and reservoirs are laid out on a 10km by 10km grid shown in Figure 3-11.

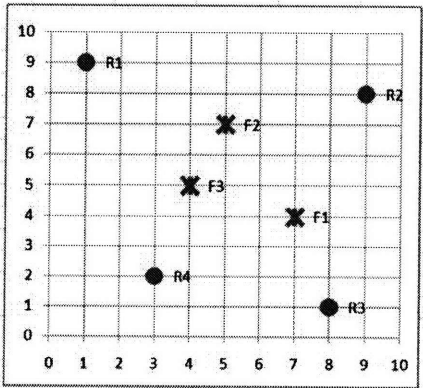


Figure 3-11. Example 3X4 Connection Layout

Using the connection generator, 2,401 connection schemes were generated for this problem where at least one facility was used and all the reservoirs had at least one connection to a facility. After calculating the distance between each facility and each reservoir, a length of pipe can be assigned to each connection in the connection matrix. A simple cost formula that relates the cost to the total length of pipe for a connection scheme and the total number of facilities used in that scheme was developed for illustrative purposes.

Connection Scheme Cost

$$= C_p * \text{total length of pipe} + C_f * \text{total number of facilities used}$$

where $C_p = \$7\text{million/km pipe}$ and $C_f = \$50\text{million/facility used}$. The values for both C_p and C_f were chosen as simple representative values for this example.

Using this cost calculation method and post processing of the generated connection schemes in a spreadsheet, the cost was calculated for each connection scheme. Figure 3-12 shows a plot of the cost for each of the 2,401 schemes. The upper (light-blue) bars show the cost for each scheme (with the scale on the left), while the lower (dark-red) bars show the number of facilities used for each scheme (with the scale on the right). Figure 3-13 shows the same information, but only for the 81 connection schemes in which each reservoir is connected to only one facility.

Using this rough screening method, one could pick out the best single or handful of connection schemes for more detailed analysis. For example, in Figure 3-12 the preferred connection schemes can be seen distinctly on the left side of the graph where the cost starts to drop more sharply as you move to the left. This distinction of the preferred set of connection schemes is less apparent in the case represented by Figure 3-13, which has a much smoother graph. The preferred set in this case would simply be the top 5 or 10 schemes, with the number of schemes in that set being any reasonable number (i.e. top 5, top 10, or top 12). Figure 3-14 shows the top three connection schemes with the lowest cost for the case represented in Figure 3-12. Both their connection matrices and illustration of their connection layout are shown. For contrast, the same is also shown for one of the most expensive connection schemes.

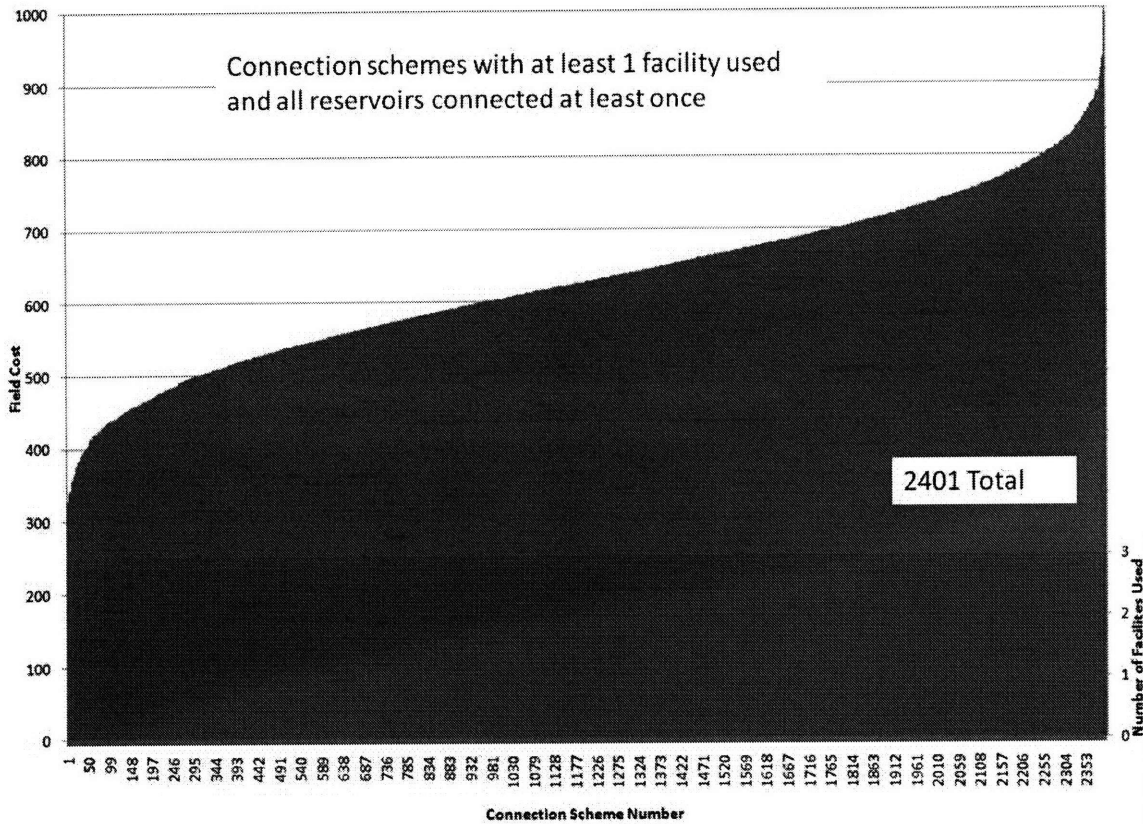


Figure 3-12. Metrics Plot of 2401 Connection Schemes for 3X4 Example

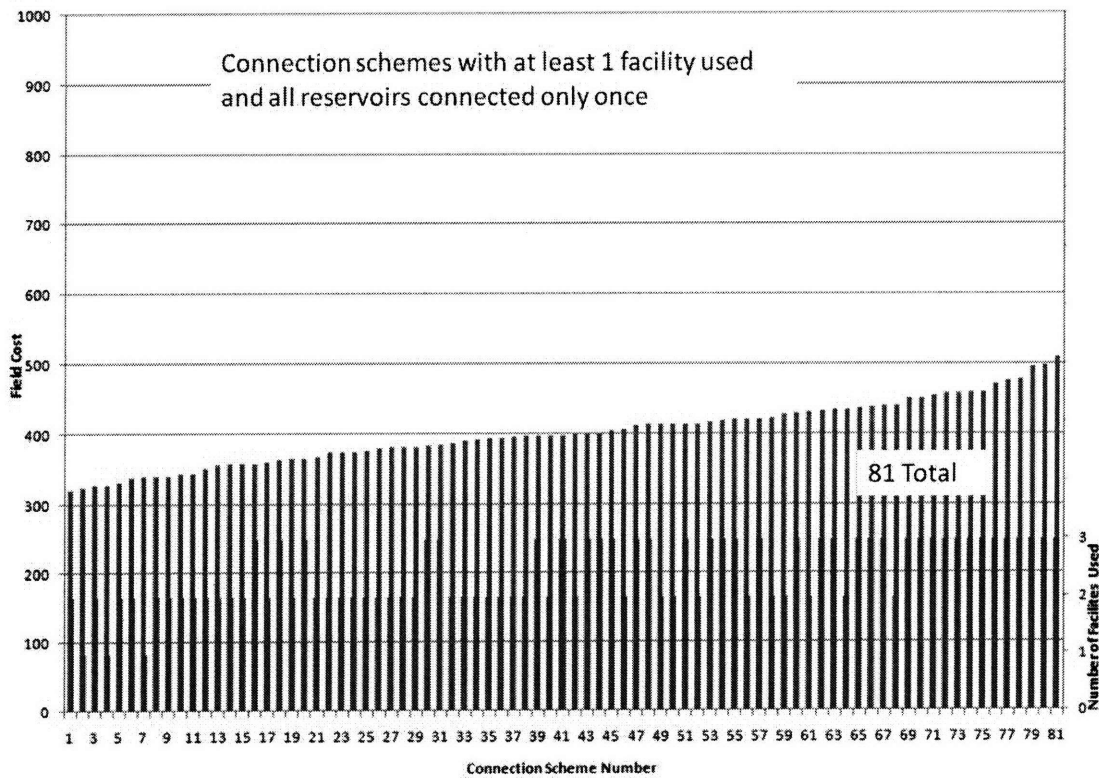


Figure 3-13. Metrics Plot of 81 Connection Schemes for 3X4 Example

1-\$321	Reservoir 1	Reservoir 2	Reservoir 3	Reservoir 4
Facility 1		x	x	
Facility 2				
Facility 3	x			x

2-\$325	Reservoir 1	Reservoir 2	Reservoir 3	Reservoir 4
Facility 1				
Facility 2				
Facility 3	x	x	x	x

3-\$327	Reservoir 1	Reservoir 2	Reservoir 3	Reservoir 4
Facility 1			x	x
Facility 2	x	x		
Facility 3				

2392-\$914	Reservoir 1	Reservoir 2	Reservoir 3	Reservoir 4
Facility 1	x	x	x	x
Facility 2	x	x	x	x
Facility 3	x	x		x

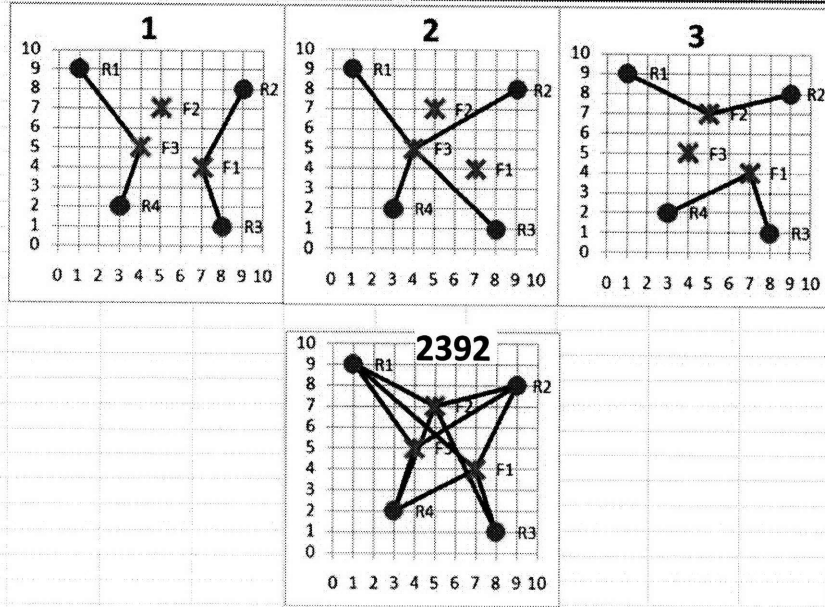


Figure 3-14. Four Connection Schemes for 3X4 Example (connection matrices and illustrations)

3.7 Summary

In summary, this chapter discussed the development of an OPN model that can be used generically to generate all possible links between two sets of elements. The model generates all the possible connection matrices, given two sets of elements to be linked and constraints about their linkage. The model uses a blocked-cascading concept to generate all possible connection schemes and limits redundant instances by its structure. There are various restrictions that a user can set to limit the number of connection schemes generated and to meet real-world constraints. Additional constraints could be implemented in the future if needed. The model discussed was developed for a specific oil project, but it is quite conceivable that this model could be used as one of several general tools for building system architecture models with OPN. One could conceive of a time where someone building an OPN model recognizes the need for a “connection generator” and that person could go to a toolbox to insert one into their model.

CHAPTER 4. Expanded Multi-Reservoir, Multi-Facility Connection Generator (N plus M Connection Generator)

4.1 Introduction

This chapter discusses an additional connection generator model built in Object Process Network (OPN)(12)(10)(14)(15). This generator is different from the connection generator in Chapter 3. Instead of connecting two sets of things that do not have connections between elements within each set, this model generates all of the connections between all elements. It is called an N plus M connection generator because it takes the N rows and M columns of the N by M generator and puts them in a square matrix that has a row and column number of N+M.

This chapter first reminds the reader of the motivation for the development of the N by M connection generator and then discusses the need that arose for a more general N plus M Generator. Next, it explains the concept for this OPN model. Then, the chapter steps through the implementation of this generator as an OPN model. Next, it discusses the addition of constraints to the generator. This is followed by a discussion on the output format and procedures for the generator. The chapter finishes with a discussion of the limitations and expansion possibilities of the model.

4.2 Motivation

4.2.1 The Problem

After some use of the generator discussed in Chapter 3, it became evident that there was another class of connection problems. This class of problems is just a more general and expanded version of the first. This problem was posed in the same research effort as that discussed in 3.2.1.(21) In that work with BP, we found that in many cases it would be useful to generate the connections between a set of facilities and reservoirs, as before, but additionally it would be beneficial to generate connections between facilities and connections between reservoirs. In general this is a problem were every element could be connected to every other element.

The original problem in 3.2.1 is represented by an N by M matrix as in Figure 3-1. The expanded problem is represented by the N plus M matrix in Figure 4-1. It is essentially an N-squared matrix. The black positions along the diagonal are the positions that would represent the connection of an element to itself, but they are blackened because in this problem this self connection is meaningless. In this case we do not have any concept related to directionality which would cause the need for detailed examination of both the upper and lower triangles in

the matrix. Instead, we only care if the elements are connected or not. Thus only the upper or the lower triangle is needed to convey this information. I have chosen to use a lower triangular representation for reasons that will become apparent later in this chapter. Figure 4-1 is a matrix representation of a connection scheme depicted as an example in Figure 4-2. The “X’s” indicate a connection between the elements in the corresponding row and column. Blanks in the matrix indicate no connection.

Figure 4-2 is the same connection scheme as Figure 3-2 in the previous chapter, but with two added connections. These two connections are examples of the two additional connection types this generator handles; connections between facilities and connections between reservoirs. As stated earlier in this section, this N plus M connection problem is just an expansion of the N by M problem. This can be seen in the N plus M connection matrix in Figure 4-1. The N by M connection matrix can be seen embedded in this N plus M matrix. It is the 3 by 4 rectangle in the lower left corner of the matrix where the facilities of the columns meet the reservoirs of the rows. The additional connection information about the two new types of connections is contained in the smaller 2 by 2 and 3 by 3 triangles that form the points of this N plus M matrix. The top 2 by 2 triangle contains information about connections between facilities and the lower right triangle contains the information about the connections between reservoirs.

	F1	F2	F3	R1	R2	R3	R4
Facility 1- F1							
Facility 2- F2							
Facility 3- F3		X					
Reservoir 1- R1		X					
Reservoir 2- R2	X		X				
Reservoir 3- R3			X				
Reservoir 4- R4				X			

Figure 4-1. Example N+M Connection Matrix

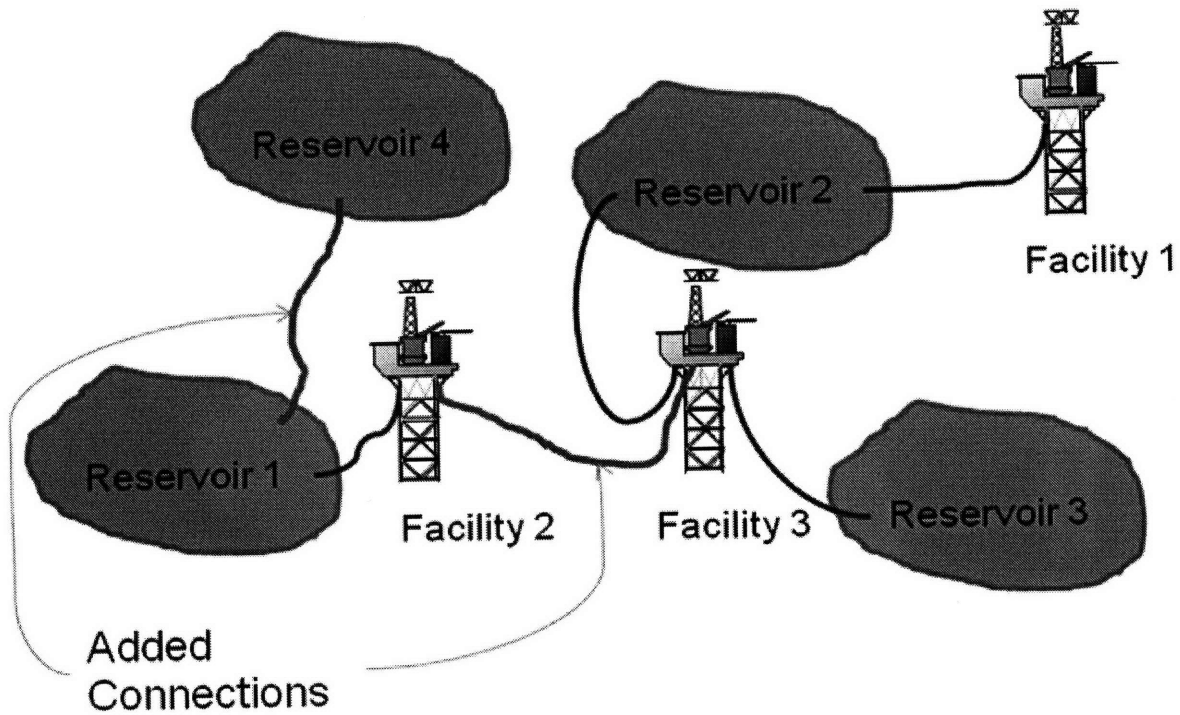


Figure 4-2. Example N+M Connection Illustration

4.3 The Concept

The concept for the solution of this problem is very similar to the blocked-cascading concept used in the N by M generator. This problem could also use a version of the looping-indexing concept discussed in 3.2.3 but it did not because of the same reasons outlined at the end of 3.2.3. An illustration of the concept for this N plus M problem is shown in Figure 4-3 (which is not an OPN). The block structures used are similar to those in the concept for the N by M problem (Figure 3-6). Each block again is responsible for generating connections for a specific row in the connection matrix. However, in this concept the number of reservoirs in the cascade is increased by one from one block to the next. This follows the form of the lower triangular connection matrix where each row has one more column than the one above. The cascade in each block still has the same structure as in the N by M concept. This structure ensures all of the possible connections for a row are generated and avoids any redundancy of connections. This N plus M concept also has the addition of a null process that allows a path in each block around the cascade that generates no connections. This adds the ability of this model to generate connection schemes where individual elements are not connected at all, including the one instance where none of the elements have a connection.

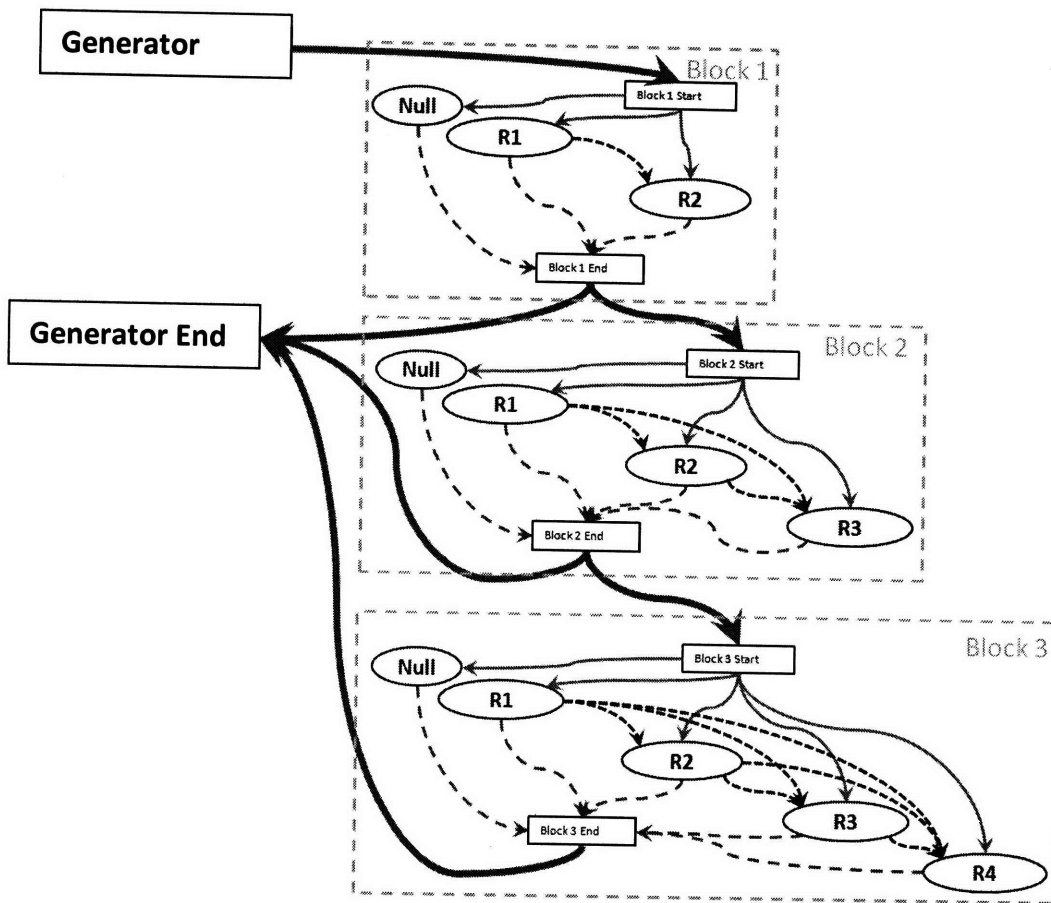


Figure 4-3. N+M Model Concept

The example N plus M model concept shown in Figure 4-3 leaves out the actual first block, which is in the model, but is not shown because it is so simplistic. This first block has a block start and end object as the rest do. It has one "reservoir" and a null process. This first block generates the connection possibilities for the first row in the matrix that only has one column. The connection possibilities are simply connected or not connected.

In this concept for larger and larger problems, blocks are simply added to the end of the model, each with one more reservoir in the cascade than the last. This idea of starting with smaller blocks and adding larger and larger blocks for larger problems is what led to the use of the lower triangle in the connection matrix as opposed to the upper triangle. If the upper triangle was used with this same concept the first block would have to be of varying size depending on the size of the problem. By using the lower triangle, every problem uses the same first, second and third blocks no matter the size of the problem. The model could be built with a certain number of blocks for the largest possible problem and then the larger blocks could simply be ignored and bypassed for smaller problems. This allows the speed of the generator to be inversely proportional to the size of the problem.

4.4 The Implementation

4.4.1 Each Block

Each block in the N plus M model is structured and operates similar to the blocks in the N by M model (see 3.3.1). There is a block start and block end and a cascade of “reservoirs” as before. The “reservoirs” in this model, however, no longer represent oil reservoirs or, in a more general sense, individual elements of one type. Instead, in the N plus M model the “reservoirs” in a cascade represent an individual element that is of any type. More specifically, a reservoir represents the connection of one column in the matrix. In this way all reservoirs named R2 in Figure 4-3 represent the connection of the second column in the matrix with each row of the matrix (each block in the model).

In N plus M model there is also a null process in each block which gives a path around the reservoir cascade that does not assign any connections. This allows the possibility of no connections as an instance of a connection scheme for each block and thus each row. This was necessary because all of the connections of one element no longer reside in one row or one column. In the N plus M matrix, all of the connections for one element are represented in an “L” shape with the 90-degree angle of the “L” on the main diagonal of the N plus M matrix (see Figure 4-5). This holds for every element in the matrix except the first and the last elements, whose connections are completely represented in the first column and last row respectively. Since connections for most elements do not completely lie in one block in the model, it is possible that in a block all of the connections it represents are restricted individually and that the allowed connections for that element lie as a column in another block (or another row). As these blocks are connected in series, it is necessary to provide a path around a reservoir cascade that may be entirely restricting the entry of a token. In addition, in a real-world sense, the connection instance where an entire row in the connection matrix is unconnected may be a real and feasible possibility.

Each block is initialized in its Block Start process. In the initialization the variables that represent each position in the connection matrix in the row that corresponds to that block are created and are set equal to “zero”. This produces the blank (unconnected) canvas on which the block will write the different connection schemes for the corresponding row. Also in the initialization of a block, the counter variable, which represents the number of connections made to the element to which the row of the block corresponds, is created and set to “zero”. Since, as one moves down the rows in the matrix, each row has the first connections for the element that row represents, there is no need for a counter for that element before its

corresponding row. That counter will be used and possibly advanced in the block in which it is created and every block thereafter for implementing constraints, which will be discussed in a section later in this chapter.

4.4.2 Blocks in Series

As in the N by M model, blocks are placed in series, where all of the connection schemes generated by the previous block seed the following block. In addition each block has one more reservoir in its cascade than the previous block, to mimic the lower triangular connection matrix, where each row is one column longer than the row above it. In this way the model can represent connection matrices of increasing size by allowing token flow through more and more blocks. Every connection matrix of $N+M \geq 2$ will use the first block, and for each step up in the size of $N+M$ from $N+M=2$, one more block will be added. In the complete model (a screen shot of which is shown in Figure 4-4), there are a total of 12 blocks which means the model is able to represent connection matrices up to $N+M=13$.

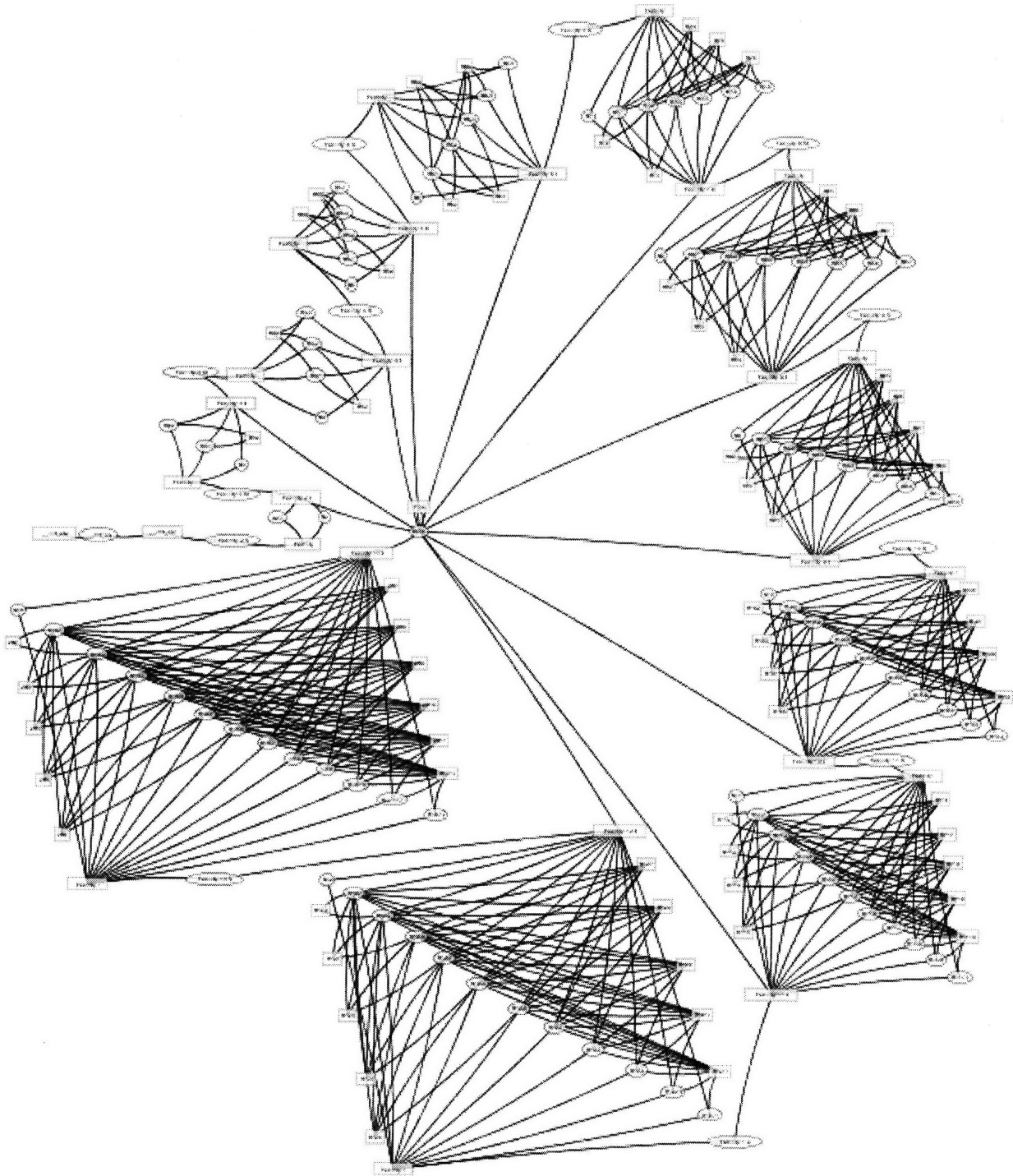


Figure 4-4. Screenshot of OPN N+M Connection Model

4.4.3 Other Elements in the Model

There are two sets of other elements in the N plus M model, as were in the N by M model. They are the same two elements as described in Section 3.3.2. These are essentially the generator start processes and objects and the generator end processes and objects. The

generator start elements generate the initial seed token and send it into the model. The generator end elements collect the finished tokens for analysis.

4.4.4 Combinations of Combinations

The combinatorics of this N plus M generator are similar to those of the N by M generator discussed in section 3.3.2. The different form of the connection matrix adds the additional connection types in the oil problem (connections between facilities and connections between reservoirs). This increases the possible number of connection schemes for a given set of facilities and reservoirs. The addition of the null reservoirs in each block also adds connection schemes which involve instances where an element is not connected at all. These two reasons increase the total number of connection scheme possibilities of an unconstrained problem from the N by M case of $(2^N - 1)^M$, where N equals the number of columns in the matrix and M equals the number of rows in the matrix. In the N plus M generator, the total number of possibilities for an unconstrained problem is represented by the following relation:

$$2^{\binom{n^2-n}{2}}$$

where $n=N+M$ or the total number of elements being considered for connection. As an example, for an unconstrained situation with five facilities and five reservoirs, there are about 2^{45} (or 35×10^{12}) connection schemes in the N plus M case. There are about 29×10^6 possibilities in the N by M case. The fact that the N plus M case has more possibilities for the same set of elements means that in order for the model to run in a reasonable amount of time (or at all) it will have to be more constrained, or reduced to a smaller problem.

4.4.5 Constraints

Once again it is important that this generator enable the use of the constraints on a problem for two reasons. First, the exponential nature of the size of the problem and the number of possible connection schemes require constraining the problem in order for it to be run at all from a computational standpoint. Second, the real world situations which these problems represent have legitimate constraints on their ability to connect different elements. The current model has three types of constraints that the user can set:

- Total number of elements
- Maximum number of connections per element
- Specific non-compatible elements (including a sub-set based on distance between elements and some maximum allowable distance)

These constraints are set in the generator by changing values of different variables in the Global Script section of the OPN model. The code for this is shown in Appendix 8.

Total number of elements

For the first constraint, the user may set the total number of elements to be connected. Doing so essentially sets the number of blocks to be used by the generator. This can also be viewed as setting the number of rows in the connection matrix. The current generator has a total of 12 blocks. So the user can select from 2 to 13 total elements.

This constraint is implemented by creating a variable called *Fmax* and assigning a value to it in the Global Script, which can be changed by the user to set the total number of elements. This variable is then used within the model to restrict tokens passing from one block to the next and from blocks to the "Generator End". Thus, tokens may only move from one block to the next if the value of *Fmax* is greater than the index of the block it is leaving. For example, if *Fmax*=2 then tokens are allowed to move from Block 1 to Block 2, but not from Block 2 to Block 3. Also, tokens are only allowed to pass from a block to the Generator End if the index of the block they are leaving is equal to the value of *Fmax*. Tokens are restricted or allowed to pass using Boolean statements on the connection arrows between elements in OPN. This is the same implementation for the total number of facilities as used in the N by M generator (see section 3.3.3).

Maximum number of connections per element

In the expanded multi-reservoir, multi-facility problem, one of the constraints that needed to be able to be set was the maximum number of connections allowed for each element. Physically this could be because a reservoir may only contain enough oil to be processed by a certain number of facilities, or because a facility only has enough oil production capacity for so many reservoirs. In terms of the connection matrix, this constraint sets the maximum number of "x's" for an element. As previously discussed, for most elements all of the positions in the matrix associated with the connection to a single element form an "L" shape, as shown in Figure 4-5. In this model, this constraint is implemented such that the maximum number of facilities can be set for each individual element.

	R1	R2	R3	R4	R5	R6	R7
Element 1 -R1							
Element 2 -R2	0						
Element 3 -R3	1	0					
Element 4 -R4	0	0	1				
Element 5 -R5	0	0	1	0			
Element 6 -R6	1	1	0	1	0		
Element 7 -R7	0	0	1	0	1	0	

Element 5 connection positions

Figure 4-5. Connections for a Single Element

This constraint is implemented by creating counting variables on each token. There is one counter for each element. These are called *R1cnt* for Element 1, *R2cnt* for Element 2, etc. All the counting variables are set to “zero” as the first block involved with the connection of that element initializes. Then, as a token passes through a “reservoir”, its *Rxcnt* variable is advanced by one. For example, as a token is passed through an R4, in any of the blocks, it adds one to the existing *R4cnt* value of the token. In addition, since each block handles the connections of a single element to multiple other elements across the columns, when a token passes through a “reservoir” in a block, the element counter for the single element being handled by that row is also advanced by one. For example, as a token passes through the R4 “reservoir” process in the block corresponding to Element 5’s row in the matrix, *R4cnt* and *R5cnt* are both advanced by one. This is because the single connection of Element 4 and Element 5 is one instance of a connection for Element 4 and for Element 5. Each counter keeps track of how many elements its corresponding element has been connected to thus far in the model. The connecting arrow leading into each “reservoir” has two Boolean statements that restrict tokens from passing that already have the maximum number of connections to the elements for the block (row) that it is currently in and with the “reservoir” it is trying to enter. This essentially sends that token to a different “reservoir” or to the end of the block. The maximum number of connections for each element are set by variables in the Global Script called: *R1max* for Element 1, *R2max* for Element 2, etc.

Specification of non-compatible reservoirs

In the oil problem, there may be a case where two elements are at such a distance from each other that one would never want to connect them; or the two elements may have very different types of oil or gas in them, such that one could not process them both at a particular facility. In the connection matrix, this constraint states that there can never be an “x” in a

particular position in the matrix corresponding to the connection of Element A and Element B. For a system with “n” elements, there are $\frac{1}{2} * (n^2 - n)$ individual pairs of elements that may not be allowed to go together. Thus, for a model with 13 elements, there are 78 pairs of elements that may not be allowed to go together.

The implementation of this constraint is done by creating flag variables in the Global Script. They can be seen at the bottom of the code in the Global Script in Appendix 6. There are 78 flag variables, each corresponding to the 78 different pairs of elements. Flag variable r_{1r2} corresponds to the pair Element 1 and Element 2 and r_{3r5} corresponds to the pair Element 3 and Element 5, for example. By setting the flag variable $r_{1r2}=1$, Element 1 and Element 2 cannot be connected. Leaving a flag variable equal to zero means the corresponding two elements can be connected. The connections between the Element “reservoirs” in the cascade use these flag variables within the model. There are Boolean statements on these connections that restrict tokens entering reservoirs if the token has already passed through a non-compatible reservoir within the generator.

Excel front-end

In order to better visualize and set the constraints in the previous section for a specific problem, an Excel spreadsheet was made to act as a front-end to the N plus M generator (shown in Figure 4-6). The reason for making this front-end was twofold. First, there was a need in the oil problem to be able to enter the latitude and longitude coordinates of each element in a problem, then set some maximum distance beyond which elements would not be allowed to be connected in the generator. This could have been done by the user typing coordinates into the Global Script before each run of the generator, but the Global Script interface is not very user-friendly. Second, with 78 flag variables all equaling one or zero and each having an encoded representation of the connection it represents, it is not easy for a user to visualize how they are setting up the generator and the connection matrix for a given problem. Not to mention that it is tedious to double-check 78 ones and zeros.

The front-end solves both of these problems. On the left side of the front-end spreadsheet is the grid which represents the connection matrix of which the user is trying to generate instances. In this grid, the user places “X’s” or leaves blanks in the positions corresponding to the different connections being considered. Placing an “X” in a position in the grid means the elements in that row and column can be connected. Leaving a blank in a position in the grid means the elements in that row and column cannot be connected. The user can then easily understand and see how the connection problem is being set up. The “X’s” and blanks in the

grid change the values of the various flag variables to the correct values of “1” or “0” in the middle column of equations. On the right side of the spreadsheet, the user can enter the latitude and longitude coordinates (in degrees) for each element. Then some maximum distance (in kilometers) that is allowed for a connection to take place can be specified. The user then clicks on a macro button and the spreadsheet figures the distance between each pair of elements (using a sphere representation of the earth). It flags which of those pairs are inside and outside the maximum allowable distance by placing “X’s” and blanks in the correct corresponding positions in the grid on the left. The user can then change the grid on the left as necessary for the problem. Once the user is satisfied the correct connections are allowed and disallowed, the column of flag variable equations in the middle of the spreadsheet is copied and pasted into the Global Script. The generator is then ready to run and the user can be sure the correct connections will be allowed and disallowed.

Grid allows setting of non-connectable elements by leaving blank spaces and putting in X's

Coordinates for each element are input, then using some maximum distance non-connectable elements are set in the grid

Code is prepared to copy in to OPN Global Script to run the model with the appropriate constraints

NAME	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
Item 1	R1												
Item 2	R2	x											
Item 3	R3		x										
Item 4	R4	x		x									
Item 5	R5		x		x								
Item 6	R6	x		x		x							
Item 7	R7		x		x		x						
Item 8	R8	x		x		x		x					
Item 9	R9		x		x		x		x				
Item 10	R10	x		x		x		x		x			
Item 11	R11		x		x		x		x		x		
Item 12	R12	x		x		x		x		x		x	
Item 13	R13		x		x		x		x		x		x

```

r1r2=0;
r1r3=1;
r1r4=0;
r1r5=1;
r1r6=0;
r1r7=1;
r1r8=0;
r1r9=1;
r1r10=0;
r1r11=1;
r1r12=0;
r1r13=1;
r2r3=0;
r2r4=1;
r2r5=0;
r2r6=1;
r2r7=0;
r2r8=1;
r2r9=0;
r2r10=1;
r2r11=0;
r2r12=1;
r2r13=0;
r3r4=0;
r3r5=1;

```

Coordinates (degrees)			
Item 1	6.784739	S	41.912222 E
Item 2	6.774739	S	41.902222 E
Item 3	6.784739	S	41.892222 E
Item 4	6.794739	S	41.882222 E
Item 5	6.804739	S	41.872222 E
Item 6	6.724739	S	41.862222 E
Item 7	6.734739	S	41.982222 E
Item 8	6.744739	S	41.972222 E
Item 9	6.754739	S	41.962222 E
Item 10	6.814739	S	41.952222 E
Item 11	6.824739	S	41.942222 E
Item 12	6.834739	S	41.932222 E
Item 13	6.844739	S	41.922222 E

- You can change the names of the items in the NAME column
- Place an x in a grid space that corresponds to an allowed connection
- Leave a blank space in the grid for a disallowed connection
- Enter the coordinates in degrees in the coordinates boxes for each item
- Set N or S or E or W (use caps)
- Set the max distance allowed between elements and click the "Set Xs" Button
- When grid is completely set, copy the all the filled cells from r1r2 to r12r13
- Then paste them in the appropriate spot in the "Global Script" of the OPN Model

Max Distance	6	KM
Set X's From Dist		

Figure 4-6. N+M Model Excel Front-end

4.4.6 Other Potential Constraint

There is one other potential constraint that has to be considered but has not yet been implemented. This constraint would set a minimum number of connections for each element.

In order to implement this constraint, one would need to create 13 variables in the Global Script. Each variable would represent the minimum allowable connections for the (potentially) 13 elements. These might be called *R1min*, *R2min*, etc for Element 1, Element 2 and so on. To implement the constraint and use these minimum variables in the model, you would need to restrict tokens from reaching the end which haven't reached the minimum number of connections for a given element. Since the last potential connection to any element occurs in the last block in any problem the restriction has only two possible locations. First, the restriction could be placed on the arrows that form the "exits" from each of the 12 blocks to the center process called Done. That would require writing 12 Boolean statements, one for each of the "exits." The other option would be to write one Boolean statement on the arrow from the process Done to the last object, Final. This is probably easiest since there is only one long Boolean statement to write, which includes all of the *Rxmin* variables compared to their corresponding *Rxcnt* counters. It would restrict any tokens that don't have all of their *Rxcnt* variables set to at least the matching number of connections allowed by the corresponding *Rxmin* variables set in the Global Script. Since this constraint would be enacted only at the end of the generator, it would not have any effect on speeding up the generator by way of pruning the space. This is because the model would still have to generate all of the restricted tokens, only to prune them at the very end.

4.4.7 Output

As in the N by M generator (see 3.4), this generator will funnel all of the tokens that have a complete instance of a connection scheme which satisfies the constraints to the generator end. Each of those tokens contains the variables that represent all of the intersections of the rows and columns of the connection matrix. Each variable will equal either zero or one. Zero indicates that there is no connection at the intersection of that column and row (reservoir and facility), and one shows that there is a connection. These "ones" will be assigned by passing through the particular "reservoir" in the particular block corresponding to a column and row. The labeling convention for these variables is as follows: R31 corresponds to the connection of Element 3 and Element 1 (or in the matrix row 3 and column 1) and R25 corresponds to the connection of Element 2 and Element 5 (row 2 and column 5), and so forth. In addition, each token that reaches the end will have the counting variables discussed above. Essentially, these tokens now have a vector containing all of these variables and data about the connections.

In this OPN model, the first quick check is to view the list of the tokens that have reached the Final object. It is there that the user can read how many tokens have arrived. The number of tokens that have arrived should equal the total number of possible connection schemes. The

user can also graphically view the path each token took and read the output vector described above for each token.

Exporting to Excel

From the OPN object marking the generator end (called Final), one click will export the tokens to an Excel spreadsheet. (This is convenient for sorting and filtering the outputs.) Within the exported spreadsheet, there will be a header row that names all of the columns, each of which represents a variable in the output vector on each token. The following rows each represent one instance of a connection possibility generated by the model. Instructions for running the model and retrieving these outputs are found in Appendix 9. To create a connection matrix from one row in the spreadsheet, one would delete or ignore the counting variables (all of which have a header label ending in "cnt"). One would then move (probably via MATLAB or an Excel macro) the elements of the vector into their corresponding positions in the connection matrix so that it looks like Figure 4-7.

	R1	R2	R3	R4	R5	R6	R7
Element 1 -R1	1	0	0	0	0	0	0
Element 2 -R2	0	1	0	0	0	0	0
Element 3 -R3	1	0	1	0	0	0	0
Element 4 -R4	0	0	1	1	0	0	0
Element 5 -R5	0	0	1	0	1	0	0
Element 6 -R6	1	1	0	1	0	1	0
Element 7 -R7	0	0	1	0	1	0	1

Figure 4-7. Example N+M Connection Matrix Output

4.5 Expansion Possibilities

A possible expansion to this generator would be to improve the front-end for easier use of the whole model by a user. First, it would be good to add a spot next to each element in the front-end to specify a maximum and possibly a minimum number of connections. This would be linked to the information in the middle column of the front-end. It would then be copied and pasted into the Global Script of the OPN model so the user won't have to do much, if any, coding in the Global Script. It would also be nice if the user could start in the front-end by setting the number of elements or size of the problem and then the front-end would only show the grid and coordinate framework for a problem of that size and no bigger. If the matrix size variable was also added to the middle column to be copied to the OPN models Global Script, then there really would be no need for the user to adjust the Global Script except to paste into it.

Finally, the front-end would now have all of the information to calculate the total number of connection scheme possibilities. This would be a great output in the front-end that would inform the user of how many possibilities to expect before running the model. This resulting output would allow the user to gauge how long the model would take to run for some given settings or if the model is feasible to be run at all.

4.6 Limitations

This generator has a couple of limitations. First, in the current version of OPN, one can only run a model that has up to maybe 100,000 possibilities. The model has been run for two hours at the longest, and it produced 36,000 tokens in that time. More likely one wouldn't want to analyze any problem larger than 1,000 architectures. The expanded multi-reservoir, multi-facility problem looked to encompass less than 100 architectures.

During the discussion of the formation of this generator, we considered using the generator with more evaluation and constraint variables. As with any model that has some generator of possibilities and evaluator of those possibilities, there is the question of how much rough evaluation and pruning should be done in the generator and how much should be left for the detailed evaluator in the end. It is believed that there is room for a few more evaluations like cost or time in this generator. These would be rough calculations of metrics for each connection scheme as it develops that could be used to prune the space even more before each is evaluated in more detail. However, there is probably very little computational ability left in this OPN model for such added features. More likely these prunings would be more easily dealt with in some post processing element that could be as simple as a spreadsheet or MatLab code.

4.7 Summary

In summary, this chapter discussed the development of an OPN model that can be used generically to generate all possible links between a set of elements. The model generates all of the possible connection matrices, given a set of elements to be linked and constraints about their linkage. The model uses a blocked concept, similar to that used in the N by M generator, to generate all possible connection schemes, and it limits redundant instances by its structure. There are various restrictions that a user can set to limit the number of connection schemes generated and to meet real world constraints. The model uses an Excel front-end to allow easier setting of the constraints. There is an additional constraint that could be implemented in the future if needed, as well as, added features to the front-end that would enable easier use. The model discussed was developed for a specific oil project, but just as for the N by M model, it is quite conceivable that this model could be used as one of several general tools for building

system architecture models with OPN. One could conceive of a time where someone building an OPN model recognizes the need for a “connection generator” and that person could go to a toolbox to insert one into their model.

CHAPTER 5. Conclusion and Future Work

5.1 Conclusions

This thesis has shown that system architecture methods and thinking can successfully be applied to the development of large, complex, commercial systems, particularly offshore oil and gas production systems. Oil and gas production systems were decomposed and analyzed. The overall problem was shown to be able to be broken down into two parts: the problem of architecting an offshore oil and gas production system which consisted of one facility producing from one oil reservoir, and the problem of architecting a field of multiple facilities and multiple reservoirs. The first was approached as a classic system architecture problem of assigning function to form. The approach to the second problem utilized the solutions to the first problem for the individual facilities and reservoirs, but approached the core of the problem as another type of classic system architecture problem, that of connection and routing.

The problem of architecting one facility and one reservoir was addressed by decomposing and analyzing the oil and gas production process using rigorous, methodical system architecture thinking. This method explicitly archived and represented the process in several graphical frameworks, including OPD's and hierarchical morphological matrices. It was shown that the underlying processes could be represented in a solution-neutral way, and that the specific forms in the overall process could also be represented in the same framework. The solution-neutral representation provides system engineers with a framework to develop "out of the box" solutions. This is enabled by allowing an orderly expansion of the architecture space as ideas evolve. This evolution of ideas in the architecture space can take place in the OPD and Morphological Matrix representations. These representations also provide a method to capture and archive industry knowledge that is not always explicitly stated. The OPD's can be used to indicate the decision points in the design space, which enables the design space to be clearly outlined. Few industries, to include the oil industry, practice methods like the one presented in this thesis, which can clearly represent the full design space in a single chart or matrix. The entire envelope of architectural possibilities is contained in the morphological matrix, which provides a great starting point for understanding the design space before developing concepts. Metrics based on performance, risk, and cost (including throughput, efficiency, TRL, construction cost, development cost, NPV, and schedule) can be developed to prune and sort the decision space. The usefulness of OPN to enumerate all of the architectures in the design space in conjunction with the use of metrics was shown. This automated enumeration can identify new combinations, highlight preferred sets of concepts, and underline functionalities or patterns common to the concepts in the preferred set. This

increases the architects' overall knowledge and understanding of the entire design space, and ensures that all options are considered, facilitating the down-selection to a handful of preferred concepts that can be carried forward for more detailed study and eventual development.

The development of the Alternate Specific Model showed that the decompositions of more specific oil systems can be used to make OPN models that successfully generate thousands of architecture concepts that, with the use of metrics like TRL and cost, offer preferred concepts similar to current industry studies. The Alternate Specific Model also illustrates the method's potential for increasing the productivity of systems engineers. This model was conceived and built in two days after only an hour-long conversation between an industry expert and a student engineer (who was familiar with the oil industry after two years of indoctrination). This demonstrates that our thinking and language of systems architecture is a quick way of decomposing an architecture problem. Now that the models are built, BP engineers can change TRL and cost values, enact different metrics, or consider different water depths and environments. With small, quick changes to the models, they can have swift answers to the nature of a new design space.

This thesis has shown how to better ensure that all options have been considered as a concept is selected for a complex system. This is true whether the aim is to develop a single oil and gas production system, or to develop an oil field where many facilities and reservoirs are connected, as in the thesis's approach to the second part of the offshore oil and gas production system problem, where we considered multiple reservoirs and multiple facilities. The approach to the second part demonstrated that even a modest set of facilities and reservoirs has a vast number of connection possibilities. This space of connection possibilities is so large and daunting that it has been overlooked and not fully explored by most system engineers. Instead, a handful of feasible possibilities are generated by committee and analyzed to find the best solution. The tools developed in this thesis automate the enumeration of this daunting space of possibilities. These possibilities can then be sorted and prioritized through the use of metrics. This method can identify new combinations, highlight preferred sets of connection schemes, and underline patterns common to the schemes in the preferred set. This increases the architects overall knowledge and understanding of the entire space of possibilities and ensures that all options are considered.

In conclusion, this thesis showed that this method of systems architecture thinking has great promise for other large, complex, commercial systems.

5.2 Recommended Future Work

There are many steps that could be taken to advance this research past the current point. The following lists those steps and is broken into the two overall problems in the offshore oil and gas production architecture problem.

One Facility and One Reservoir Models

Reconcile the processes, forms, and attributes represented in the general oil and gas production system model and the alternate specific model used in the TRL metric case. In particular, note all the processes and forms in the general model that are not represented in the specific model. Clarify why they are not represented. Ensure that the general model captures all the processes, forms and attributes found in the specific model. This may include expanding and adding to the general model.

Enhance the architecture generator models to enable a user to input various location, environmental, and other attributes to “tune” the model to specific situations. In particular, implement a way to enable a user to enter information about the situation (possibly including: water depth, reservoir characteristics, environment, distances, geographical and geological data). Refer to the objects under Situation on Figure 2-2, as well as all the attributes for the locations as shown in Figure 2-7. Link these inputs to metrics to be able to give more accurate estimates of the various metrics, particularly cost.

Implement new metrics as outlined by BP. These might include schedule, risk, production performance, more accurate cost estimates, environmental impact, social impact, developmental requirements, ice protection requirements, flexibility, extensibility, and commonality. The implementation of any of these metrics will require access to, and the gathering of, more data. This may include significant work with OGM (Oil and Gas Manager) software for cost and schedule data. OGM clearly gives detailed cost data, but the use of its construction man-hour outputs could be used to produce schedule estimates.

Work on the development of ice protection architectures. Our work was limited by the lack of industry progression in this area. In particular, component models of ice-structure interactions and natural ice behavior are still not to a level that easily enables their use in system-level modeling and thinking. As these component models become better, work on the development of ice protection architectures can go forward.

Multiple Facility and Multiple Reservoir Models

Develop the connection generator to consider not only how elements in a field are connected, but what sequence of smaller connections could lead to those final connection schemes. This work is of particular interest to Prof. Oli de Weck. It includes thinking about all the sequence paths that lead to a connection scheme and potentially implementing a generator that produces not only all connection schemes but also all the connection sequences that lead to a connection scheme. The size of this problem (generating all the possible connection schemes and all the possible sequences that lead to each of those schemes) is potentially the original connection problem size raised to another exponent. One idea is that the current connection generator could also generate these sequences if the reservoirs in the reservoir cascade were fully connected; in that case the order in which the token goes through the then “non-cascading” reservoir cascade would be captured and imply a sequence possibility.

Enhance the connection generators’ Excel based front-end to encapsulate all the constraint settings for the connection generator. In particular, consider adding the ability to set the total number of elements, and the maximum and potentially minimum number of connections per element, to the Excel front-end so that the user need only copy and paste once into the tool’s Global Script.

Enhance the connection generator front-end to calculate the total number of possible connection schemes based on the constraint settings, so the user has an idea of the size of the set of possibilities the generator is about to produce.

In the connection generator models, implement the extra constraints suggested in sections 3.5, 4.4.6, and 4.5.

Appendices:

APPENDIX 1. OPM Symbols

 System Boundary

 Interacts with

 Enables/Consumes

 Instrument of

 Operator of

 Decomposes to

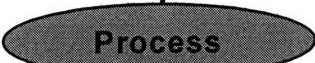
 Has Attribute of

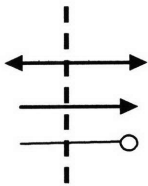
 Specializes to

 Has Instance of

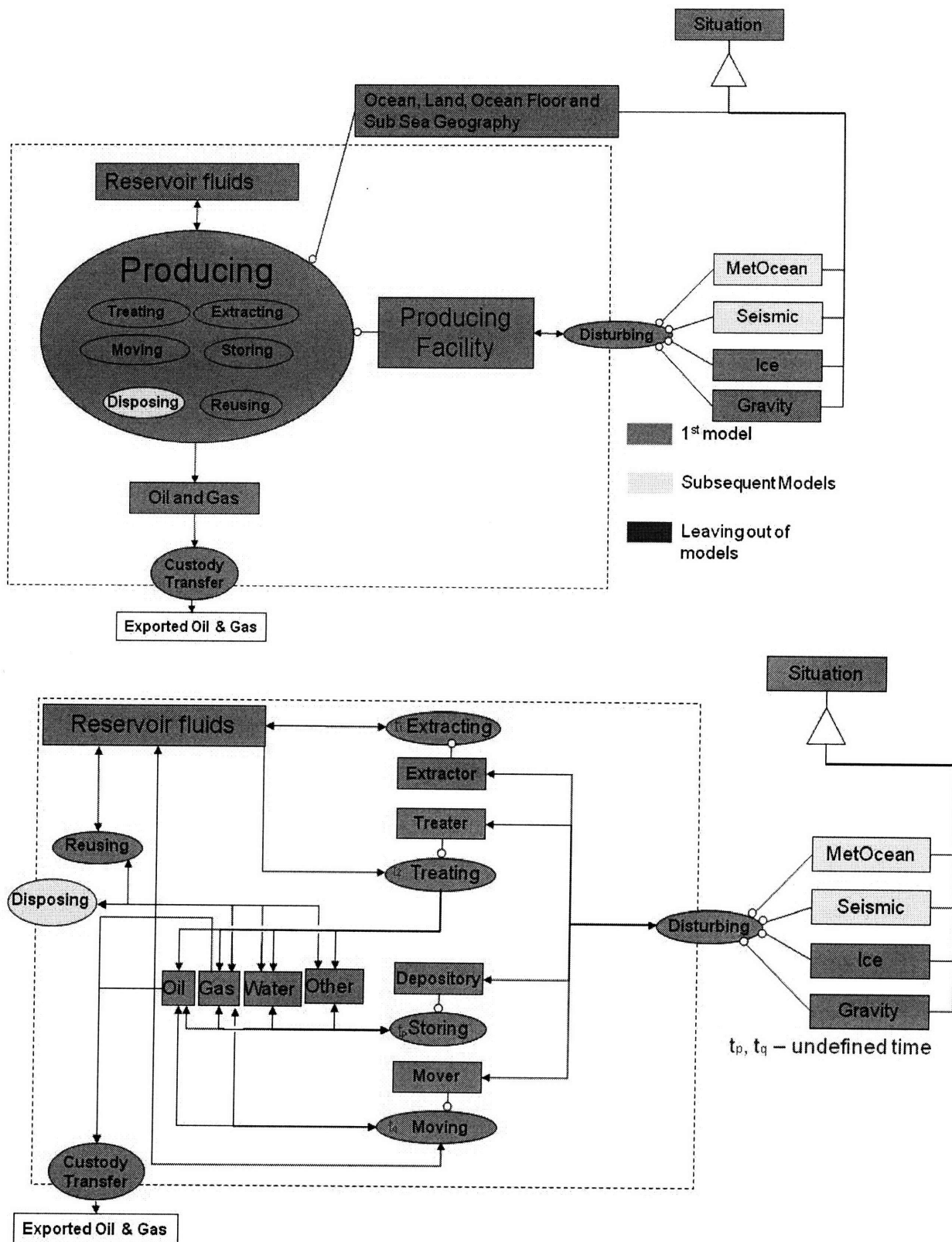
 Object
A piece of form;
something tangible

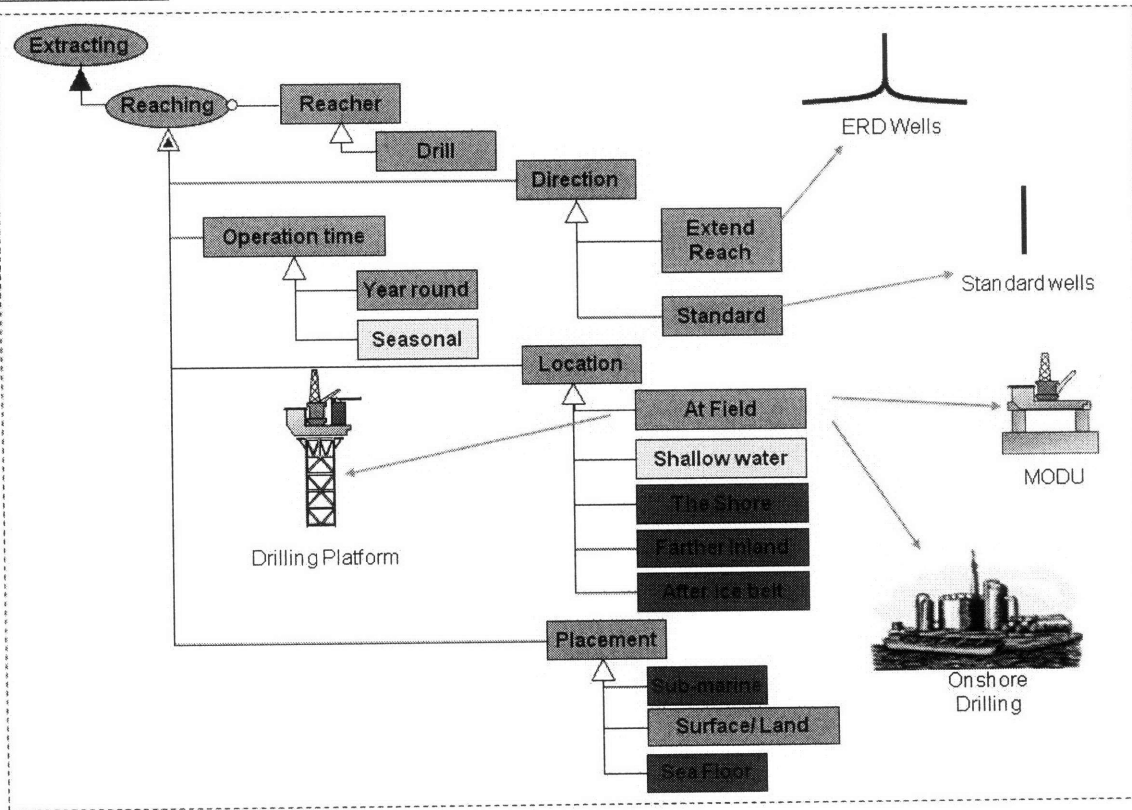
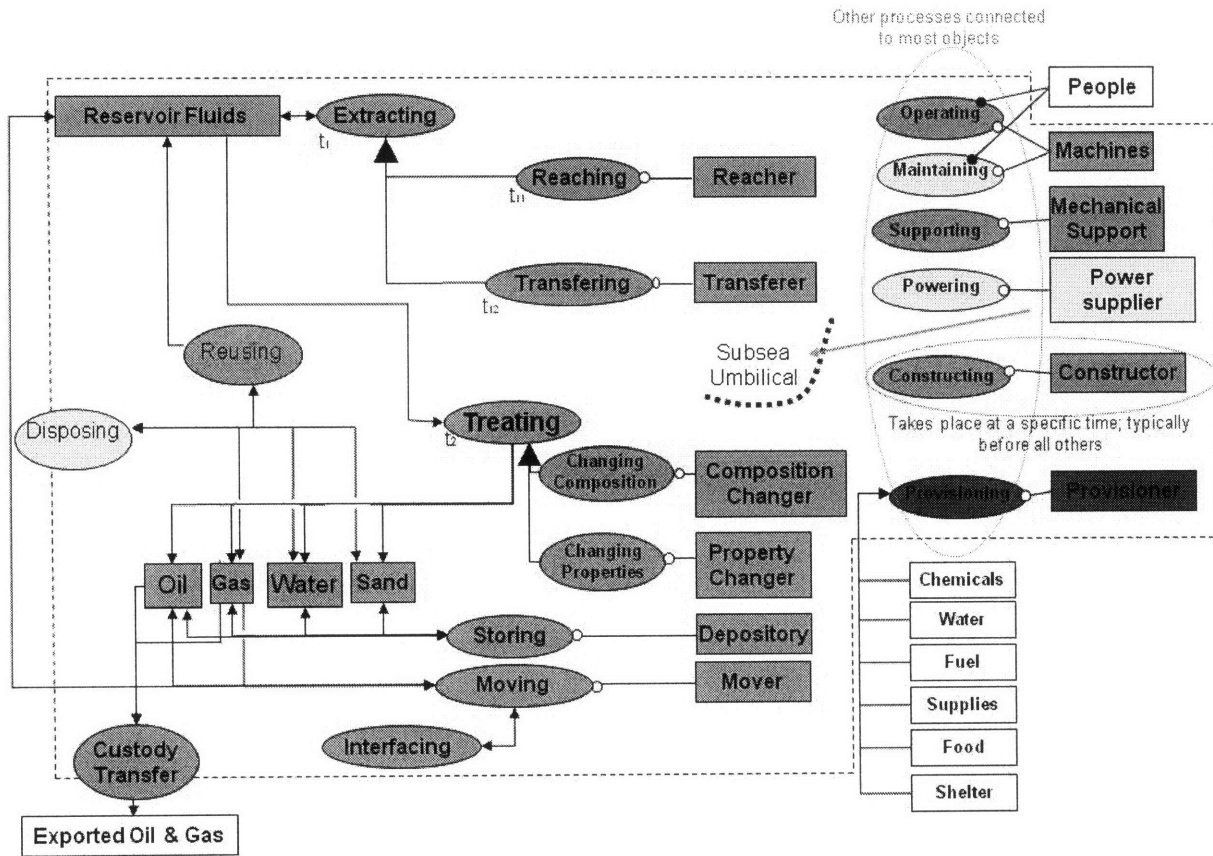
 Process
An action of some
kind; not tangible

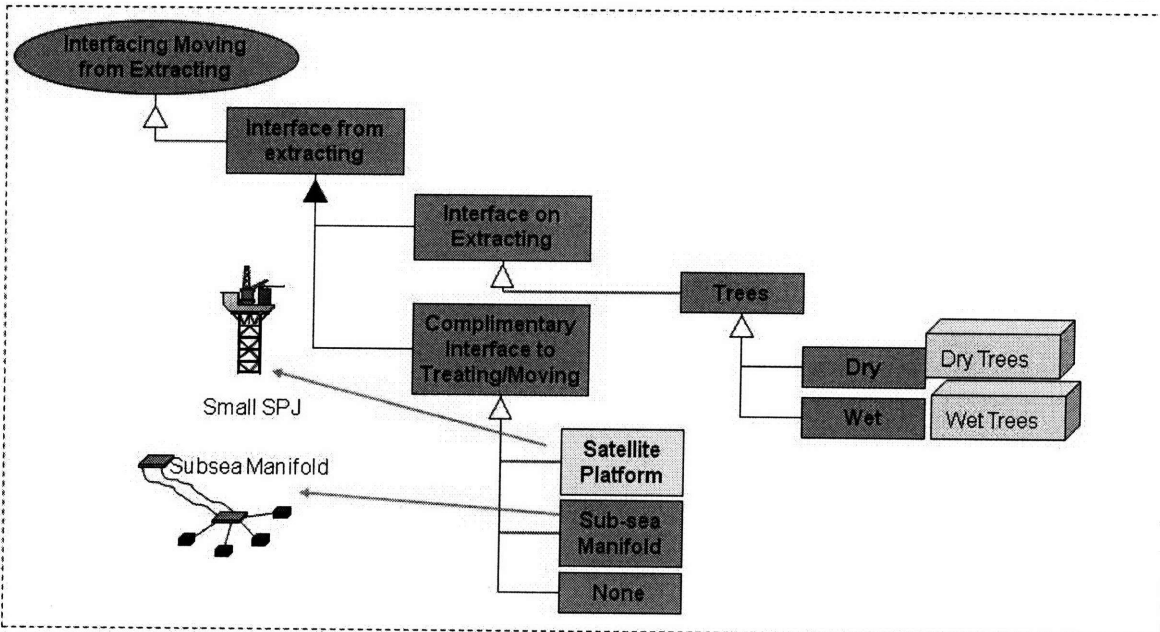
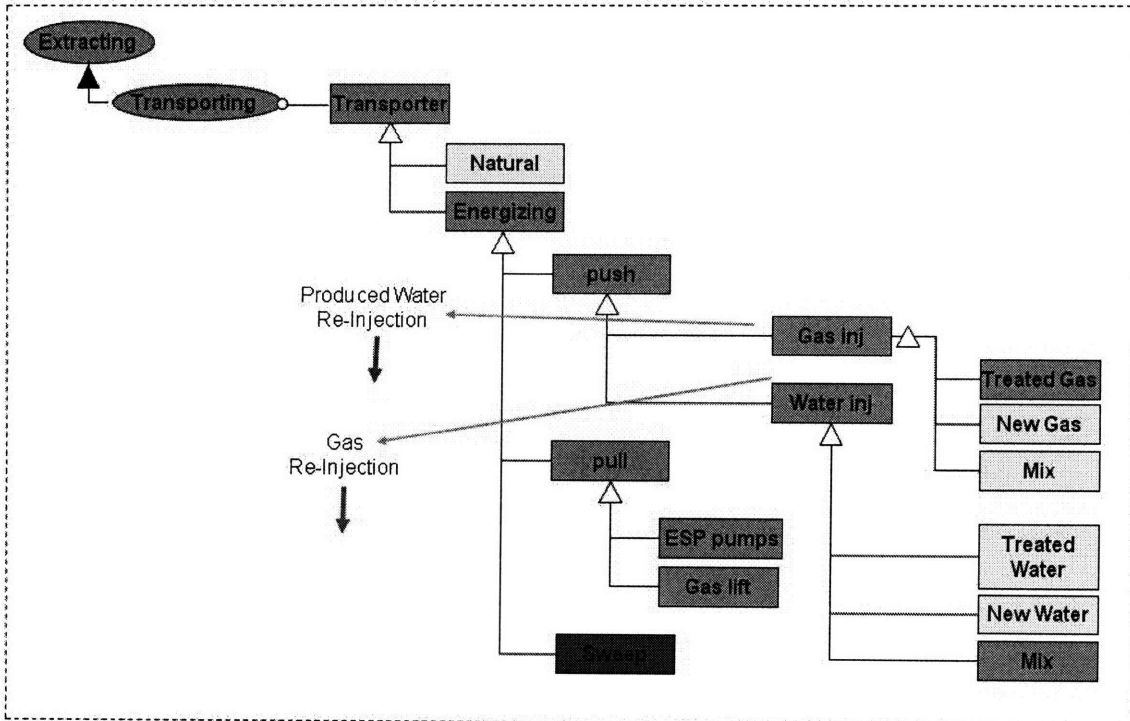
 Process
A process that takes
place at the boundary

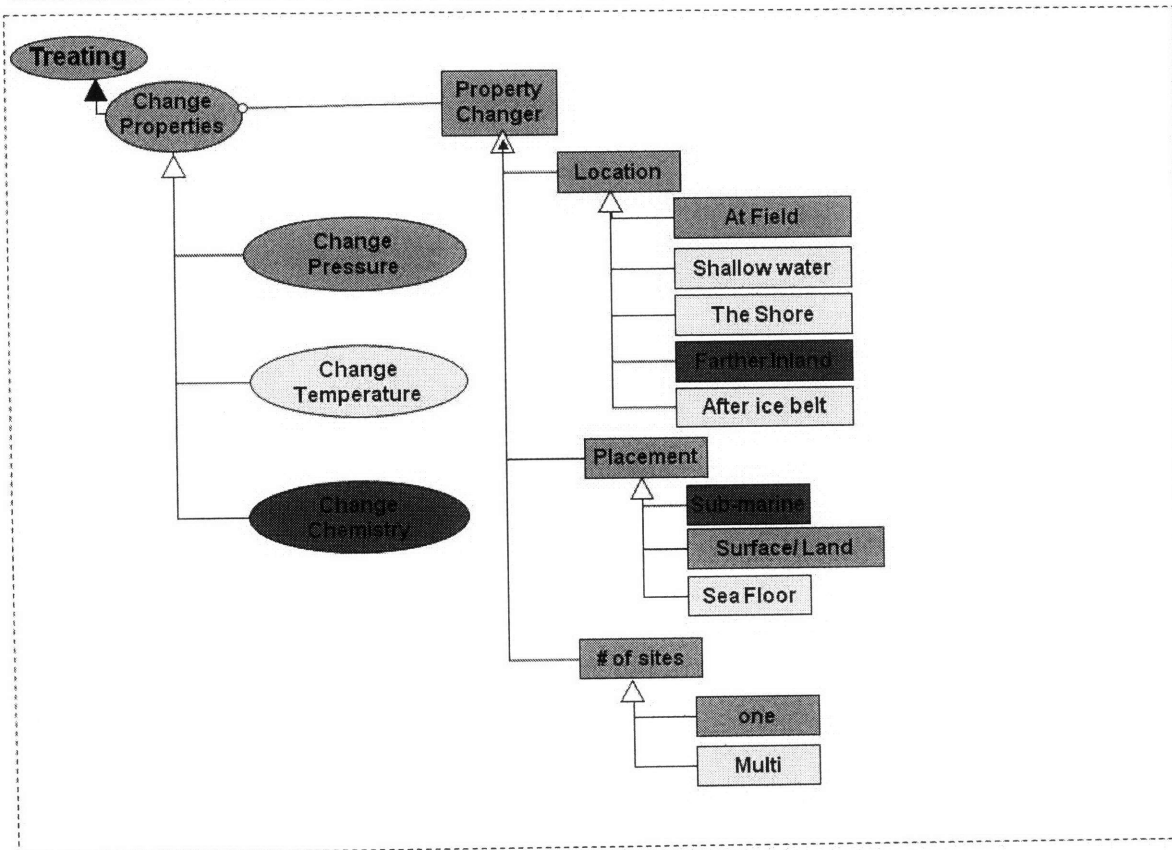
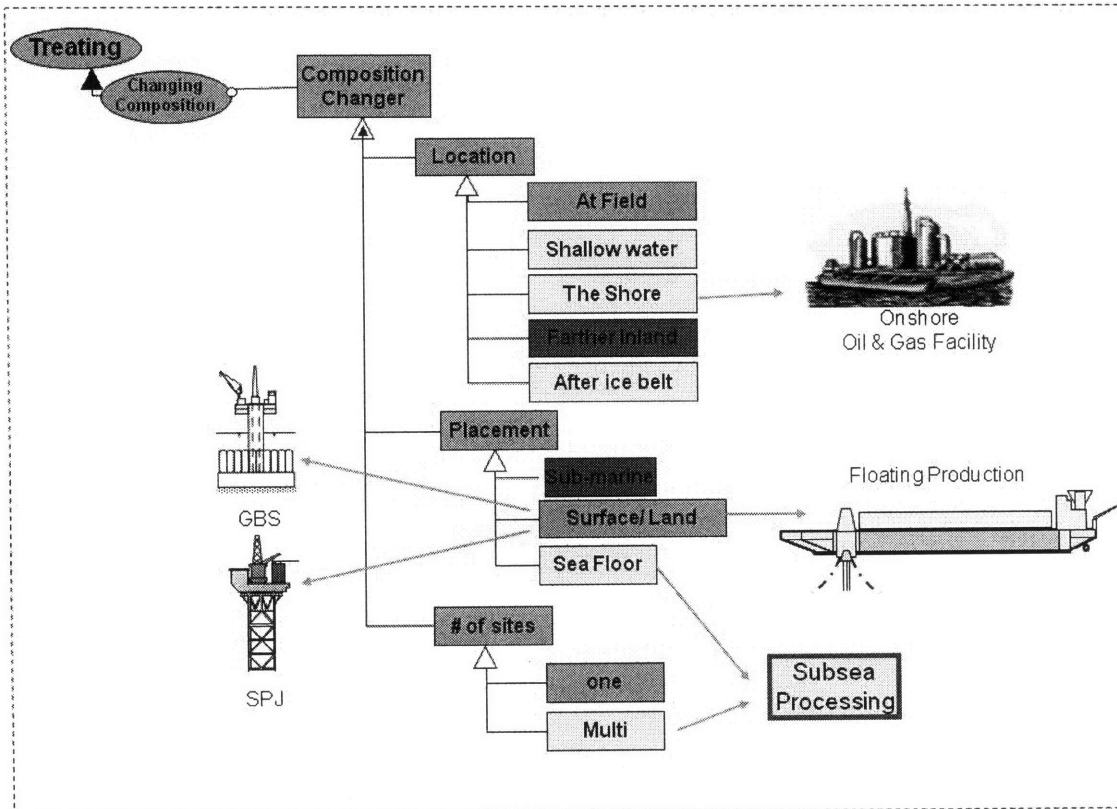

Something crosses
the system boundary;
Usually requires an
interface of some kind

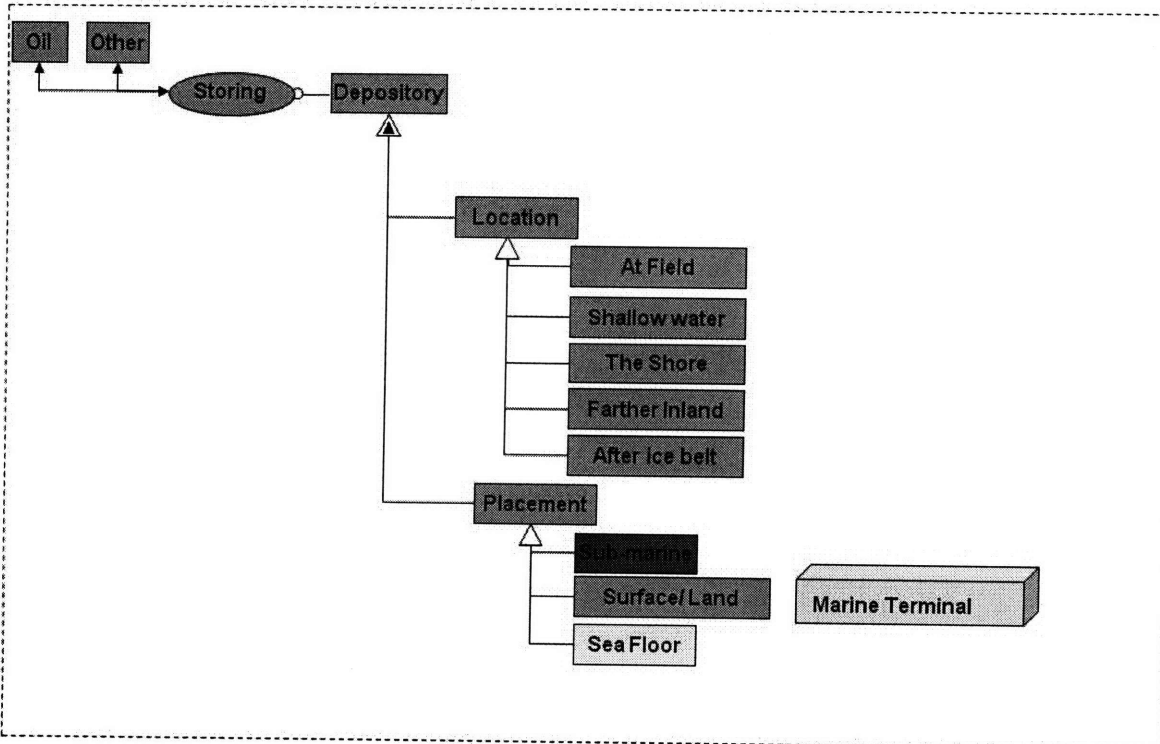
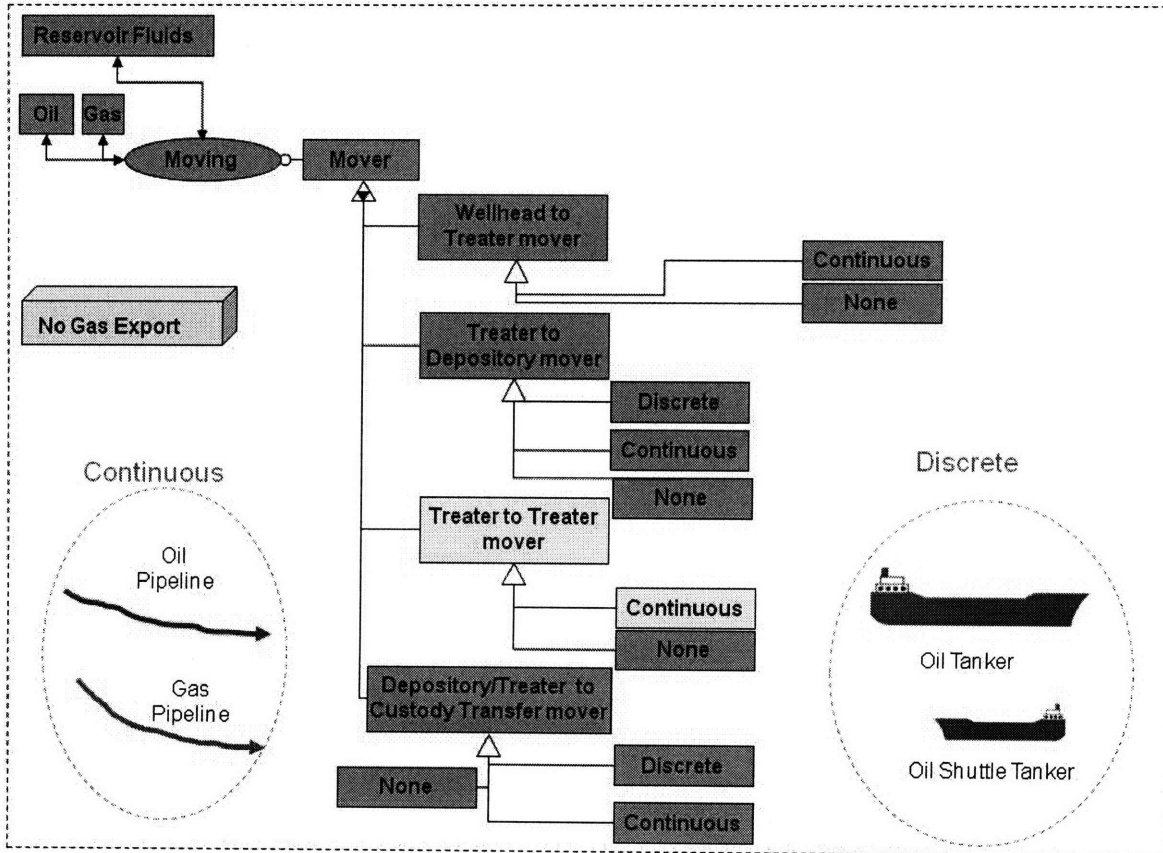
APPENDIX 2. Oil Decomposition Levels

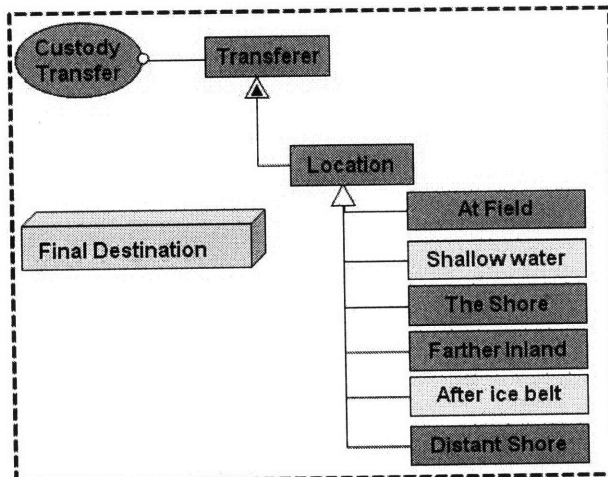
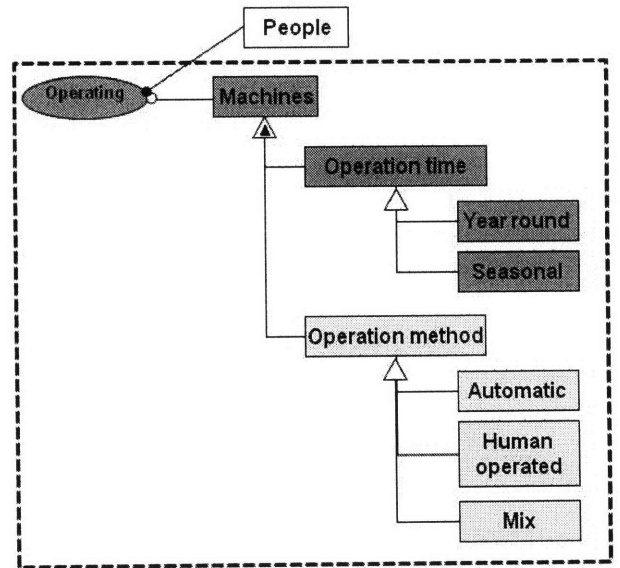
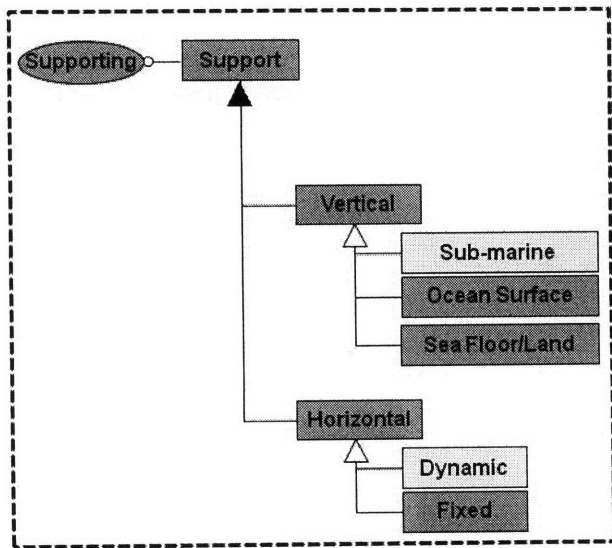
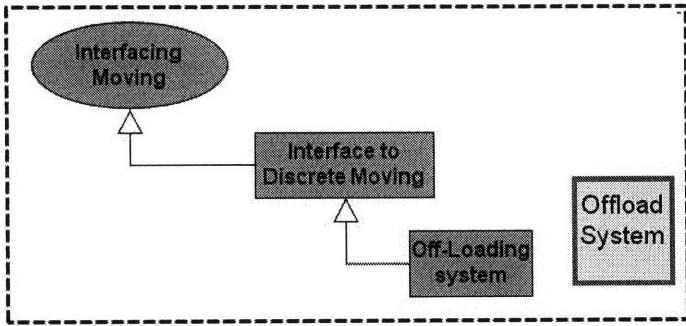












APPENDIX 3. Constraints and Explanations

- 1) no standard drill with shallow water well location
 - *Assume the reservoir would not be in shallow water so extended reach drill would have to be used*
- 2) no platform driller with wet or mixed trees, no MODU with dry or mixed trees
 - *Platform drillers only drill dry trees and MODU's only drill wet trees*
- 3) no dry trees with subsea manifolds; wet or mixed trees must have subsea manifold
 - *Subsea manifolds are only for wet trees and must be used with all wet trees*
- 4) no standard drill with shallow water, the shore or after ice belt treater location
 - *Assume the reservoir would not be at these locations so extended reach drill would have to be used*
- 5) no dry trees on sea surface treater placement, no wet trees with on land treater placement,
 - *Dry trees can only be used with a few types of floating platforms (spars and TLPs) not represented here; if you did have wet trees going to an on land treater then it would be cheaper just to do extended reach drilling both of which are situations in which the reservoir would be close to the shore and thus too trivial to be modeled by this model*
- 6) no mixed trees with on land or sea surface treater placement
 - *fixed platforms are the only treater placement that can possibly have mixed trees (other than TLPs and spars which are not represented in this model)*
- 7) no on sea floor or on land treater placement or dry or mixed trees or the shore treater location with Semi or FPSO substructure,
 - *Semisubmersibles and FPSO are on the sea surface and thus not on the shore and with wet trees only*
- 8) no at field or shallow water or after ice belt treater location or platform driller or on sea floor or on sea surface treater placement with no substructure
 - *If there is no substructure a facility can only be on land or on the shore*
- 9) no the shore or after ice belt treater location or on land or on sea surface treater placement with GBS or SPJ substructure,
 - *GBS and SPJ's go at the field or in shallow water and are placed on the sea floor*
- 10) must be no storage facility for FPSO treater
 - *FPSO is also a Storage facility by definition so assume there wouldn't be a separate one*
- 11) no on land storage placement with at field or shallow water or after ice belt storage location,
 - *for locations in the water there is no land to place a storage facility on*
- 12) no on sea surface storage placement with the shore or farther inland storage location,
 - *for location on land you would not have water to place a storage facility on or a sea floor for that matter*
- 13) no on the sea floor storage placement with the shore or farther inland or after ice belt storage location
 - *for location on land you would not have a sea floor to place a storage facility on, assume the same for after ice because the water is too deep*

- 14) no the shore or after ice belt or on land or on sea surface with GBS or SPJ,
 • *the same as 8 only for a storage facility instead of a treating facility*
- 15) no on sea surface or on land or the shore with Semi or FPSO,
 • *same as 7 only for a storage facility instead of a treating facility*
- 16) no at field or shallow water or after ice belt or on sea floor or on sea surface or FPSO with none
 • *if you do not have a storage facility substructure then a storage facility can only be at the shore and on land and the treater cannot be a FPSO*
- 17) no dry trees with pipeline from wellhead to treater
 • *if you have a pipeline from the wellhead to the treater then assume you don't have dry trees (a satellite platform would change this)*
- 18) must be no mover from treater to storage for FPSO treater and if there is no storage facility,
 • *no need for a mover from treater to storage if there is no storage facility or if there is an FPSO treater because you treat and store at the same facility so no moving is required*
- 19) no pipeline mover from to storage for Semi Treater
 • *assume not common to have a pipeline from a Semi (becoming more common)*
- 20) no pipeline mover to custody transfer with FPSO treater or shipshape storer or semi treater
 • *Shipshape facilities including FPSO cannot hook up to pipelines*
- 21) no offloading facility with pipeline mover or none for treating to storage mover, must be none if no storage facility
 • *an offloading facility is only for tanker movers; no need for an offloading facility from a storage facility that doesn't exist*
- 22) no offloading facility with pipeline mover or none for mover to custody transfer
 • *no need for an offloading facility to mover that doesn't exist (there actually could be but the mover would not be in the system)*
- 23) if no storage facility then 'none' for 'to custody transfer mover' if match with treater location,
 • *if there is no storage facility then the oil will go from the treater to custody transfer and if those two are at the same location then we assume there is no need for a mover*
- if the treating location is ...
- 24) ...at the field then the wellhead location must be at the field
 • *Think about the locations as required sequence at field to shallow water to the shore to farther inland OR at field to after ice belt to the shore; Then the overall process will only go in the direction of those two paths; never in the reverse of those two paths. You can start the overall process at most locations but then you can't ever go to a next step in the process in the reverse direction in one of the two sequences. For instance you would never drill and treat in shallow water and then store or custody transfer at the field but you could do both on shore. If you are treating at one location then the wellhead must be at the same location or at a location closer to the start of the location sequence*
- 25) ... in shallow water then the drill type must be extended reach
 • *similar to 1*

- 26) ... in shallow water and the wellhead is in shallow water then mover from wellhead must be none
- *no need for a mover if they are in the same location*
- 27) ... in shallow water and the wellhead is at the field then mover from wellhead must be pipeline
- *the mover must be continuous i.e. pipeline, for any movement before treating*
- 28) ...the shore then the drill type must be extended reach and the mover from wellhead must be pipeline
- *see 25 and 27*
- 29) ...after ice belt then drill type must be extended and the wellhead must be at field and mover from wellhead must be a pipeline
- *similar to 1 and 27*
- if storage location is...
- 30) ...in shallow water then treating location must be at the field if there is a mover to storage or after ice belt if the mover to storage is tanker
- *see 24 if there is a mover to storage that means the storer and treater are not in the same location; you can only have a mover to storage at the location after ice belt of a tanker, pipelines aren't practical in water assumed to be too deep*
- 31) ...at the shore then the treating location can't be at the shore
- *For the model assumed a storage facility is only considered separately if it is at another location otherwise storage may or may not take place at the treating facility on shore we assume that is just part of the treater design not an architecture consideration*
- 32) ...at the shore then and there must be a mover to storage if treating location is at field or in shallow water
- *since the treater and storer are in different locations there must be a mover between them*
- 33) ...at the shore and the treating location is after ice belt then mover to storage must be a tanker
- *assume there will be no pipelines going to after the ice belt since after the ice belt facilities are assumed to be ship-shaped*
- 34) ...after ice belt and the treating location is after ice belt then mover to storage must be a none
- *no need for a mover they are at the same location*
- 35) ...after ice belt and the treating location is at field then mover to storage must be a tanker
- *see 33*
- 36) ...farther inland then the treating location must be at the shore and the mover to storage must be pipeline
- *The only need for storing further inland comes if you have a treating facility on shore And you must move between the two with a pipeline. There are no land tankers (suppose you could truck it but that's not very practical)*

Coded abbreviation for each decision used as variables in OPN Model code

Value assigned to each decision variable to represent each individual choice. Values are in gray below corresponding individual choice. Default value is zero meaning not choice was made.

APPENDIX 4. Abbreviations and encoding used in Oil OPN

Function	Form	Attribute	DT	Possible Choices					
Extracting-Reaching	Wells	Drilling	<i>DT</i>	Standard			Extended Reach		
				1			2		
		Well Head Location	<i>WHL</i>	At field			Shallow Water		
				1			2		
	Driller		<i>DRT</i>	Platform		MODU		Both	
				1		2		3	
Extracting-Transport	Reservoir Driver -push		<i>RDS</i>	Gas and Water Inj		Gas Inj		Water Inj	
				3		1		2	
Interface Extracting	Tree Type		<i>TT</i>	Dry		Wet		Mixed	
				1		2		3	
	Facility Between Wellhead and Treating		<i>SM</i>	Satellite Platform		Sub-sea Manifold		None	
				1				0	
Treating	Treating Facility	Geographic Location	<i>TGL</i>	At field		Shallow Water		The Shore After Ice Belt	
				1		2		3 4	
		Vertical Placement	<i>TVP</i>	On land		On Sea Surface		On Sea Floor	
				1		2		3	
	Substructure		<i>TSB</i>	GBS	SPJ	Semi	FPSO	Artificial Island	None
				1	2	3	4	5	0
Storing	Storing Facility		<i>S</i>	Yes				No	
				1				0	
		Storing Geographic Location	<i>SGL</i>	At field	Shallow Water	The Shore	Farther Inland	After Ice belt	None
				1	2	3	5	4	0
	Storing Vertical Placement		<i>SVP</i>	On Land		On Sea Surface		On Sea Floor	
				1		2		3	
	Substructure		<i>SSB</i>	GBS	SPJ	Shipsshape		Artificial Island	None
				1	2	3		4	0
Moving	Mover from Wellhead to Treating Facility		<i>MWT</i>	Pipeline				None	
				1				0	
	Mover From Treating Facility to Storage		<i>MTS</i>	Pipeline		Tanker		None	
				1		2		0	
	Mover To Custody Transfer		<i>MCT</i>	Pipeline		Tanker		None	
				1		2		0	
Interface Moving	Interface to Tanker from Treating Facility		<i>IT</i>	Off-loading System				None	
				1				0	
Interface Moving	Interface to Tanker to Custody Transfer		<i>ICT</i>	Off-loading System				None	
				1				0	
Custody Transfer	Custody Transferer	Location	<i>CTL</i>	At field	Shallow Water	The Shore		Farther Inland	Distant Shore
				1		3		5	6

APPENDIX 5. TRL Scale (From sponsor presentation)

Technology Readiness Levels

TRL	Development stage completed	Definition of development stage
0	Basic R and D paper concept	Basic scientific/Engineering principles observed and reported. Paper concept, no design history. No analysis or testing completed.
1	Proof of concept as a paper study or R and D experiment	(a) Technology concept and/or application formulated (b) Concept and functionality proven by analysis or reference to features common with/to existing technology No design history. Essentially a paper study not involving physical models but may include R&D experimentation
2	Experimental proof of concept using physical or mathematical model tests	Concept design or novel features of design validated by a physical model/dummy functionally tested in a laboratory environment or mathematical modelling with proven validity. No design history. No environmental tests.
3	Prototype system function, performance and reliability tested	(a) Item prototype built and put through (generic) functional and performance tests including reliability and robust design test programme in relevant environments or equivalent testing through rigorous mathematical modelling using proven techniques. (b) Extent to which application requirements are met are assessed and potential benefits and risks demonstrated.
4	Pre production system environment tested	Designed and built as production unit (or full scale prototype) and put through its qualification programme in (simulated or actual) intended environment but not installed or operating. System prototype function and performance requirements demonstrated in the intended operating conditions and environment.
5	Production system interface tested	Designed and built as production unit (or full scale prototype) and integrated into intended operating system with full interface and functional test but outside the intended environment.
6	Production system installed and tested	Production unit (or full scale prototype) built and integrated into the intended operating system. Full interface and function test programme performed in the intended (or closely simulated) environment and operated for less than three years.
7	Production system field proven	Production unit integrated into intended operating system, installed and operating for more than three years with acceptable reliability supported by performance monitoring.

APPENDIX 6. Data for the Alternate Specific Architecture Model

	Architecture Section	Concept Number	Critical Components	TRL 1	Cost 1	TRL 2	Cost 2
1	Production Facilities	1,2,7,8,9,12	Processing Facilities	7	H	7	H
2		3	Dry Trees	7	L	7	L
3		3,4	Process and Drill Facilities (ops)	7	M	7	M
4		4,5,6	Dry Trees	6	L	6	L
5		5,6	Process Capacity	7	M	7	M
6		9,12	Swivel	7	M	4	M
7		10	Subsea HPPs	7	L	4	L
8		11	Dependent on host facility	7	L	7	L
9	Hull and Layout	1,2,3	Hull	3	H	3	H
10		1,2,3	Tandum Offloading	3	M	3	M
11		1,2,3	Concrete Hull concept	1	H	1	H
12		1,2,3	Storage Design >300mb	3	M	3	M
13		3	Construction of Larger Hulls	3	M	3	M
14		3	Process & Drill Layout (construcability)	3	M	3	M
15		4	Storage (dry Tanks)	0	M	0	M
16		4	Storage (wet Tanks)	4	M	4	M
17		4	Offloading	3	M	3	M
18		4	Concrete Hull	0	H	0	H
19		5	Hull	4	H	3	H
20		6	Hull (post typhoon)	4	H	3	H
21		7,8	Hull	7	H	7	H
22		9	Hull(turret/spread moored w/ offload buoy)	7	H	7	H
23		10,11	NONE				
24		12	Dry Tow	7	L	7	L
25		12	Wet Tow	4	L	4	L
26		12	Split Hull	7	H	7	H
27	Moorings	1,2,3,7,8	Chain/wire/chain/Polyes	7	L	3	L
28		1,2,3,7,8	Suction anchor design	4	L	1	L
29		1,2,3,7,8	Suction anchor Install	7	M	4	M
30		1,2,3,7,8	Piles (driven)	7	L	1	L
31		5,6	Tendons	2	M	0	M
32		5,6	Foundations-design	6	L	4	L
33		5,6	Foundations-installation	6	L	1	L
34		5,6	Tendon/tcp interface	3	M	1	M
35		9,12	Spread moored/turret	7	M	3	M
36		9,12	Chain/wire/chain	7	L	4	L
37		9,12	Suction anchors design	7	L	1	L
38		9,12	Suction anchors install	7	M	7	M
39		10,11	NONE				
40		12	DP	7	M	7	M
41	Risers	1,2,3,9,10,11,12	SLHR	4	M	2	M
42		1,2,3,4,9,10,11,12	SCR's	1	M	1	M
43		1,2,3,4,9,10,11,12	Flexibles	4	M	1	M
44		4	TTR's	5	L	2	L
45		5,6	TTR's Production + injection	5	L	2	L
46		5,6	Tensioners/stroke	5	L	2	L
47		5,6	Completion design / Analysis	4	L	2	L
48		7,8	SCR's	7	M	2	M
49		7,8	SLHR	4	M	2	M
50		7,8	Flexibles	4	M	1	M
51	Subsea	1-10,12	Trees/controls	7	M	5	M
52		1-10,12	Manifold	7	L	5	L
53		1-10,12	Flowline	7	H	5	H
54		1-10,12	Umbilicals	7	M	5	M
55		1-10,12	Re-usable flexibles (low cost Chinese)	7	L	1	L
56		11	Pumps (multi-phase)	6	L	4	L
57		11	Subsea Separation	4	H	3	H
58		11	Raw Water Injection	4	H	4	H
59		11	Compression	2	M	1	M
60		11	Power distribution	6	L	2	L

61	Drilling Facilities	1	Offshore rig Transfer	2	M	2	M
62		1	HP Transfer hoses from TSV	7	M	7	M
63		1	Ballasting system for transfer	2	L	2	L
64		1	Power Umbilical	7	L	7	L
65		1	Fluid/cuttings Return hoses	7	L	7	L
66		3	Combined Drill/pod layouts	2	M	2	M
67		4	As per GOM	7	M	7	M
68		2,5,6,7,8,9,10,11,12	Semi Drill Rigs	7	M	7	M
69		8	SIMOPS procedures	7	L	7	L
70		11	semi's With esp's	5	M	5	M
71	Drilling Risers	1	Tensioned riser off caisson	3	M	3	M
72		1	Riser size (18 3/4)	1	M	1	M
73		1	Tensioning equip	1	L	1	L
74		1	SIMOPS During rig moves	7	L	7	L
75		2,3	Riser	7	M	7	M
76		4	As per GOM	7	M	7	M
77		5,6	S/S BOP	6	M	2	M
78		5,6	Surface BOP/riser	5	M	2	M
79		5,6	with pre-drill	7	L	7	L
80		7,9,10,11,12	Riser	7	M	7	M
81		8	Semi Drilling	7	M	7	M

Component Cost Bins (\$millions)			
H	M	L	VH (pipeline)
100	25	1	300

APPENDIX 7. Operation of the N by M connection generator model

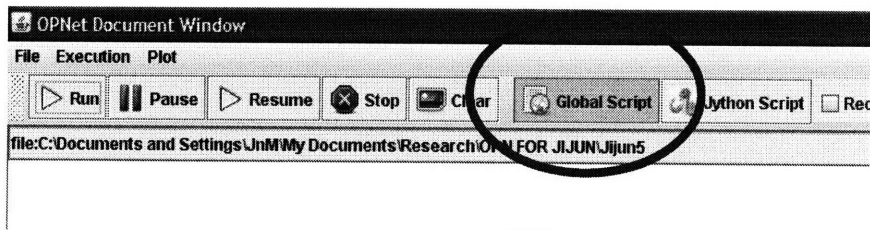
When first opening the model, if you see no model in the window, zoom way out with the mouse roller until you see the model, then zoom in on it. For some reason the graphics does not always center the model in the window.

The screenshot displays the OPNet Document Window interface. On the left, a complex network diagram is visible, consisting of numerous nodes and connecting lines. The right side of the window features a table with the following columns: Row#, Name, Object, Process, and steps. Below the table are buttons for 'Zoomable View' and 'Birds Eye View', along with 'Save Token Graph' and 'Save Token Trace'. At the bottom right, there is a 'Static Properties' section with 'Current Object' and 'Most Recent Process' fields, and a table with 'property name' and 'value' columns. The bottom of the window shows a toolbar with 'Zoom In', 'Zoom Out', 'Static Layout', 'Remember Layout', 'Transform', 'Pick', and 'Edit' buttons, and a Windows taskbar at the very bottom.

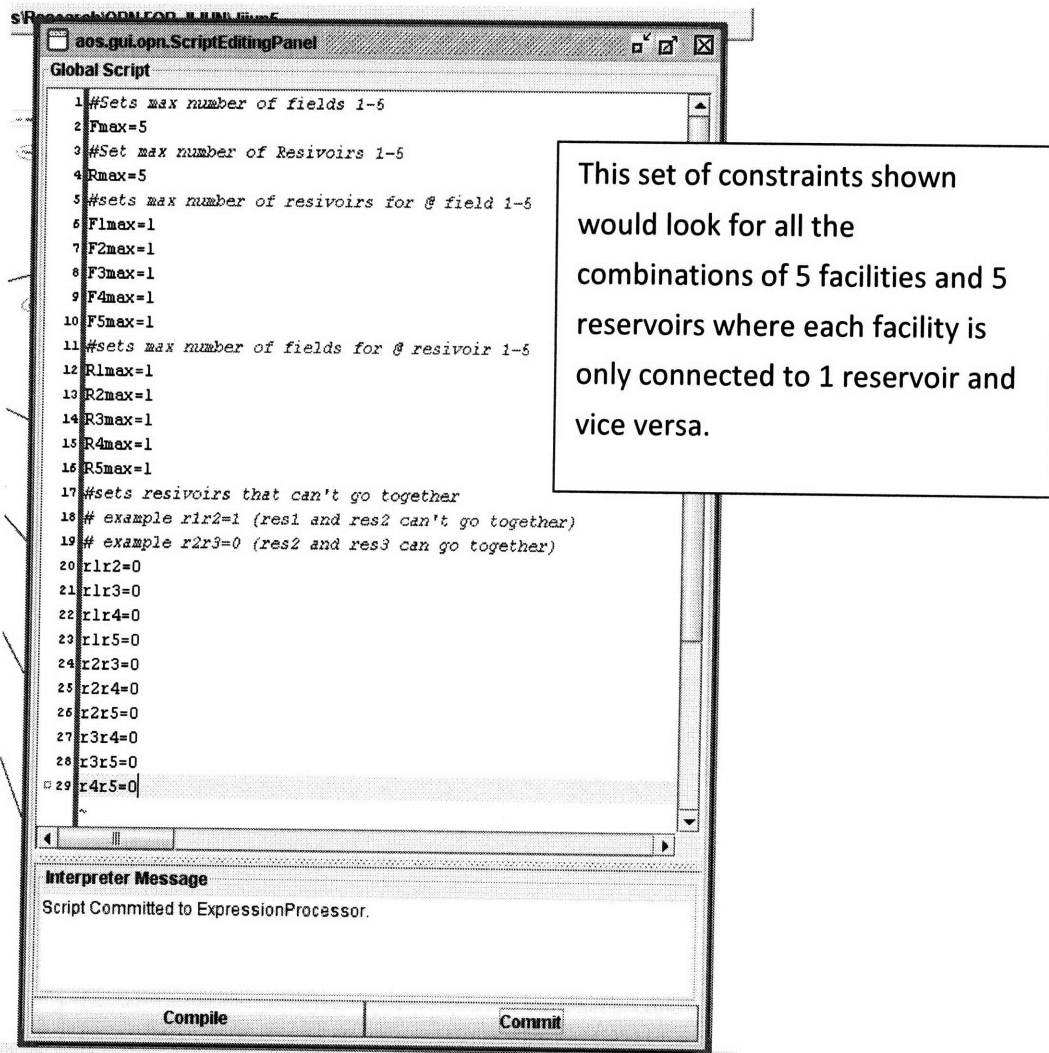
Row#	Name	Object	Process	steps
0	T@_l	init	NA	0
1	T@_l	init	init_proc	1
2	T@_l	Field 1	Field 1 st	2
3	T@_l	Field 1	Res5-1	3
4	T@_l	1R4	Res3-1	3
5	T@_l	1R3	Res2-1	3
6	T@_l	1R4	Res2-1	3
7	T@_l	Field 1	Res2-1	3
8	T@_l	1R5	Res2-1	3
9	T@_l	Field 1	Res3-1	3
10	T@_l	1R5	Res3-1	3
11	T@_l	1R5	Res4-1	4
12	T@_l	1R4	Res3-1	4
13	T@_l	Field 2	Field 2 St	4
14	T@_l	Field 2	Field 2 St	4
15	T@_l	Field 2	Field 2 St	4
16	T@_l	1R5	Res4-1	3
17	T@_l	1R5	Res4-1	5
18	T@_l	1R5	Res3-1	4
19	T@_l	1R5	Res4-1	4
20	T@_l	Field 2	Res1-2	5
21	T@_l	Field 1	Res4-1	3
22	T@_l	2R4	Res1-2	5
23	T@_l	Field 2	Field 2 St	4
24	T@_l	Field 2	Res1-2	5
25	T@_l	2R4	Res1-2	5
26	T@_l	2R5	Res1-2	5
27	T@_l	2R5	Res1-2	5
28	T@_l	2R3	Res1-2	5
29	T@_l	1R5	Res1-1	3
30	T@_l	2R2	Res1-2	5
31	T@_l	2R3	Res1-2	5
32	T@_l	Field 2	Res1-2	5
33	T@_l	2R4	Res1-2	5
34	T@_l	2R5	Res1-2	5
35	T@_l	2R2	Res1-2	5
36	T@_l	2R5	Res 4-2	6
37	T@_l	Field 1	Res1-1	3
38	T@_l	1R2	Res1-1	3
39	T@_l	Field 3	Field 3 St	6
40	T@_l	1R3	Res1-1	3
41	T@_l	1R4	Res1-1	3

Step 1. Set constraints in Global Script

Click on Global Script.



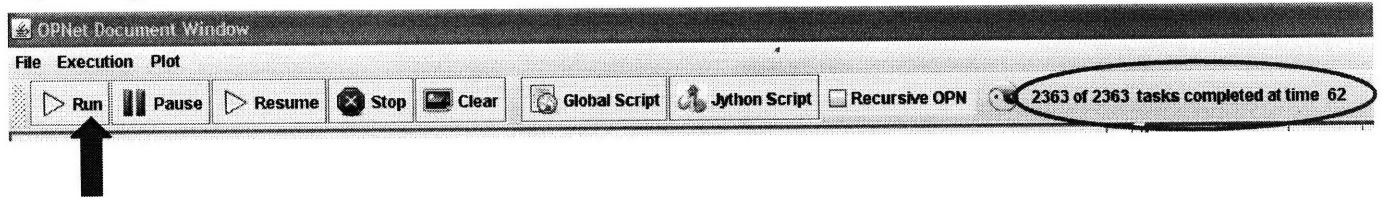
Set constraints by changing variables.



Comments in the code above each constraint should be self-explanatory. When finished setting constraints, click 'compile' and then 'commit'.

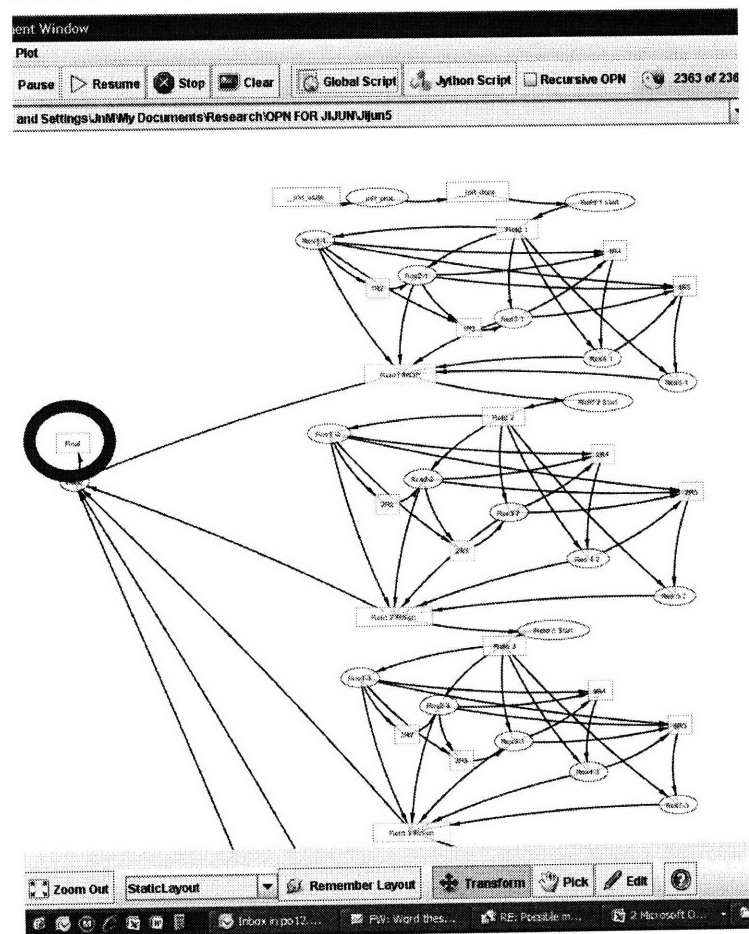
Step 2. Run OPN

Click Run. Wait for counter to stop running. Run time range is 3-seconds to 10 plus minutes, depending on size of model.



Step 3. Export outputs

Right Click on the object called 'Final'. Click on 'Edit this Object'. Click on 'Export to Excel'. Choose a file name and location and click 'Export'.



Step 4. Reading Data

Open the Excel file. The first row is the header with the name of each variable for each column. Each following row is one instance of a possible connection of facilities and reservoirs. All the Columns with variables ending in 'cnt' are counting variables used in the model. You should delete all the columns with these counting variables there are up to 10 of them. Now the variables have the following naming scheme R31 corresponds with Reservoir 3 and Facility 1. 0(zero) means not connected and 1(one)means connected. (Further example, R52 corresponds with Reservoir 5 and Facility 2). To make the actual connection matrix, the first 5 columns would be the first row of the matrix, the second 5 columns (6-10) would be the second row of the matrix,...etc until you have a 5x5 matrix where the columns are reservoirs and the rows are facilities.

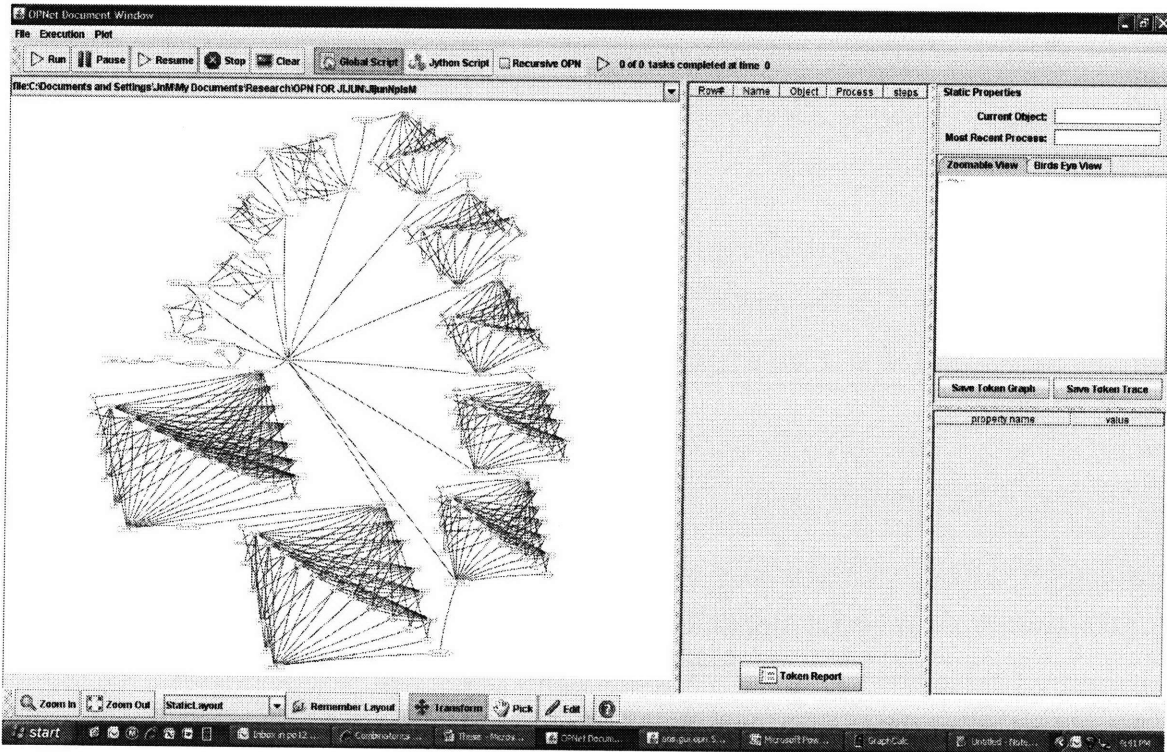
APPENDIX 8. Global Script code for N plus M generator

```
#Sets number of elements (N+M) 2-13
Fmax=8
#sets max number of connections for @
# element 1-12
R1max=1
R2max=1
R3max=1
R4max=1
R5max=1
R6max=1
R7max=1
R8max=1
R9max=12
R10max=2
R11max=2
R12max=2
R13max=2
#sets elements that can't go together
# example r1r2=1 (res1 and res2 can't go together)
# example r2r3=0 (res2 and res3 can go together)
r1r2=0
r1r3=0
r1r4=0
r1r5=0
r1r6=0
r1r7=0
r1r8=0
r1r9=0
r1r10=0
r1r11=0
r1r12=0
r1r13=0
r2r3=0
r2r4=0
r2r5=0
r2r6=0
r2r7=0
r2r8=0
r2r9=0
r2r10=0
r2r11=0
r2r12=0
r2r13=0
r3r4=0
r3r5=0
r3r6=0
r3r7=0
r3r8=0
r3r9=0
r3r10=0
r3r11=0
```

r3r12=0
r3r13=0
r4r5=0
r4r6=0
r4r7=0
r4r8=0
r4r9=0
r4r10=0
r4r11=0
r4r12=0
r4r13=0
r5r6=0
r5r7=0
r5r8=0
r5r9=0
r5r10=0
r5r11=0
r5r12=0
r5r13=0
r6r7=0
r6r8=0
r6r9=0
r6r10=0
r6r11=0
r6r12=0
r6r13=0
r7r8=0
r7r9=0
r7r10=0
r7r11=0
r7r12=0
r7r13=0
r8r9=0
r8r10=0
r8r11=0
r8r12=0
r8r13=0
r9r10=0
r9r11=0
r9r12=0
r9r13=0
r10r11=0
r10r12=0
r10r13=0
r11r12=0
r11r13=0
r12r13=0

APPENDIX 9. Operation of the N Plus M connection generator model

When first opening the model, if you see no model in the window, zoom way out with the mouse roller until you see the model then zoom in on it. For some reason the graphics does not always center the model in the window.



Step 1. Set constraints with the front end

In order to better visualize and set the constraints for a specific problem, an Excel spreadsheet was made to act as a front-end to the N plus M generator. Without the front end the user must type coordinates in the Global Script before each run of the generator, but the Global Script interface is not very user-friendly. Also, without front end there are 78 flag variables in the Global Script that must be set to implement the constraints on specific connections. Each flag variable equaling one or zero and each having an encoded representation of the connection it represents. It is not easy for a user to visualize how he or she is setting up the generator and the connection matrix for a given problem.

On the left side of the front-end spreadsheet is the grid which represents the connection matrix of which the user is trying to generate instances. In this grid, the user places X's or leaves blanks in the positions corresponding to the different connections being considered. Placing an X in a position in the grid means the elements in that row and column can be connected. Leaving a blank in a position in the grid means the elements in that row and column cannot be

connected. The user can easily understand and see how the connection problem is being set up. The X's and blanks in the grid change the values of the various flag variables to the correct values of 1 or 0 in the middle column of equations. On the right side of the spreadsheet, the user can enter the latitude and longitude coordinates (in degrees) for each element. Then some maximum distance (in kilometers) that is allowed for a connection to take place can be specified. The user then clicks on a macro button and the spreadsheet figures the distance between each pair of elements (using a sphere representation of the earth). It flags which of those pairs are inside and outside the maximum allowable distance and places X's and blanks in the correct corresponding positions in the grid. The user can then change the grid as necessary for the problem. Once the user is satisfied the correct connections are allowed and disallowed, the column of flag variable equations is copied from the spreadsheet and pasted into the Global Script. The generator is then ready to run and the user can be sure the correct connections will be allowed and disallowed.

Grid allows setting of non-connectable elements by leaving blank spaces and putting in X's

Coordinates for each element are input, then using some maximum distance non-connectable elements are set in the grid

Code is prepared to copy in to OPN Global Script to run the model with the appropriate constraints

NAME	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
Item 1	R1												
Item 2	R2	x											
Item 3	R3		x										
Item 4	R4	x		x									
Item 5	R5		x		x								
Item 6	R6	x		x		x							
Item 7	R7		x		x		x						
Item 8	R8	x		x		x		x					
Item 9	R9		x		x		x		x				
Item 10	R10	x		x		x		x		x			
Item 11	R11		x		x		x		x		x		
Item 12	R12	x		x		x		x		x		x	
Item 13	R13		x		x		x		x		x		x

```

r1r2=0;
r1r3=1;
r1r4=0;
r1r5=1;
r1r6=0;
r1r7=1;
r1r8=0;
r1r9=1;
r1r10=0;
r1r11=1;
r1r12=0;
r1r13=1;
r2r3=0;
r2r4=1;
r2r5=0;
r2r6=1;
r2r7=0;
r2r8=1;
r2r9=0;
r2r10=1;
r2r11=0;
r2r12=1;
r2r13=0;
r3r4=0;
r3r5=1;

```

Coordinates (degrees)				
Item 1	6.764739	S	41.912222	E
Item 2	6.774739	S	41.902222	E
Item 3	6.784739	S	41.892222	E
Item 4	6.794739	S	41.882222	E
Item 5	6.804739	S	41.872222	E
Item 6	6.724739	S	41.862222	E
Item 7	6.734739	S	41.962222	E
Item 8	6.744739	S	41.972222	E
Item 9	6.754739	S	41.982222	E
Item 10	6.814739	S	41.952222	E
Item 11	6.824739	S	41.942222	E
Item 12	6.834739	S	41.932222	E
Item 13	6.844739	S	41.922222	E

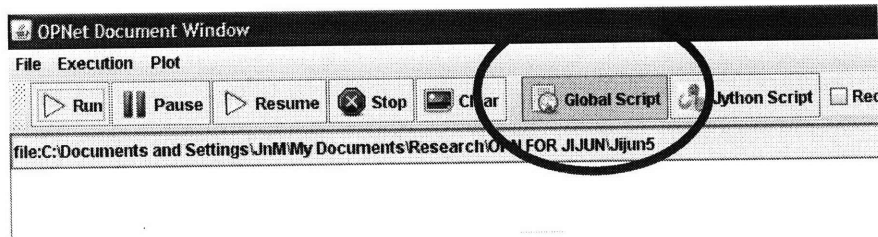
Max Distance 6 KM

Set X's From Dist

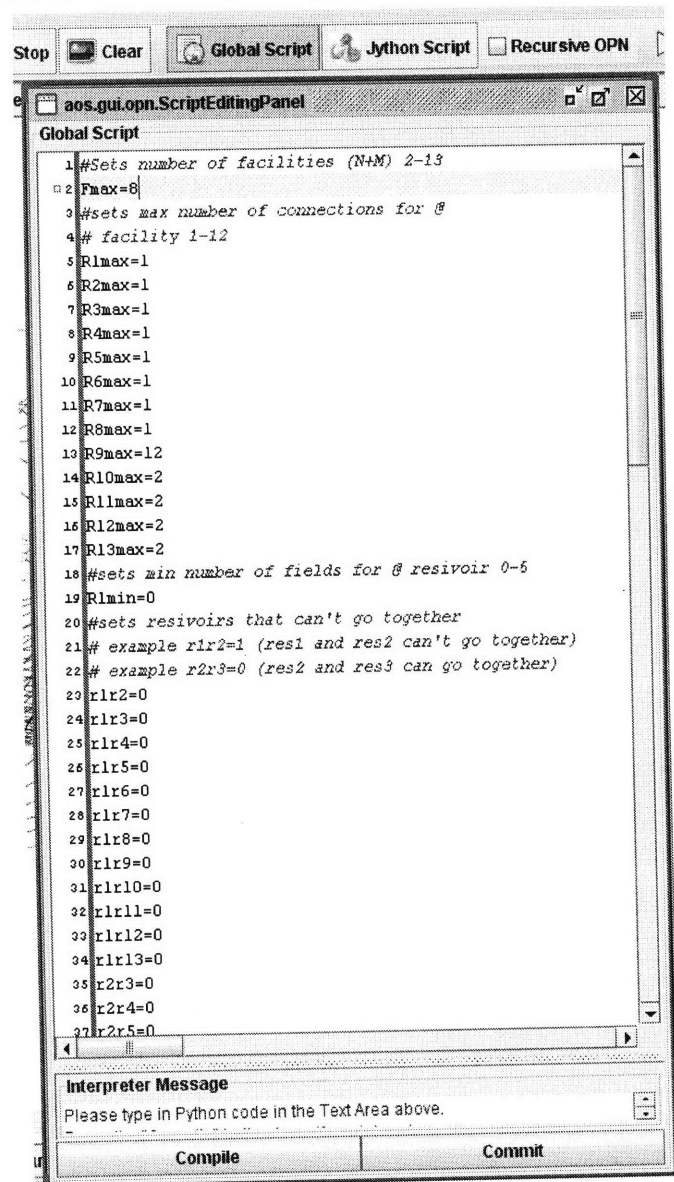
- You can change the names of the items in the NAME column
- Place an X in a grid space that corresponds to an allowed connection
- Leave a blank space in the grid for a disallowed connection
- Enter the coordinates in degrees in the coordinates boxes for each item
- Set N or S or E or W (use caps)
- Set the max distance allowed between elements and click the "Set Xs" Button
- When grid is completely set, copy the all the filled cells from r1r2 to r12r13
- Then paste them in the appropriate spot in the "Global Script" of the OPN Model

Step 2. Set constraints in Global Script

Click on Global Script.



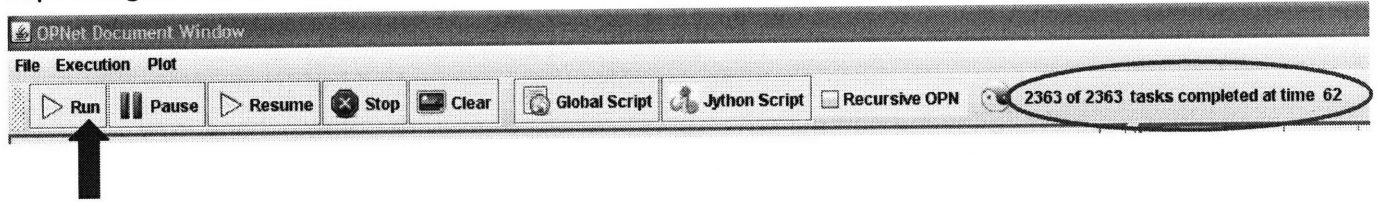
Set constraints by changing variables.



Comments in the code above each constraint should be self-explanatory. When finished setting constraints click 'compile' and then 'commit'.

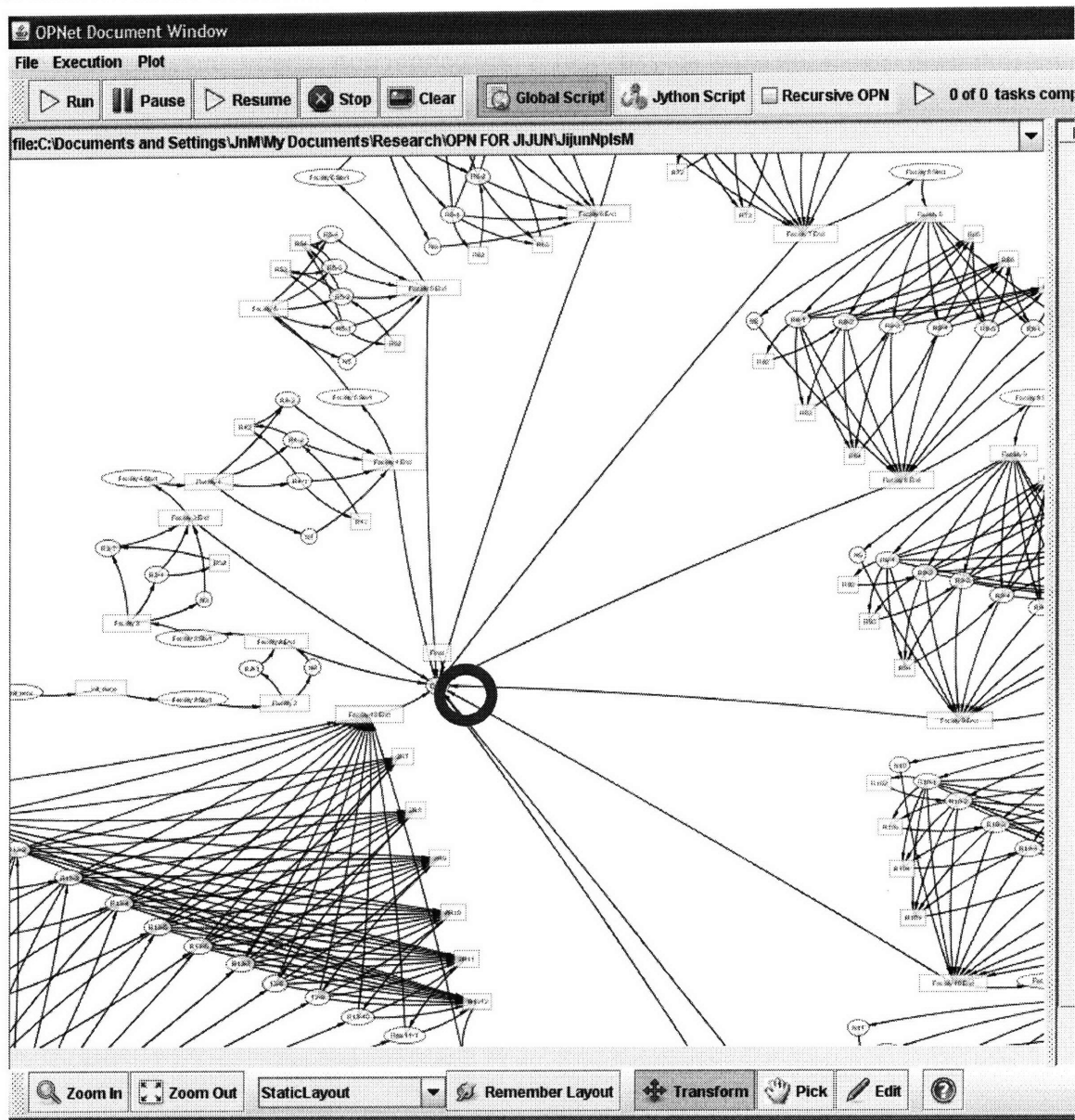
Step 3. Run OPN

Click Run. Wait for counter to stop running. Run time range is 3-seconds to 60 plus minutes, depending on size of model.



Step 4. Export outputs

Right Click on the object called 'Final'. Click on 'Edit this Object'. Click on 'Export to Excel'. Choose a file name and location and click 'Export'.



Step 5. Reading Data

Open the Excel file. The first row is the header with the name of each variable for each column. Each following row is one instance of a possible connection of facilities and reservoirs. All the Columns with variables ending in 'cnt' are counting variables used in the model. You should delete all the columns with these counting variables there are up to 13 of them. Now the variables have the following naming scheme R31 corresponds with Column 3 and Row 1 in the connection matrix. 0(zero) means not connected and 1(one)means connected. (Further example R52 corresponds with Column 5 and Row 2 in the connection matrix). (NOTE: This naming convention is different than the convention in the N by M to generator.) To make the actual connection matrix, move the elements of the matrix to their respective positions in the matrix based on the name of the column in the excel spreadsheet in which they fall. Remember that each row in the excel spreadsheet is one connection scheme instance, so each row of the spreadsheet will form it's own lower triangular connection matrix.

BIBLIOGRAPHY

1. **Robinson, Bob; Barr, Alastair; Flanagan, Tomas; Schroeter, Robert; et al.** *Conversations with BP Sponsors*. September 2006 - May 2008.
2. **Harel, D.** Statecharts: a visual formalism for complex systems. *Science of Computer Programming*. 8, 1987.
3. **Jacobson, I., et al.** *Object Oriented Software Engineering, A User Case Driven Approach*. Reading, MA : Addison Wesley, 1992.
4. **De Marco, T.** *Structured Analysis and System Specification*. New York : Yourdon Press, 1978.
5. **Rumbaugh, J., et al.** *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ : Prentice Hall, 1991.
6. **Booch, G.** *Object-Oriented Analysis and Design, 2nd edn.* Redwood City, CA : Benjamin Cummings, 1994.
7. **Coad, P. and Yourdon, E.** *Object Oriented Analysis, 2nd edn.* Englewood Cliffs, NJ : Prentice Hall, 1991.
8. **Embley, D. W., Kurtz, B. D. and Woodfield, S. N.** *Object-oriented Systems Analysis*. Englewood Cliffs, NJ : Prentice Hall, 1992.
9. **Shlaer, S. and Mellor, S. J.** *Object-Oriented System Analysis*. Englewood Cliffs, NJ : Prentice Hall, 1988.
10. **Dori, Dov.** *Object-Process Methodology*. Springer, 2002.
11. **Crawley, Edward.** *ESD.34 System Architecture*. [Course Notes]. MIT, January 2006-December 2007.
12. **Koo, Ben.** *A Meta-Language for Systems Architecting*. Engineering Systems, MIT. Cambridge, MA : , 2005. PhD Thesis.
13. **Simmons, Willard.** *A Framework for Decision Support in Systems Architecting*, Cambridge, MA : PhD Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, February 2008.

14. **Simmons, Willard, Koo, Ben and Crawley, Edward.** Space Systems Architecting Using Meta-Language. *56th International Astronautical Congress*. 2005.
15. **Simmons, W., Koo, B. and Crawley, E.** Architecture Generation for Moon-Mars Exploration Using an Executable Meta-Language. *AIAA Space*. 2005.
16. **Crawley, Edward and Simmons, Willard.** *Towards a Formalism for System Architecture -- From Value to Architecture*. MIT. October, 2006.
17. **Leffler, William L., Pattarozzi, Richard and Sterling, Gordon.** *Deepwater -- Petroleum Exploration and Production -- A Nontechnical Guide*. PennWell, 2003.
18. **Raymond, Martin S. and Leffler, William L.** *Oil and Gas Production in Nontechnical Language*. PennWell, 2006.
19. **Rozenblum, Ziv.** *Object-Process Networks & Object Process Diagrams -- Implimentation Issues of Oil Exploration Systems*. MIT. Cambridge, MA : s.n., 2007. Thesis.
20. **Maurer, M.** *Structural Awareness in Complex Product Design*. MUNICH : TU MUNICH, 2007.
21. **de Weck, Oli and Lin, Jijun.** Conversations about MIT research project sponsored by BP. *BP Tieback Flexibility Case Study*. November 2007-May 2008.
22. **Cameron, Bruce G.** *A Quantitative Method For Comparing Benefit Across Exploration Architectures*. MIT. Cambridge, MA : s.n., 2007. Thesis.
23. **Maier, Mark W. and Rechtin, Eberhardt.** *The Art of Systems Architecting*. Boca Raton, FL : CRC Press, 2002.
24. **Rechtin, E.** *Systems Architecting: Creating and Building Complex Systems*. Englewood Cliffs, NJ : Prentice Hall, 1991.
25. **Sage, Andrew P. and Lynch, Charles L.** Systems Integration and Architecting: An Overview of Principles, Practices, and Perspectives. *Systems Engineering* 1. 1998, Vol. no. 3.