Object-Process Networks & Object-Process Diagrams - Implementation Issues for Oil Exploration Systems

by

Ziv Rozenblum

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

May 2007

Signature of Author_____

Ziv Rozenblum
System Design and Management Program
February 2007

Certified by_____

Edward Crawley
Thesis Supervisor
Engineering Systems Division

Certified by_____

Patrick Hale
System Design and Management Program
Director

Object-Process Networks & Object-Process Diagrams - Implementation Issues for Oil Exploration Systems

by

Ziv Rozenblum

Submitted to the System Design and Management Program
on May 2007, in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Management

Abstract

This thesis discusses different ways to increase the ability of OPN (Object-Process-Network), a system-architecture meta-language, to model complex systems and identify architectures to address those systems. It proposes ways to generate “out-of-the-box” solutions to technical problems, to weight a portfolio of objects as a possible way to achieve system functionality, to examine which of the possible subsystems should be within the system boundary and to rank architectures based on the value they create for the system stakeholders. The thesis also provides case studies to examine the practical aspects of those proposals.

Thesis Supervisor: Edward Crawley
Title: Professor, Engineering Systems Division

Acknowledgements

I had many figures affecting my life and my work in the recent years. I feel grateful and fortunate to get their advice and to listen to their wisdom.

A great inspiration and guidance was given to me in the past year and a half by Professor Edward Crawley, my thesis advisor. He was able to influence me with his wisdom and grace in many areas. He has truly been my mentor during this whole process. I also want to thank my research partners, James T. Keller and Wen Feng, for sharing all those hours of exploration and for making it such an enjoyable journey. My sincere gratitude goes to Bob Robinson and Alastair Barr for shaping this remarkable cooperation.

I feel so lucky to have so many teachers who were kind enough to share their valuable knowledge with me. I would like to thank Professors David Simchi-Levi and Don Clausing as well as Professors Joseph Lassiter and Frank Cespedes from HBS. Special thank to my teachers Professors Thomas Allen and Oliver de-Weck for listening, advising and directing.

I owe a debt of gratitude to a unique group of friends, peers and cohort. Thank you for being there for me, for challenging me and expanding my horizons. I learned from them so many things that are much larger than this thesis.

Finally, I feel grateful to be the second half of my wife, Orit Beitler. Anyone who knows us understands the effect she had over my life. Her endless support and advice played a huge role in this journey. I owe her special thanks for encouraging me to pursue my dream. Another special thanks to Mr. Uzi Beitler with his endless energy to run, arrange, support, listen and advise. I would like to finish by thanking my parents, Sara and Yossi, for planting and nurturing those seeds that bloomed to become what I am.

Contents

List of Figures

List of Tables

1 Introduction

This thesis summarizes a study aimed to support BP in its multi-year, multi-billion dollar development effort to explore the best way to extract oil from a specific deep ocean reservoir. This reservoir imposes many technological difficulties due to the arctic and seismic nature of the environment. The unstable political environment, as well as the abundance of stakeholders, imposes additional difficulties. Those stakeholders besides having several attributes that define their value are often conflicted in how they estimate those attributes. In addition to profit, the different stakeholders could value political power, environmental friendliness, time, etc. For example, one stakeholder might aspire for political power on his side whereas the other would prefer increasing his own political power. Finding the right architecture is a delicate task of dealing with a complex system and turning it into a feasible structure while balancing among all the different stakeholders.

The objective of the research was to support the BP team through the development of a system architecting tool and methodology. This tool was produced with the intention of being generally applicable to BP's oil exploration and production system architecture decisions. Additionally, this function-based system-architecting tool was built in such a way that it could further aid the leading team in identifying creative, "out of the box" solutions.

The research was based on a modeling approach to the development of systems that describe both their structure and behavior in a single model. That approach, called OPM (Object Process Methodology), was developed by Professor Dov Dori [4]. OPM uses a graphic tool, called OPD (Object–Process Diagrams), as a single model of the structural, functional, and dynamic system aspects. Furthermore, the dynamic architecting tasks were done using OPN (Object Process Network), a meta-language developed by Professor Edward Crawley and Professor Ben Koo [1], [7], that assisted in creating and evaluating the different architecture options.

This thesis will focus on the first phase of the project, aimed at creating a working infrastructure. It will provide a summary of the issues that were raised during the

research, which are applicable to system architecture in general. Each issue will be accompanied by a specific example from the BP project.

Chapter Two discusses the concepts, importance and different definitions of the terms systems and systems architecture. Additionally it elaborates on the importance of models in the system architecture context and focuses on the tools and approaches that will be used in this thesis.

Chapters Three to Six discuss some of the issues that were raised when trying to structure BP's exploration system into the OPD and OPN framework. The suggested solutions for those issues will usually have broader implications than for OPN per se. Thus, for each issue, I will describe the proposed solution and its application for the current research, and discuss possible implications for other aspects of System Architecture.

Chapter Three discusses an algorithm that can significantly increase the ability of OPN to simulate real life decisions faced by an architect. It does this by allowing some flexibly to the model to decide on the best combination of forms that maximizes stakeholder's value without being constrained by the architect. Thus, by utilizing that functionality correctly, a full gamut of solutions can be explored, answering questions like, should I build that system from small number of high capacity forms, a large number of low/mid capacity forms or a combination?

Chapter Four proposes a method to generate out-of-the-box solutions. In our research we tried to generate out of the box solutions taking top-to-bottom approach. This method is both theoretical and practical, and thus can span an entire process from raising the problem to finding the right solution. Additionally, it is not limited by current practices and thus can offer new ways to deal with specific problems.

Chapter Five deals with the possibility to change the system boundary as the system model is being built. Finding the best architecture to offer value to stakeholders can be affected by the definition of the system boundary. Often a lean system can offer greater value for the invested resources than a comprehensive one. This chapter proposes having an entity on the boundary layer, representing change of ownership. That entity will offer value to the system architect in the form of a possible formulation of the boundary and easier definition of interfaces.

Chapter Six presents four levels of possible connections between the stakeholder's model and the process-object model. Those connections are needed in order to "measure" the value each architecture generates for the system's stakeholders. It starts with two separate models, connected by human interface, and ends with a suggestion for fully coupled models.

Finally, the thesis is concluded with a short summary and a discussion of further research.

2 Background

2.1 System

Definitions

There are several definitions of a system. This section will briefly discuss some of the definitions, select the one to be used at the rest of the thesis, and describe the rationale behind that selection.

Crawley [1] defines a system as:

> *"A set of interrelated elements which perform a function, whose functionality is greater than the sum of the parts."*

This definition is supported by Dori [4]:

> *"A System is an object that carries out or supports a significant function."*

According to those definitions, the connection between elements/objects and their cross-interfaces to function is in the heart of system. I find this connection very important since it creates the link between the systems and the system architect. The system architect can affect/control the elements/objects and sometimes their inter-relations and thus can affect the functionality of the system. It is important to note that the system architect can affect the type of functionality that will emerge from a system as well as the "goodness" of that functionality, by selecting one specific form over another. Of course "goodness" of functionality is a subjective matter. A discussion of how to measure it will appear later in this thesis.

Maier and Rechtin [3] give another view of the definition of a system:

> *"system is a collection of different things, which together produce results unachievable by the elements alone."*

This definition enlarges the previous definition to include the context in which the system exists. Instead of viewing the system as a set of elements/objects that interlink

and perform functions, it is being perceived as a thing that achieves results. There are two differences from the previous definitions that should be emphasized. The first is that result is used instead of functionality, which brings the context of the system into the definition. The second difference is that a system is defined by Maier and Rechtin as a collection of different things, which is basically more comprehensive than objects as was defined earlier (wishes for example, can be categorized as a thing but not as an object).

System definition used in this thesis

In this thesis I will use Crawley's definition of a system for two main reasons. The first is that although context and results are important parts of a system's success, I believe that systems exist even before achieving results. It is more the potential to perform functions that define the system. In other words, a system that was never put into action is still a system.

The second reason relates to the term "collection of different things" that was used by Maier and Rechtin [3] to define the embodiment of a system. I believe that this term is too broad. Some non-physical "elements" like feelings should not be part of the building blocks of a system, especially in the system-engineering context.

2.2 System architecture

Definitions

The term system architecture, like the word system, has many definitions. I will focus on two groups of definitions that differ in their view of the emergence of function and the importance of concept. The first group defines system in the context of the elements that build it. The second enlarges the definition to include the function the system achieves.

I will use two definitions as representative of the first group. Frey [12] defines system architecture as:

"The structure, arrangements or configuration of system elements and their internal relationships necessary to satisfy constraints and requirements."

Ulrich and Eppinger [13] support that definition while emphasizing the grouping of those elements. They define system architecture as:

"The arrangement of the functional elements into physical blocks."

Both definitions focus on the physical[1] layer as the essence of system architecture. A different view is proposed by a second group of definitions. Two complementary definitions are proposed by Crawley [1]:

"The embodiment of concept, and the allocation of physical/informational function to elements of form, and definition of interfaces among the elements and with the surrounding context."

And:

"architecture is the details of the assignment of function to form, and the definition of interfaces."

This definition emphasizes the embodiment of a concept through the usage of elements of form. System architecture is the art and science of the assignment of those elements to achieve the required functionality.

[1] I use physical in its wider context to include flows and stocks like information that can be controlled by the system architect.

Again, Dori [4] supports that definition:

"System architecture is the overall system's structure-behavior combination, which enables it to attain its functions while embodying the architect's concept."

Dori also suggests a more detailed view of concepts and function in the context of system architecture:

"Concept is the system architect's strategy for a system's architecture."

And:

"Function is an attribute of object that describes the rationale behinds its existence, the intent for which it was built, the purpose for which it exists, the goal it serves, or the set of phenomena or behaviors it exhibits."

System architecture definition used in this thesis

In this thesis, I will use the second group of definitions. I believe that an important part of system architecture is the connection between the physical layer of objects and the layer of functions performed by the system. This connection is especially important to the system architect since it allows a constant check of the "goodness" of the system.

2.3 System modeling

Systems can be viewed from different aspects by different stakeholders. Clients are interested in the functionality of the system whereas the designers are more interested in the forms that build the system. A system architect needs to communicate with all related functions in order to be able to discuss their perspectives of the system. Models allow that kind of communication. According to Dori [4]:

"a model is an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system."

Thus, models are a possible way to project the system through highlighting different aspects of it.

Maier and Rechtin [3] support this view:

> *"Models are the primary means of communication with clients, builders, and users; models are the language of the architect."*

Following those definitions, the roles of models in the system context include [3]:

1. Communication with clients, users and builders.
2. Maintain system integrity through coordination of design activities.
3. Assisting design by providing templates and organizing and recording decisions.
4. Explore and manipulate solution parameters and characteristics; guiding and recording aggregation and decomposition of system functions, components, and objects.
5. Performance prediction and identification of critical system elements.
6. Provide acceptance criteria for certification for use.

A model presents a view of the system. A view is defined by Maier and Rechtin [3] as:

> *"A view is a representation of a system from the perspective or related concerns or issues."*

Models can be textual or visual representations of the system based on the context the model is being built for. There are six types of possible views [3]:

TABLE 1: Types of model views

Model view	Description
Purpose/objective	What the client wants
Form	What the system is
Behavioral or functional	What the system does
Performance objectives or requirements	How effectively the system does it
Data	The information retained in the system and its interrelationships
Managerial	The process by which the system is constructed and managed

In this thesis I will elaborate about two model views that mainly deal with the form and functional views of a system, called OPM[2] and OPN.

2.4 Types of models used in this thesis

This thesis will focus on two types of models:

1) Object-Process model – Capturing all the different functions performed by the system as well as the possible forms to achieve those functions. As a whole, a complete model represents the entire gamut of forms and functions that can create the relevant system whereas a specific instance of that model represents a specific combination of forms and functions. That specific instance is regarded as a possible architecture.

   It is important to note that the completeness of a model is a subjective thing that depends on the viewpoint of the model builder and user. A complete model consists of all the function within the relevant system boundary including adequate level of decomposition. Additionally, this kind of model can also represent attributes associated with each form, function and their interrelations. These attributes can be used to estimate emergence of functions as well as expected value.

2) Stakeholder's model – Capturing the different stakeholders that are connected to the system, their relative weight, and interrelations as well as the value flow. There are many usages in the system architecture context for the stakeholder model. The first is to use that model to identify critical parameters that are important to system value creation. Those parameters can be used later to rank and evaluate the possible architectures. Other usages might be the ability to quantify the "power balance" between two adjacent stakeholders, or to identify which stakeholder has more effect over the other stakeholder value. A third usage might be finding those stakeholders that have more influence than others on the overall value.

[2] As part of OPM I will also discuss OPD.

Both models can be static – capturing the relevant data or dynamic – capturing the relevant data and exploring the different permutations.

2.5 Object Process Methodology (OPM) and Object–Process Diagram (OPD)

Definition

Object Process Methodology (OPM) [2], [8] is a modeling approach to the development of systems that describes both their structure and behavior in a single model. The basic building blocks of OPM are two equally important classes of entities: objects and processes[3], which are related through a variety of links among them by relationship.

OPM uses a single graphic tool, the Object–Process Diagram (OPD) set, as a single model of the structural, functional, and dynamic system aspects.

OPD Language

OPD uses a set of symbols as a base for its modeling language. Those symbols are used to describe the objects and processes as well as the relations between them [1]. The basic symbols are used to describe objects and processes.

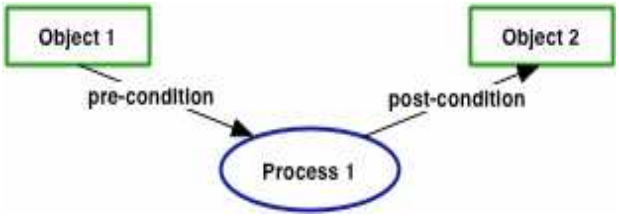


FIGURE 1: OPD representation of objects, processes and their relationship [6].

Further, OPD uses a set of symbols to describe possible relations and links between those objects and processes [1]. Those relations can be found in appendix A along with an example of a model built in OPD.

[3] Objects are things that exist, while processes are things that transform objects [4].

2.6 Object-Process-Network (OPN)

Definition

I use for that section the OPN (Object-Process-Network) definition as it was defined by Crawley [1], [6].

> *"OPN is a visual and computable meta-language that assists with systems architecting tasks"*

And its aim is to:

> *"Improve the thoroughness and efficiency of system architecting, by automating the mechanical tasks in architectural reasoning and model construction, using an executable meta-language."*

Crawley [1] also defines the different usages of OPN. It can be used to describe and partition the space of architectural alternatives, allowing the system architect a clearer view of the system. Additionally it can be used to generate and enumerate the set of instances of feasible system models. That usage is very powerful since it allows the system architect to view a full range of possible architectures that are associated with the system in question. Once those possibilities are created, the architect can simulate and order the performance metrics of the generated models.

OPN uses processes and objects as building blocks to represent systems [7]. Processes capture the transformational activities, whereas objects represent the states of the system.

Following is a screen shot of OPN [6]:

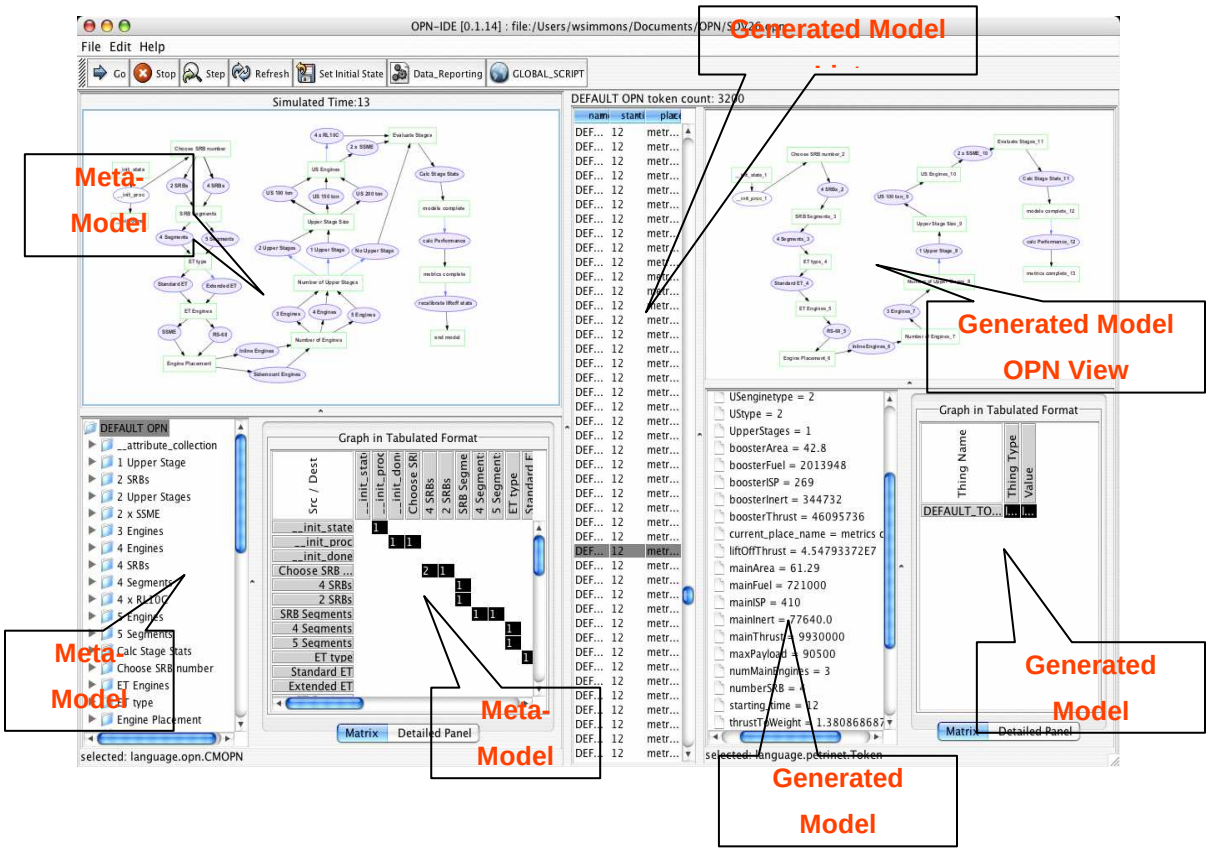


FIGURE 2: OPN screen shot

OPN context used in this thesis

This thesis will focus on one of the benefits that OPN can offer to the system architect - exploring the gamut of possible architectures to build a system and suggest those architectures that offer the highest value to the stakeholders. In that context, OPN can be viewed as a framework that uses the Meta-Model created by the architect to generate the entire gamut of possibilities of architectures. This Meta-Model contains all the functions of the system (down to a certain level of decomposition) as well as all the different form possibilities to perform those functions (see FIGURE 1). While generating each of the possible architectures, the framework also calculates its expected value to the stakeholders. Value can come from the specific forms that are selected or from the interconnections between the forms. This value allows the architect to rank the generated architectures according to the value they create for the stakeholders.

Another OPN usage that will be used in this thesis is the generation and valuation of stakeholder's maps. In that case the meta-model created by the architect represents the inner-relations between the stakeholders. The architect can use OPN to value each of the connections between the stakeholders, in search of those who have the highest influence on the overall value. Focusing on these will simplify the stakeholder's model and allow the architect to explore effective connections between architecture and value.

3 How can an iterative process be implemented in OPN?

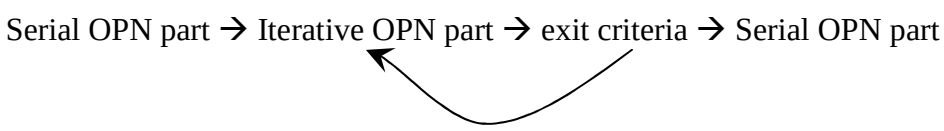
3.1 Introduction

OPN and OPD are structured in such a way that one process and form follows the other until achieving the intent of the system. In OPN the execution of the Meta-Language is done by following one process and object at a time. This section will deal with the issue of building a loop process into that structure. That will allow OPN to go upstream the object-process path. In particular it will focus on loops as a way to simulate an architecture where forms can repeat themselves an unpredicted number of times. That can be used to solve bottlenecks in the system, to increase the range of solutions tested or to allow the model ad-hoc adjustments.

It is important to note that those issues can be addressed to a certain level without incorporating loops. The architect can "hard code" all the different repeats and possibilities into the model. That method, while supplying the above requirement, has some drawbacks. It reduces the "out-of-the-box" creativity embedded into OPN and there is a potential cost and increased computational time.

This chapter will present a possible algorithm that, when implemented in OPN, will support an iterative process. This algorithm can be incorporated into OPN without any required changes at the software level. An example of an implementation will be presented at the end of the chapter.

The algorithm is structured around a phase termed "exit criteria", which is basically a binary decision – whether the part that was marked as repeatable should be repeated once more. Thus, the OPN structure will be iterative in the following way:



The iterative OPN part will repeat itself, until the exit criteria are met.

3.2 The Problem

OPN allows for maximum flexibility in selecting forms to increase stakeholder's value. Any combination of forms is possible as long as the ratio of one form per function is maintained. The issue is that often that flexibility is bounded by bottlenecks somewhere along the flow of form selection.

For example, in building an oil exploration system, the oil company is usually being given an access to the reservoir for a limited number of years. Since the oil company is interested in extracting as much oil as possible within that time period, the capacity of the exploration system is a very important attribute to consider in the architecture. For simplicity, we can assume that there are four basic functions to consider: extract, treat, move and store. The overall capacity of the system will be determined by the lowest capacity form associated with one of those functions. That means that this lowest capacity form becomes the bottleneck of the system. In real life, when reaching that capacity, an additional form might be added to the system (to perform the same function) in order to release the bottleneck. That way, there is more than one form associated with the specific function.

Bottlenecks are not the only cases where multiple forms might be considered. Increased stakeholder's value is another possible reason. Even in cases where one form can achieve the required capacity, the system architect should consider using other combinations to generate more value to the system stakeholders. For example, looking again at the oil exploration system, the system architect might consider the following trade-off for a treating sub-system:

1. Use one big and expensive system that has the overall required capacity.
2. Use two small and less expensive systems that together have the required capacity.

The decision is not only price-related. Using two forms will probably increase the overall utilization of the subsystem but might have a negative effect on schedule and price. Making that kind of decision up front might be very complicated, especially in cases where there are more than two possibilities.

Another possible reason to use multiple forms is a desire for redundancy. The system architect might consider adding other forms to create redundancy for critical subsystems or for the entire system.

As mentioned, the solution is basically to add additional forms under the existing function. The end result is that some functions will have more than one form targeted at achieving that function. That process can be implemented in OPN even without loops. The algorithm to do that would require some preparations by the system architect:

1. Decide in advance which forms can be used more than once and how many times.
2. Incorporate logic into the model **at each intersection** of possible additional form[4] that will support a decision whether that specific form should be added to the specific architecture.

Each of those form selections will include all the possible forms plus a NULL selection. In that case, the OPN will choose the required number of forms, putting NULL in the rest. There are some problems associated with that process. First it substantially increases the complexity and implementation time of the OPN model. It includes not only the additional time required to repeat each form definition, but also the additional complexity associated with the selection criteria that appears in every form selection (answering the question – is that form really needed). Moreover, every change to one of the forms will have to be repeated for all similar forms.

Another drawback is that the architect needs to decide in advance how many times each process/form can repeat itself and in which combinations, an act that can reduce the natural creativity embedded in OPN to present "out-of-the-box" solutions. The reason is that definition of possible number of repeats limits the model to that number and thus might affect the ability of the model to offer solutions that the architect did not think about.

[4] The system architect can also incorporate the logic after the OPN finishes the architecture generation process. In that case the OPN will generate all the possible permutations. Those that violate the logic rules will be screened afterward. I believe that incorporating the logic into the OPN process is more common especially in highly complex systems since that reduces the total running time of the application.

The final drawback relates to increased computational time. Hard coding all the possibilities into the model significantly increases the computational time since there is no flexibility to reduce the model size in lighter cases. The process will consume the same resources even when applied to systems that require fewer repeats.

The following section offers a possible algorithm that can solve these problems.

3.3 The Algorithm

The proposed algorithm will deal with loop conditions at OPN. This algorithm will cover cases where a process gets redirected back by offering exit criteria. These criteria will block the redirection whenever the exit condition is met.

This algorithm will act as follows:

1. Perform the process of adding forms to functions in a serial way until getting to the exit criteria.
2. Check iteration exit condition. This condition can be intent fulfillment level, physical feasibility, etc.
3. In case the iteration exit condition was not met:
    1) The iteration process will start over.
    2) At each stage of the iterative part one of the following can be performed:
        i. Add additional form.
        ii. Do nothing (which is implemented by adding a NULL form).

    The decision what to do will depend on the result of the iteration exit condition as well as the inner logic of the form selection process as was embedded in the OPN.
4. In case the iteration exit condition was met, the OPN process will continue without performing additional iteration.

Building the model requires a preparation phase:

1. Define all the functions that might require additional forms and add them to a loop within OPN.
2. Define the criteria that will dictate how many forms are required for each function.

FIGURE 3 shows the architecture creation phase:

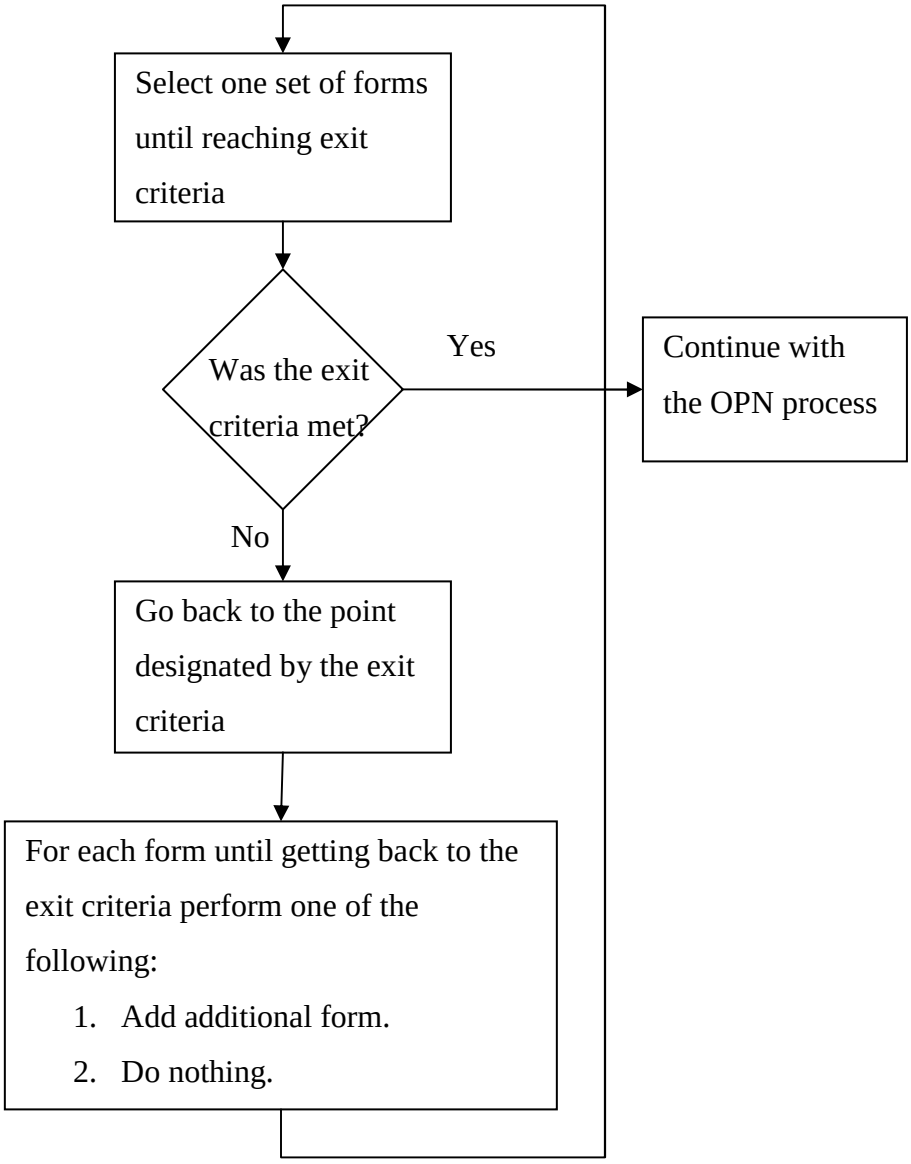


FIGURE 3: Algorithm for implementing loops in OPN

There are some important points to emphasize about that algorithm. Each loop is associated with some forms. The criteria of the loop should appear right after the last form associated with that loop, because the iteration process consumes a lot of resources and thus should be kept to minimum number of forms selected. From that exact reason,

the loop dominated by the exit criteria should be kept as minimal as possible. Thus, in case the exit criteria were not met, the process should return the minimal number of steps that still includes all the relevant forms.

3.4 Example

The problem

For that example, I assume a simplified model of oil exploration system where the only important subsystems are those that perform extracting and treating. They determine the cost of the system as well as the capacity and the building duration. I further assume that there are three potential forms that can be associated with each of those subsystems. For simplicity I call them small, medium and large. Following is a summary of the input parameters to the problem:

Possible Forms:

Extracting Forms:

TABLE 2: Extracting forms parameters

Name	Barrel capacity (M barrels/year)	Building time (years)	Building cost ($M)
Small Size	120	3	800
Mid Size	160	4	1600
Large Size	200	6	2000

Treating Forms:

TABLE 3: Treating forms parameters

Name	Barrel capacity (M barrels/year)	Building time (years)	Building cost ($M)
Small Size	140	4	1200
Mid Size	200	5	1600
Large Size	220	7	2200

Other Inputs:

Field capacity: 3000M Barrels.

Total leasing time: 20 years.

Price per barrel: $15[5]

Discount rate: 7% year.

The goal is to find the right portfolio of forms that will optimize the stakeholder's profit. In that problem we will treat profit as the only parameter that determines stakeholder value and assume that all stakeholders are interested in as much profit as possible.

The tradeoff for finding the right portfolio is building cost vs. expected profit. Thus adding treating systems for example will increase the expected oil production (and the expected profit) up to a certain level controlled by the extracting system capacity and maximum field capacity. On the other hand, the additional treating systems will cost additional money and will consume more building time. Another issue to consider is the total time of the lease that affects the total time the company has to extract oil from the field.

I decided to use NPV (Net Present Value) to calculate the profit of the system. The formulation I used for that as well as the mathematical formulation of the entire problem can be seen in appendix C.

As a matter of fact, this is a simplified version of a bigger optimization problem aimed at finding the right portfolio to maximize stakeholder's value. There are actually several ways to solve that kind of optimization problem. One of the possibilities is to use an optimization algorithm. Since most of the problems are non linear in nature and most of the parameters are discrete, a genetic algorithm might be helpful. Another option might be to try to predict the connection between profit and the different parameters (in our case the portfolio of extracting and treating products) using methods like Design Of Experiments.

[5] Assuming operational costs are negligible, price per barrel will be considered as profit per barrel.

This example will show an alternative way, using an iterative process at OPN to explore the entire gamut of possible solutions and find the best portfolio that maximizes the profit of the system.

Exploring some possible solutions to the problem using spreadsheet

There are several important facts to consider regarding the problem:

1. There are several possible treating and extracting forms, each with its own cost, time and capacity characteristics.
2. Each treating and extracting subsystem can be constructed of one or more of those possible products.
3. In case more than one form was used for either the extracting or treating subsystems, the building of those forms is done in a serial way. Building one form will start only after the building of the previous one ends[6].
4. Both treating and extracting can start only after all the forms are ready.

Following are three possible portfolios of Extracting and Treating[7]:

TABLE 4: Portfolio of three possible solutions for a simplified oil exploration system

	Option 1	Option 2	Option 3
# of small extracting facilities		2	
# of Medium extracting facilities			2
# of Large extracting facilities	1		
# of small treating facilities		2	
# of Medium treating facilities			2
# of Large treating facilities	1		

[6] A comparison to a model where forms can be built in parallel can be found at appendix E.

[7] Those are only some of the possible solutions, as will be demonstrated later

Based on that portfolio, following are the calculations of cost, building time, yearly capacity, yearly cost[8] (during the years when the subsystems are built) and the yearly profit[9]. Furthermore, based on the field capacity, it is possible to calculate whether within the leasing time all the oil in the field will be extracted[10]:

TABLE 5: Economic figures for the three possible solutions

	Option 1	Option 2	Option3
Total cost	4200	4000	6400
Years to Build	7	8	10
Total yearly capacity	200	240	320
Total leash time capacity	2600	2880	3200
Did it reach max capacity	No	No	YES
Yearly cost	600.0	500.0	640.0
Yearly Profit	3000.0	3600.0	4500.0

Incorporating those figures into an NPV table:

TABLE 6: Discounted yearly income for the three possible solutions

Year	Option 1 - discounted cost (M$)	Option 2 - discounted cost (M$)	Option 3 - discounted cost (M$)
1	-560.7	-467.3	-598.1
2	-524.1	-436.7	-559.0
3	-489.8	-408.1	-522.4
4	-457.7	-381.4	-488.3
5	-427.8	-356.5	-456.3
6	-399.8	-333.2	-426.5
7	-373.6	-311.4	-398.6
8	1746.0	-291.0	-372.5
9	1631.8	1958.2	-348.1
10	1525.0	1830.1	-325.3
11	1425.3	1710.3	2137.9
12	1332.0	1598.4	1998.1
13	1244.9	1493.9	1867.3
14	1163.5	1396.1	1745.2
15	1087.3	1304.8	1631.0
16	1016.2	1219.4	1524.3
17	949.7	1139.7	1424.6
18	887.6	1065.1	1331.4
19	829.5	995.4	1244.3
20	775.3	930.3	1162.9

[8] Yearly Cost = Total Cost / Years To Build.

[9] Yearly Profit = (Yearly Production X Price per barrel) / # of Production Years.

[10] The calculation formula to find the entire oil extraction potential is:

NUMBER OF PRODUCTION YEARS X YEARLY CAPACITY

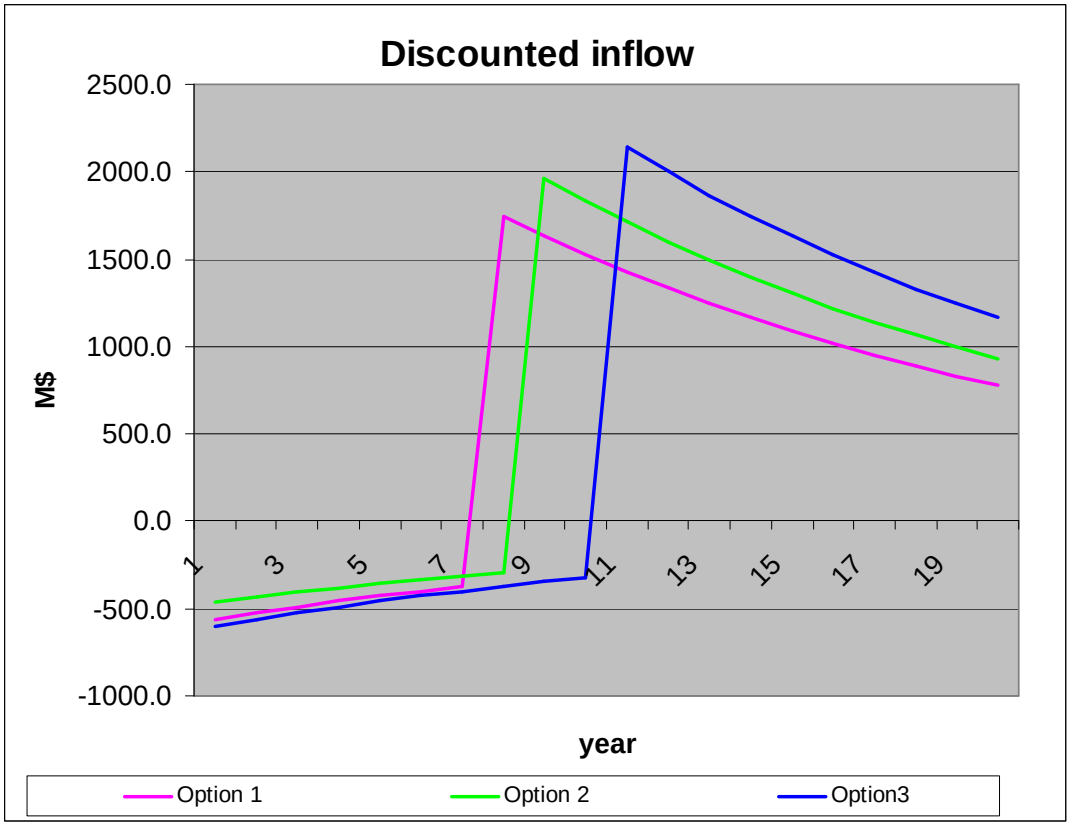


FIGURE 4: Cash flow for the three treating and extracting solutions

As can be seen, option 1 reaches production fastest (after 8 years) but extracts the minimum amount of oil from the field[11] within the leasing time, relative to the other options. Option 2 extract more on average but does it for fewer years. Option 3 reaches the maximum capacity of the field but it starts extracting only after 11 years.

The anticipated profit of each of the options is:

TABLE 7: Anticipated profit for the three treating and extracting solutions

	Option 1	Option 2	Option3
Anticipated discounted profit after 20 years ($M)	12,380.6	13,656.1	11,571.9

[11] There is a direct relation between the positive cash inflow and the oil capacity

Thus, looking at profit alone, option 2 seems the preferred architecture. Incorporating other real-life consideration requires enlarging this range of solutions to include all possible forms portfolios and other considerations beside profit, such as Risk, technology readiness, preferred forms (as a means to gain political power, for example), etc. That might turn into a non-linear optimized problem, which is much more difficult to solve in the above method.

Implementing in OPN

The following OPN model will calculate the optimized portfolio of extracting and treating facilities after finding all the possible combinations. The value calculation will be based on NPV although any other consideration can be incorporated into the model.

In order to build that mode an exit criterion should be defined. That will signal the process that there is no point in adding more extracting and/or treating forms.

Exit criteria:

Enough capacity was built to exceed the potential capacity of the field

OR

Building time exceeds the total leasing time (divided by some factor)

The OPN implementation looks as follows:

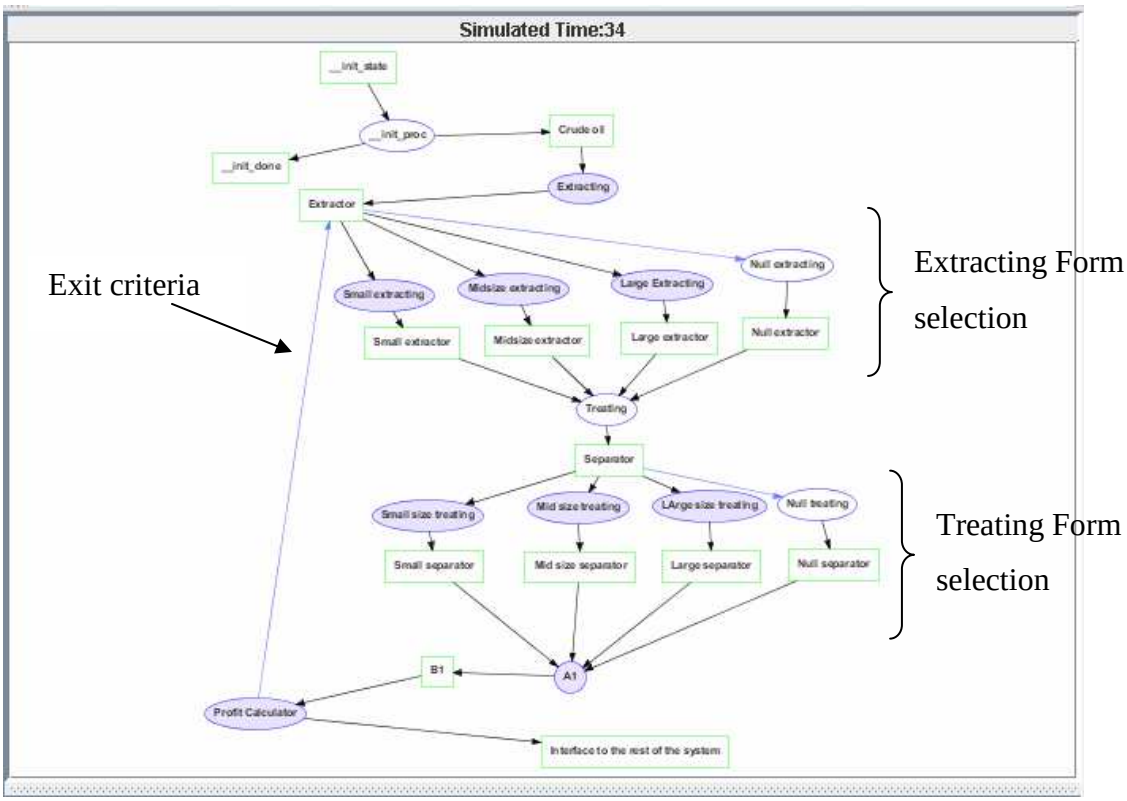


FIGURE 5: OPN model with loops to optimize subsystem selection

The implementation of the NPV function was done using the global script. The function is being called every time the program enters the Profit Calculator. The script used for that is:

```
def NPVCalc(TotalCost,TotalBuildingTime,TotalCapacity,TotalProductionTime):
  #declare constants
    i=1;
    interest = 1;
  #calculate NPV
    YearlyCost = TotalCost/TotalBuildingTime;
    while (i <= TotalLeasingTime):
      i=i+1;
      interest = interest*(1+r);
      if (i <= TotalBuildingTime+1):
        NPV = NPV - YearlyCost/(interest);
```

else:

*NPV = NPV + (TotalCapacity*PricePerBarrel)/(interest);*

else:

return [NPV];

Result analysis

Running the model reveals there are 149 possible different Extracting and Treating subsystems combinations. A complete list of the different combination can be found in appendix D. It is important to note that not all solutions actually extracted all the oil capacity. Some did not reach that capacity within the leasing time. Some of those solutions might still be valuable, since they reach production very fast and thus have a smaller effect of the interest rate on the overall profit.

The following figure presents the total discounted profit of the 149 combinations found by OPN. The combinations are arranged from the highest expected profit to the lowest.

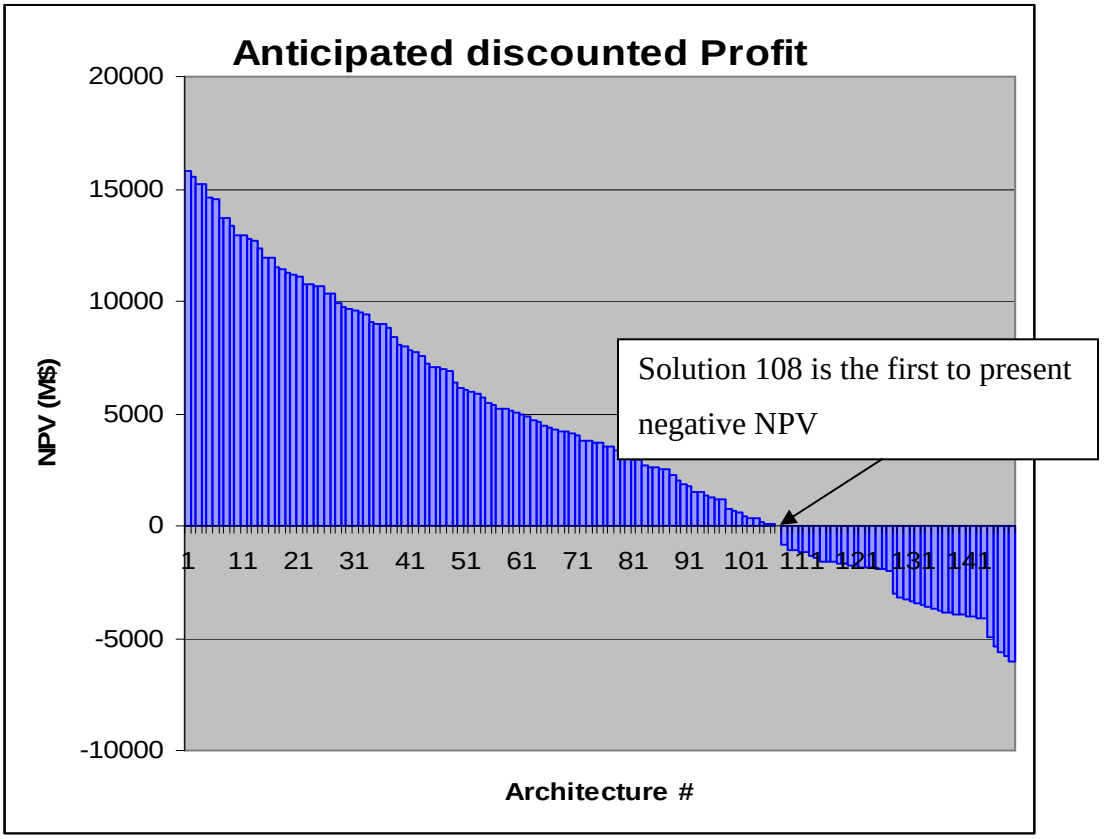


FIGURE 6: Anticipated profit for all possible architectures

As can be seen there are over $20B difference in the expected discounted profit between the most profitable to the lowest profitable combinations! The table and figure below shows the 20 most profitable combinations. It is hard to find a "golden role" that will connect the form selection to profit. Some solutions utilize extensive capacity built in long period of time (like solution number 15) whereas others build small capacity fast (like solution number 19).

TABLE 8: Anticipated profit for top 20 architectures

Solution #	# of Large Extracting forms	# of Medium Extracting Forms	# of Small Extracting Forms	# of Large Treating Forms	# of Medium Treating Forms	# of Small Treating Forms	Profit (M$)
1	0	1	1	0	0	2	15837.63
2	1	0	1	0	1	1	15530.68
3	0	2	0	0	1	1	15244.01
4	0	2	0	0	0	2	15240.50
5	1	0	0	0	1	0	14627.50
6	1	1	0	0	2	0	14509.30
7	1	1	0	0	1	1	13719.11
8	0	0	2	0	0	2	13656.13
9	0	1	1	0	1	1	13376.60
10	0	1	0	0	1	0	12966.03
11	1	0	1	0	2	0	12928.93
12	0	1	0	0	0	1	12768.24
13	0	2	0	0	2	0	12647.98
14	1	0	0	1	0	0	12380.60
15	1	1	0	1	0	1	11950.68
16	1	0	2	0	0	3	11942.75
17	0	0	2	0	1	1	11509.18
18	0	1	2	0	0	3	11417.41
19	0	0	1	0	0	1	11283.64
20	2	0	0	0	2	0	11147.36

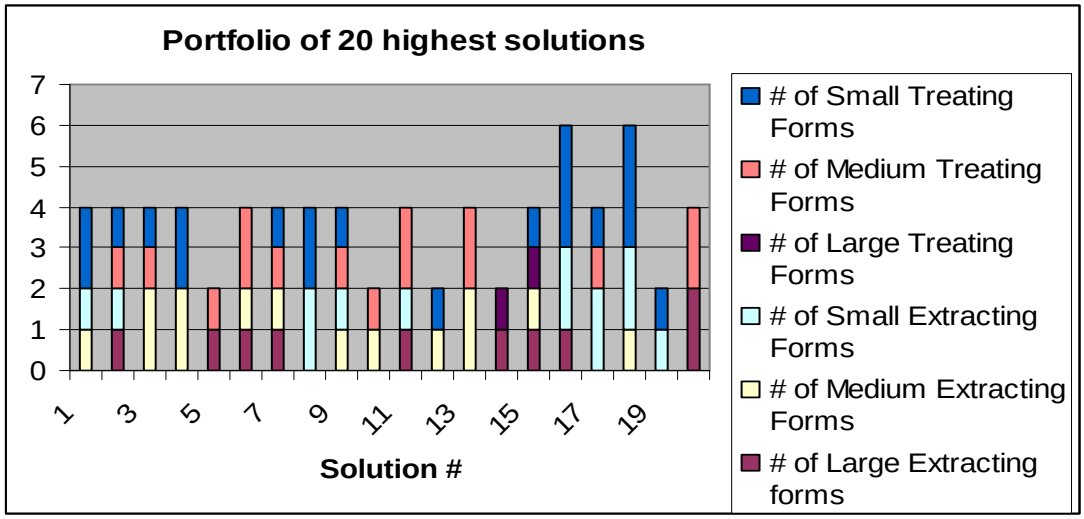


FIGURE 7: Portfolio breakdown for 20 top architectures

Two of the three solutions that were calculated earlier actually appear in the top twenty solutions. Building one large extracting form and one large treating form appears as solution #14. Building two small size extracting forms and two small size treating forms appears as solution #8. Thus, by expanding the range of possible solutions, the OPN was able to generate higher NPV by finding a more profitable combination.

3.5 Conclusion

The suggested algorithm can significantly increase the ability of OPN to simulate real life decisions faced by the architect. It does so by allowing the model to flexibly adjust the number of forms associates with each function. By allowing that flexibility, the model can decide on the best combination of forms that maximizes stakeholder's value without being constrained by the architect. Thus, by utilizing that functionality correctly a bigger range of solutions can be explored, answering questions like, should I build that system from small number of high capacity forms, large number of low/mid capacity forms or a combination?

4 Suggested framework that facilitate generation of "out-of-the-box" solutions to technical problems

4.1 Introduction

Generating "out of the box" solutions is a task usually associated with creativity. Several attempts have been made to create an "ordered creativity". Some of these attempts have focused on creating better tools that encourage creativity while others have focused on capturing creativity. One of the interesting methodologists in that respect is TRIZ [10], [11]. Its underlying assumption is that all technological systems evolve along certain universal directions that are governed by laws of evolution. Thus, if a current system design is given, the future design can be repeatedly predicted. The law of increasing degree of ideality, for example, states that [10] "evolution of technological systems proceeds in the direction of increased degree of ideality" Using that method, "out-of-the-box" solutions can be generated by following those laws of evolution.

Our research generated "out-of-the-box" solutions taking a "top-to-bottom" approach. This approach allows (in contrast to TRIZ) exploration of large range of solutions, not only the next step in evolution. We first answered the question: what are all the possible ways to address that issue in a domain and discipline neutral space? The second step was to apply that answer to a real problem.

We believe that there are several advantages to this approach. First, it creates a framework that spans the entire gamut of possible solutions and is not limited to an existing one. It is also a good basis for a brainstorming process aimed at finding innovative solutions to the problem. Since it is discipline neutral, once the framework is created it can address any similar problem, thus increasing the efficiency of the future problem-solving process.

To demonstrate the proposed approach, I will present a real-life problem we tried to solve. As part of our research we were asked to give fresh and out-of-the-box suggestions how to solve problems inherent to that specific project. One of the biggest problems faced was how to deal with ice accumulation that causes a serious threat to the system productivity for about six months every year. We took this problem as a test case since it was among the most urgent problems in that project that had a possible huge effect on the profitability of the project (a shutdown of six months every year will significantly affect the overall project NPV). Additionally, our sponsoring company had a small amount of accumulated knowledge in that area, especially in deep water, which increased our room for maneuvers in searching for new solutions.

The first step was to approach the question: What are all the possible ways to protect something from something else.
We were able to summarize all those ways into four methods (that we translated into OPD diagrams), which in our view represent all the possible ways that a system can protect itself from another system/force:

Those methods are:

1. Resilience.
2. Avoidance.
3. Isolation.
4. Redundancy.

These options served as a basis for a brainstorming session to explore the entire gamut of possibilities of protection against ice. Additionally, these options can be incorporated into the OPN process and thus use the OPN's inherent ability to generate all possible permutations.

In this section, I will present those four methods as well as their OPD representation, discuss and demonstrate their applicability to other areas of engineering (like electronics) and dive into the specific ice protection issue.

4.2 Finding all the ways to protect something from something else

When taking the specific domain and discipline out of the problem space, it seems there are only four basic theoretical ways to protect something from something else. Those four ways can be mapped into six practical methods of protection. This section will present these basic methods as well as their practical representation.

4.2.1 Basic representation of interference

Interferences are a result of cross relations between an instrument and its surrounding. For example, corrosion is the result of interaction between the certain kinds of form material and the surrounding oxygen.

An OPD representation of interferences can be viewed in FIGURE 8. It is important to note that in a specific system, the interference is not a direct result of the different functions of the system but rather of the forms selected to perform those functions. Continuing with the corrosion example, if the corrosion occurred at the support of a bridge, it is not a result of the supporting system but rather an interface with the specific metal form that was selected to perform the supporting function. In that specific example, changing that form, for example to a stainless steel based structure, might solve that problem.

Another important point is that the interference is caused by an environmental form of some kind. Corrosion needs oxygen, and electromagnetic interferences need electromagnetic wave in order to exist. Additionally, the interference itself is a function and sometimes a special form. The interference can be caused by an external form (collision) or from within the instrument (explosion).

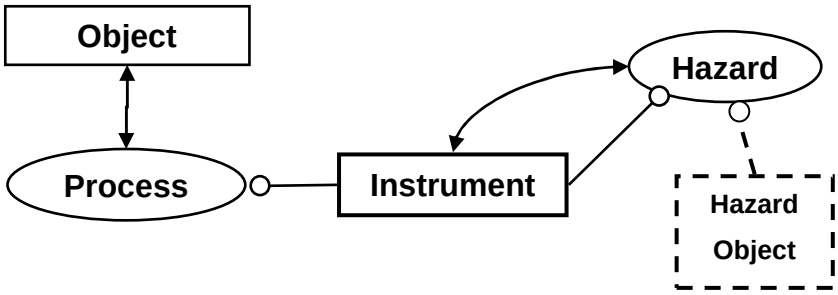


FIGURE 8: OPD representation of interferences

Following that understanding, we tried to identify groups of methods that mitigate or even eliminate interferences. The rationale is that finding those representations and defining them in a system-neutral way (through OPD) can be a good basis for a brainstorming process to generate out-of-the-box solutions to deal with specific interferences issues.

4.2.2 The four basic methods to protect something from something else

In order to focus the research we altered the question to: “what are the basic ways to protect something from something else?” That question yielded four different system-neutral possible solutions:

1. **Resilience**

This system-neutral method is targeted at increasing the resistant ability of the instrument.

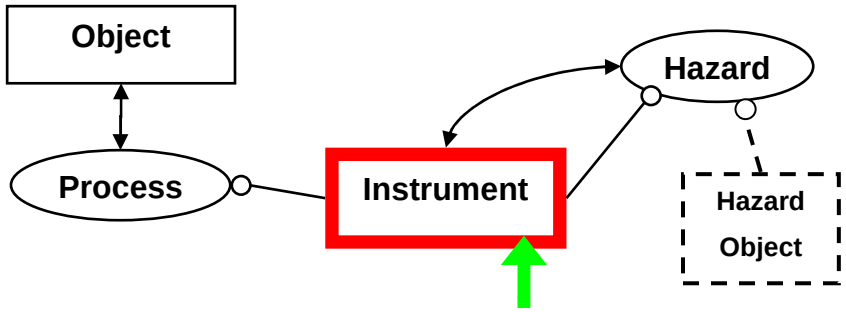


FIGURE 9: Resilience protection against interferences

Looking at examples from a solution-specific domain reveals some different options for resilience. One is to increase the protection of the instrument against the interference in such a way that although the interference strength remains the same, the instrument is better equipped to withstand it, for example, altering the upper layer of a metal plate to protect it against corrosion or making a stronger structure to protect against side winds. Flexibility is another way to increase a form’s ability to resist interference. In nature we can see some evidence of that kind of solution. Many plants for example, are very flexible as a means to protect against wind.

Another option for resilience is to change the form so the potential effect of the interference will be reduced, for example, changing the cross section of a structure to reduce the effect of side winds.

2. Avoidance

Since interference is associated with some kind of form, another way to protect against interferences is simply to eliminate the hazard.

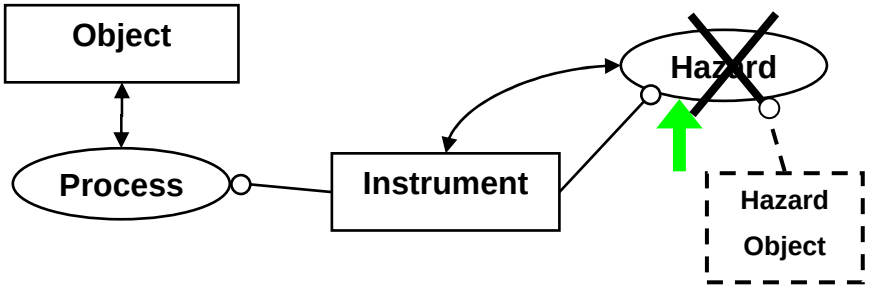


FIGURE 10: Avoidance protection against interferences

Avoidance can be done in several ways. One is to remove the hazard. For example, one of the problems in a deep ocean oil rig in an arctic environment is big ice blocks that collide (and sometimes cover) the oil rig system and thus potentially reduce its efficiency and operability window. Several methods can be applied that move the ice from the oil rig environment. Another option for avoidance is to move the instrument to a different environment with reduced level of interferences (or no interferences at all), for example, moving an iron system to an oxygen-free environment to protect it against corrosion.

3. Isolation

Another protection option would be to attack and possibly eliminate the interface between the hazard and the instrument. While there is some resemblance between that method and resilience, the difference is very clear. In resilience the alleviation is achieved by altering the instrument, whereas in isolation the interface is altered. A good example would be the distinction between the two following methods of corrosion protection:

1. Altering the upper layer of the metal that interacts with the surrounding oxygen (resilience).
2. Applying coating to the metal that eliminate the interaction between the metal and oxygen (isolation).

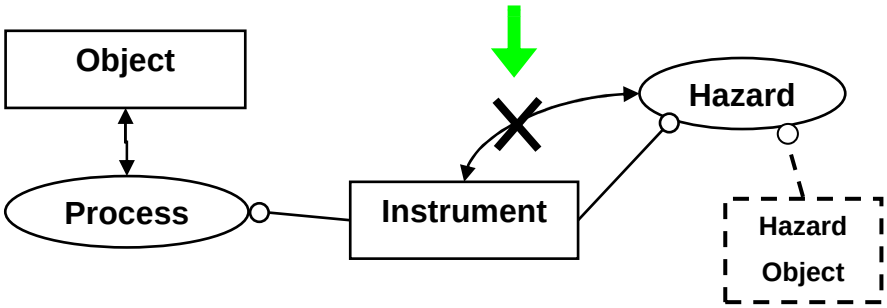


FIGURE 11: Isolation protection against interferences

This kind of protection can be achieved in several ways. One can apply a boundary layer to protect against the interference, for example, a Faraday cage to exclude electrostatic influences. A different approach is to alter the environment, for example, surrounding the system with a water environment as a damping method against vibrations.

4. **Redundancy**

Having additional instruments to perform the same function is another option for protection. In that regard there are actually several sub-options:

1. Having the same type of instrument as a backup in case the first instrument fails.
2. Having a different type of instrument as a backup. This kind of solution might be applicable in dealing with interferences that do not have accumulative nature but rather a threshold for causing malfunction. A redundant system in that case might be less adequate to perform the function but will be able to withstand the high level of interference, for example, a subsystem form that will stop working at certain radiation level. Having two of the same kind of that form will not increase the overall radiation resistance, since both forms will fail at the same time. A solution might be adding a different kind of form as backup that, although not as efficient as the first will be able to resist high levels of radiation. Another case where having a different backup system might be the preferred solution is when the primary instrument is too expensive to duplicate. For example, in some cars the spare tire is of poor quality in order to reduce the overall cost while allowing redundancy in cases of flat tires.
3. Having another instrument that works together with the first one in such a way that the effect of the interference on each of the instruments is reduced. Those kind of protections are applicable to interferences that can be divided using additional instruments. One example is of interferences that interact with the instrument through force or pressure (like colliding obstacles or friction). In that case, having an additional instrument (for example, an engine) will split the effect of the interface.

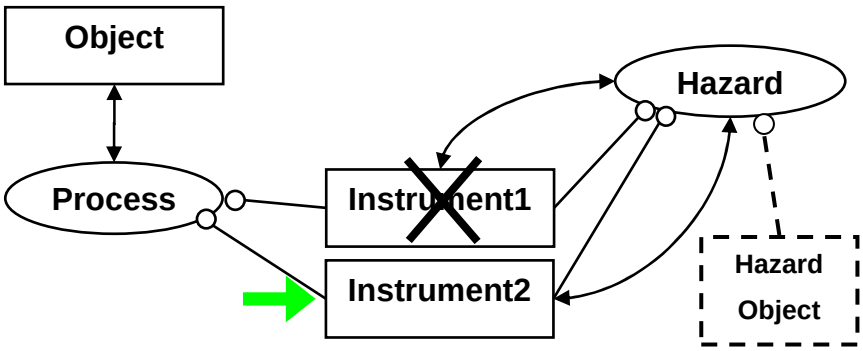


FIGURE 12: Redundancy protection against interferences

Conclusion

This section presented four general ways to protect something from something else. Those methods have an advantage of being system-neutral. That means they are general enough to be used as a base for a brainstorming session regardless of the actual system the solution is meant for.

The following section will focus on a specific example of utilizing those methods generate a solution for a real-life engineering issue.

4.3 Implementation example - Ice protection

This section will demonstrate how those four ways of protection are used in a real life problem –protecting an oil exploration system from ice. The specific oil exploration system consists of a group of supporting structures that interact with ice. Each of these structures supports a different subsystem (extracting, treating, etc.) that in turn is supposed to service a Deep Ocean oil field. The ice itself usually forms in large accumulations. This has two possible negative effects on those structures. The first is the impact of the floating accumulations that can cause the structures to fail. The second is that ice can accumulate on top of those structure (especially those that are close to the water surface) and thus harm the operability of the subsystem that is being supported by those structures.

The process described in this section was developed to serve two goals. The first is to move from system-neutral solutions (as was mentioned in the previous section) to system-specific possible solutions, in order to give the architect a set of possible solutions to his specific system. The second goal is to allow the architect to use those possible solutions to tailor a specific solution for each of the oil exploration subsystems.

Generally, the process can be split into the following stages:

1. Translating the system-neutral solutions to a family of system-specific solutions.
2. Using each family to generate a specific possible solution for a specific subsystem.

This section will elaborate on that process, discuss some of the practical implementation issues and demonstrate a specific implementation in OPN.

Translating the system-neutral solutions into a family of system specific solutions

The first stage in moving to system specific solutions is to understand the system in question. The OPD representation of each of the relevant structures looks as follows:

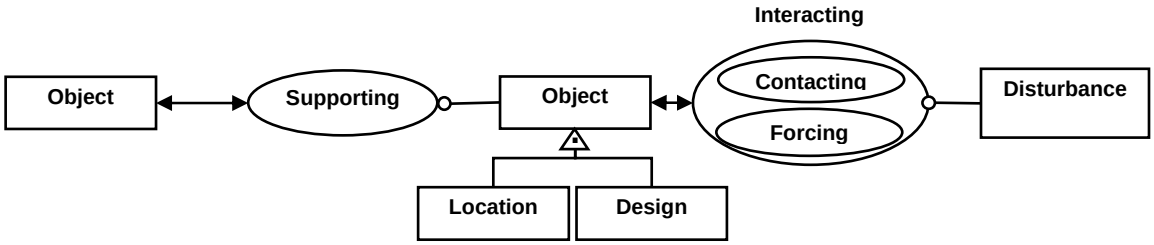


FIGURE 13: Basic representation of a supporting structure and its interface with ice

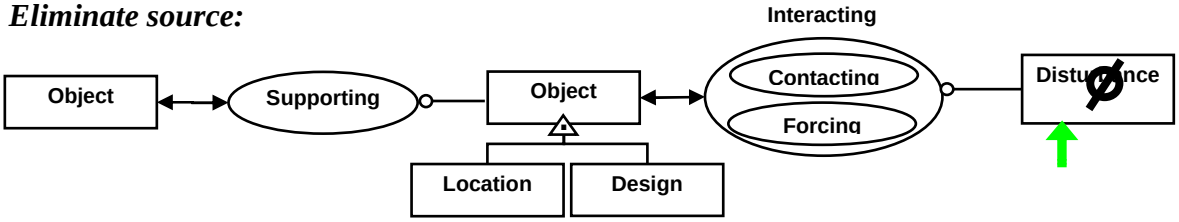
Additionally, we adjusted the list of the four basic methods to the specific problem. First, we realized that resilience can be extended to three generally possible solutions to protect against ice: absorbance, minimizing force and strengthening. Second, we decided to take into account two possible types of avoidance since it can be done at the interference level or at the instrument (structure) level. Additionally, we decided not to incorporate redundancy in the range of possible solutions. The cost of each of the subsystem is relatively very high and thus, we do not believe that redundancy is a feasible solution.

Using these assumptions, the four system neutral ways of protection were enlarged into six different patterns to protect against ice:

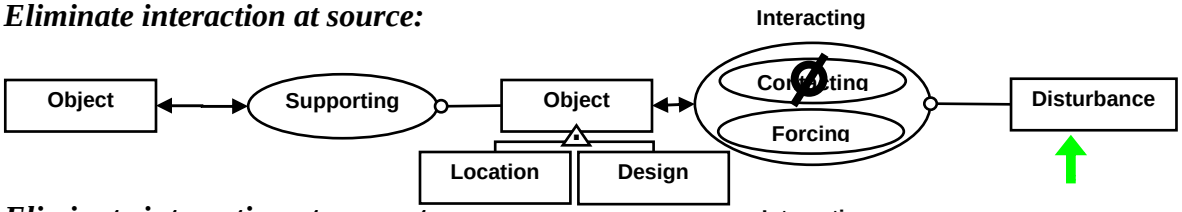
1. Eliminate source.
2. Eliminate interaction at source.
3. Eliminate interaction at support.
4. Use intermediate object to minimize force; allow contact to support.
5. Use intermediate object to absorb force; no contact to support.
6. Withstand interaction.

The following figure shows the OPD representation of the six possible patterns:

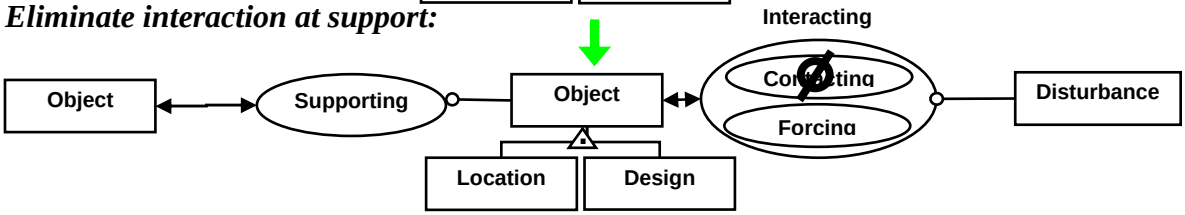
Eliminate source:



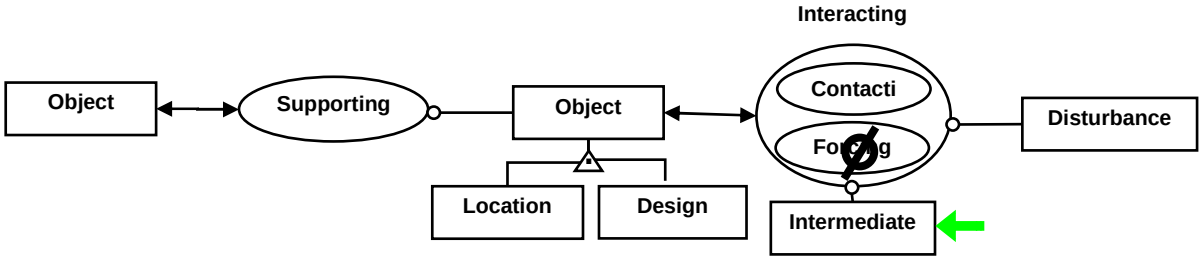
Eliminate interaction at source:



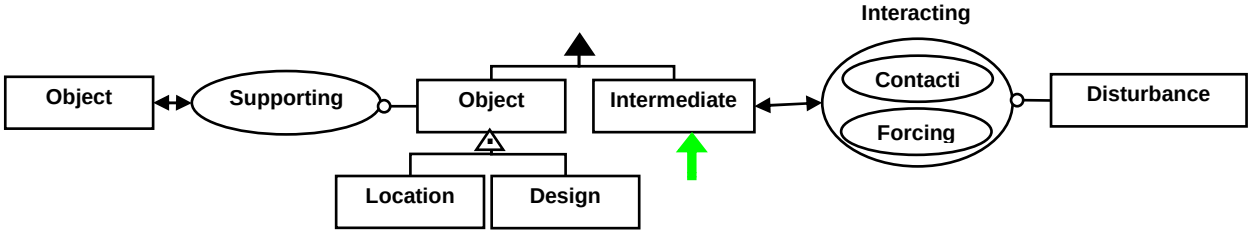
Eliminate interaction at support:



Use intermediate object to allow contact, but minimal force:



Use intermediate object to absorb force but not on support:



Withstand interaction

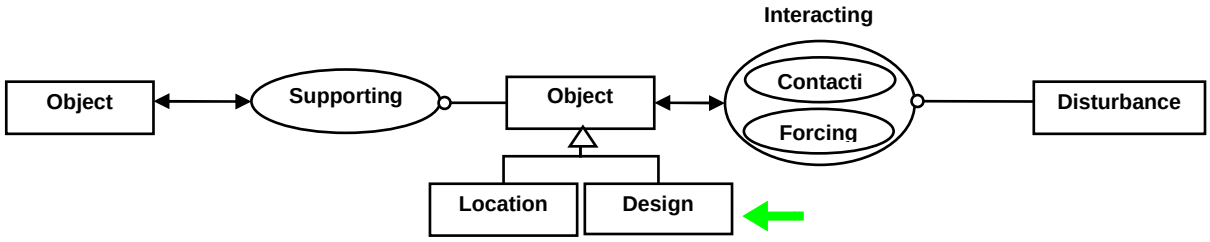


FIGURE 14: Patterns to protect against ice

Utilizing patterns to protect against ice to generate specific solutions

Following a brainstorming process, we were able to extract those patterns into specific possible solutions. Those possibilities will be further investigated in order to choose the best one for each of the subsystem structures. The following table summarizes the solutions we were able to extract from each pattern:

TABLE 9: Specific solutions to protect against ice

Pattern name	Specific solutions				
Eliminate source.	Heat water	Chemically treat water/ice	Agitate water		
Eliminate interaction at source.	Move Ice away	Break Ice Naturally	Break Ice artificially		
Eliminate interaction at support.	Move support to land	Move support to Ice free Water	Move support underwater	Skim support over ice	Use Ice as support structure
Use intermediate object to minimize force; allow contact to support.	Fairing	Shock observer	Semi submerged		
Use intermediate object to absorb force; no contact to support.	Shielding	Bumpers	Build artificial Island/Berm		
Withstand interaction.	Strengthen structure	Minimize cross-section			

The OPD representation of the entire suggested range looks like the following:

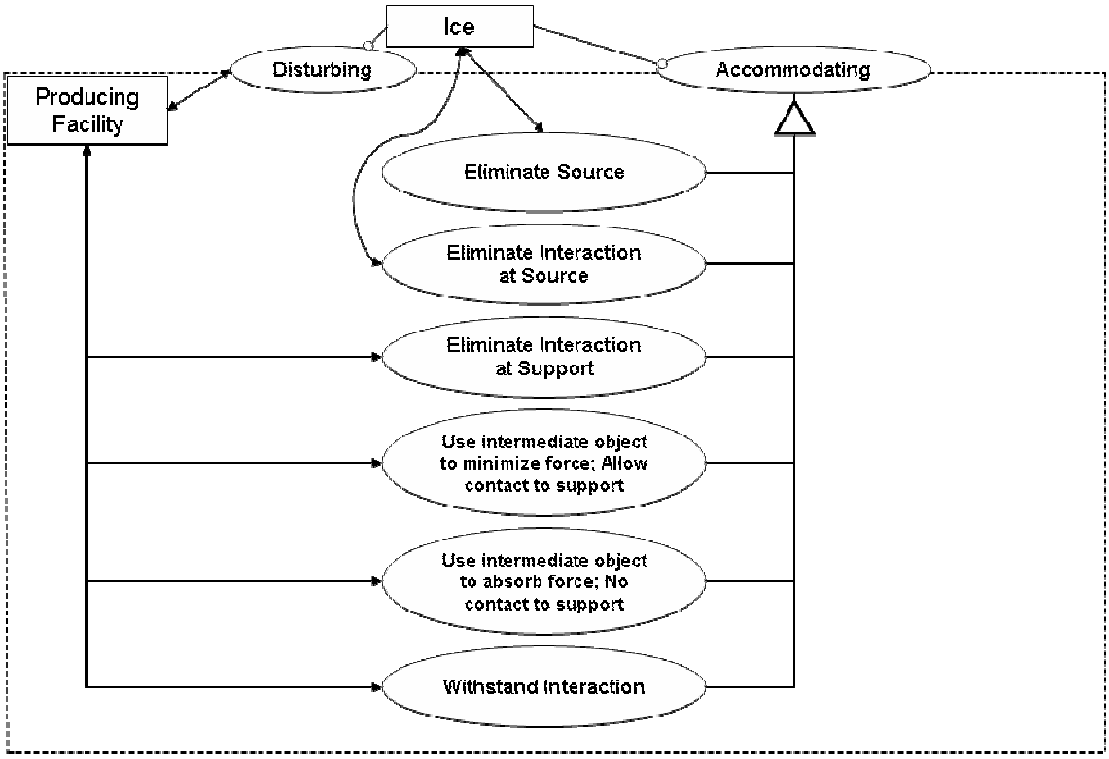


FIGURE 15: Ice protection specific solutions – level 1 decomposition

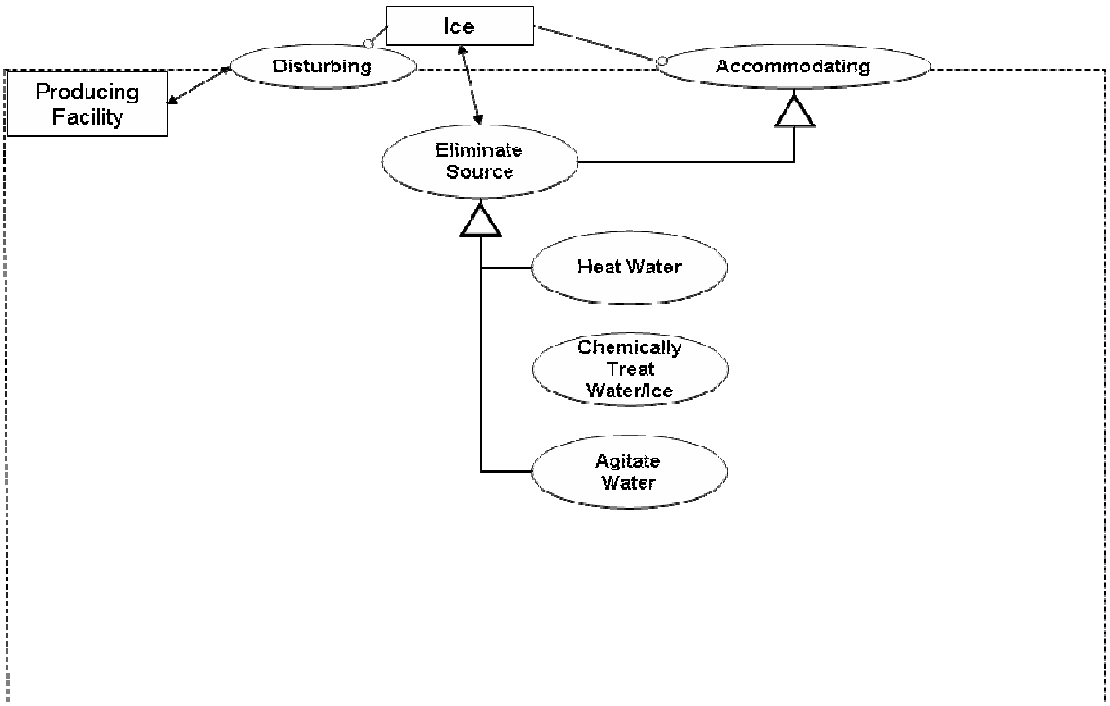


FIGURE 16: Ice protection specific solutions – level 2 decomposition – Eliminate source

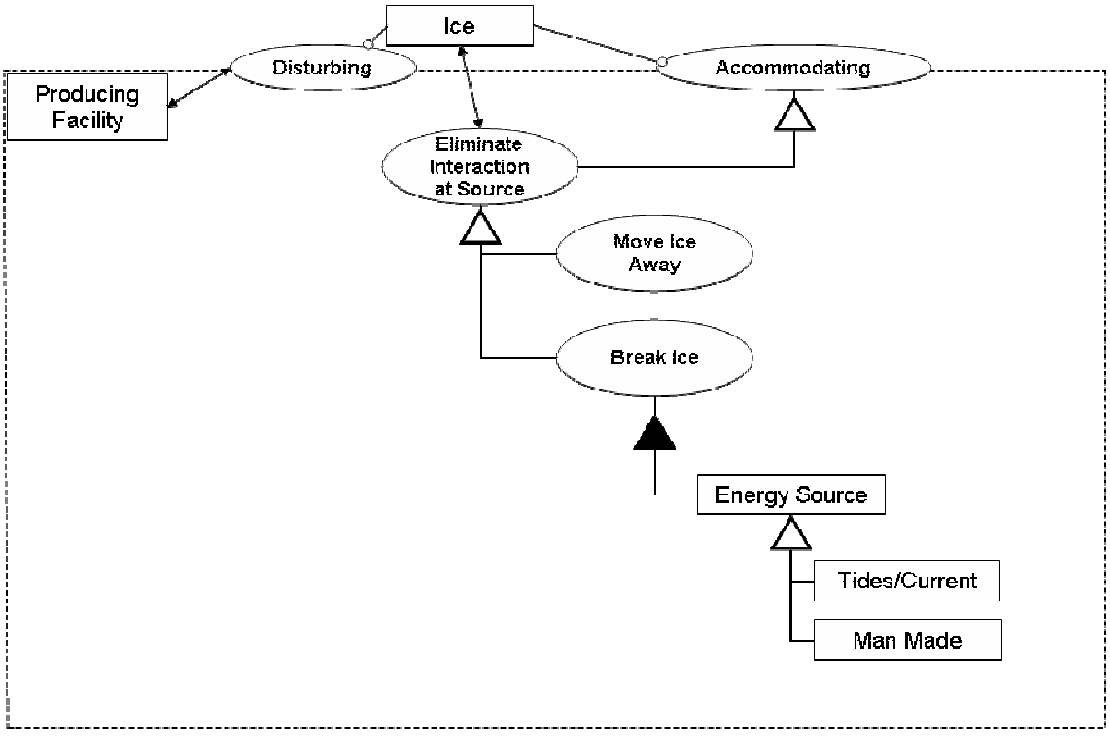


FIGURE 17: Ice protection specific solutions – level 2 decomposition – Eliminate interaction at source

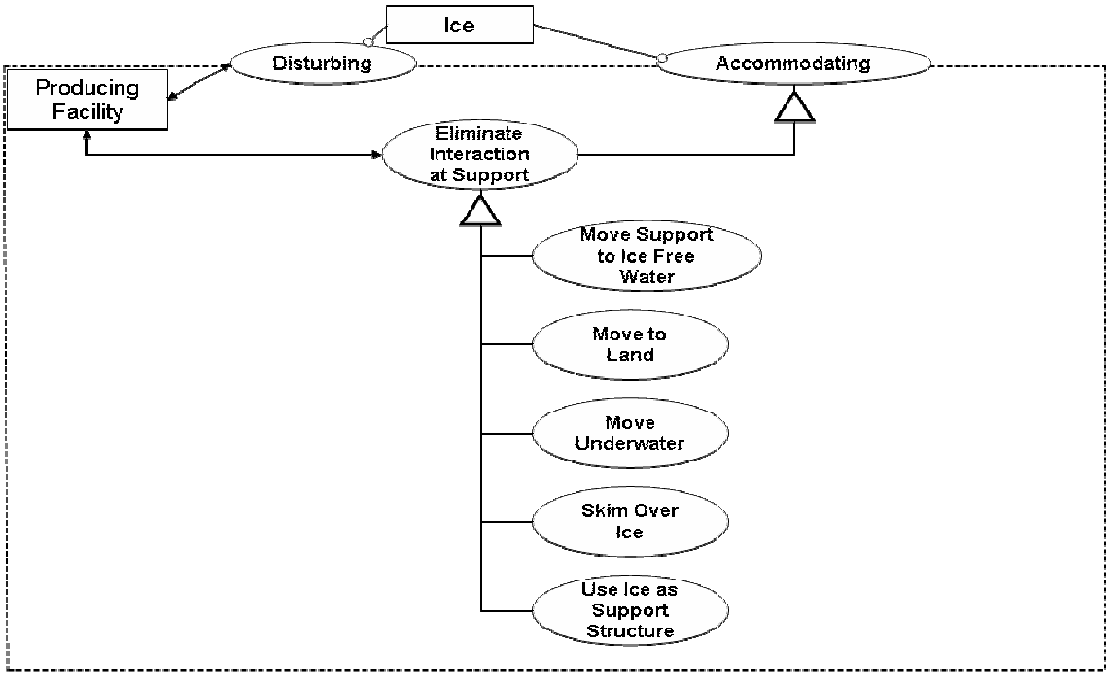


FIGURE 18: Ice protection specific solutions – level 2 decomposition – Eliminate interaction at support

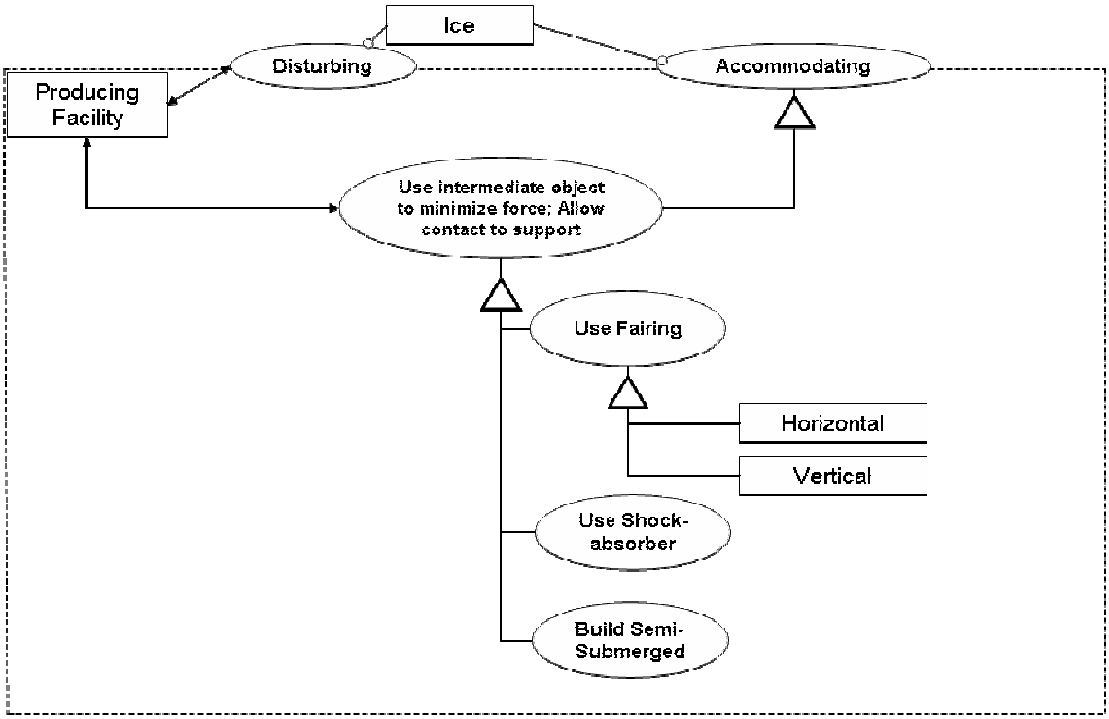


FIGURE 19: Ice protection specific solutions – level 2 decomposition – Use intermediate object to minimize force: allow contact to support

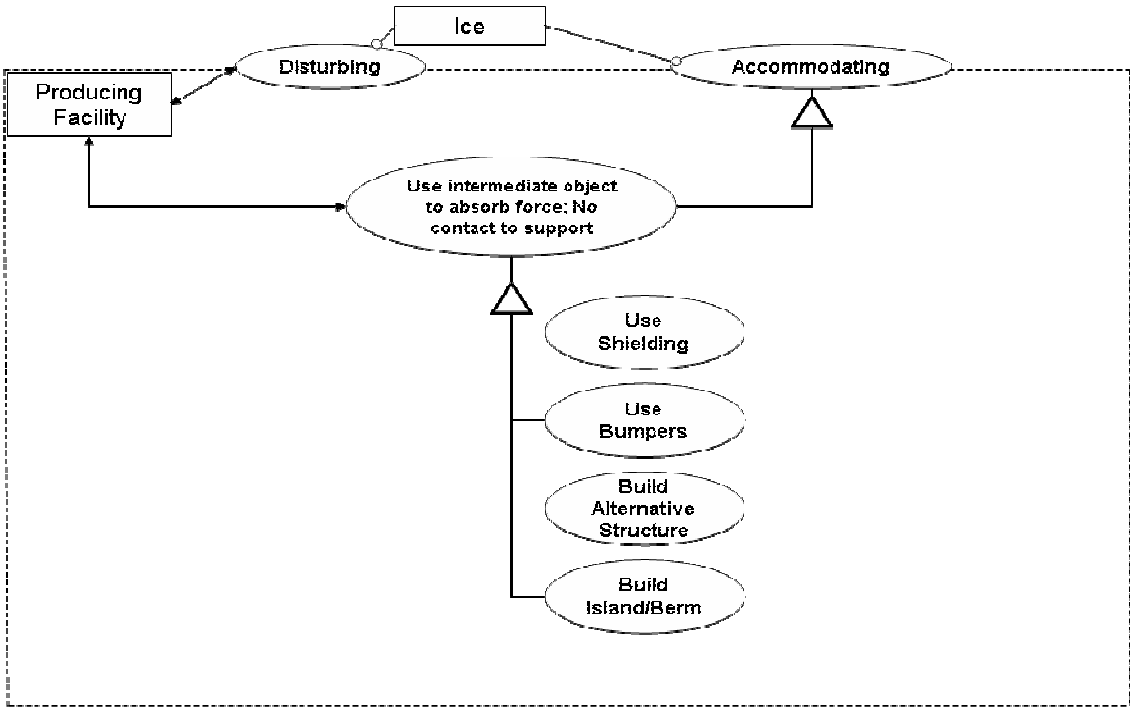


FIGURE 20: Ice protection specific solutions – level 2 decomposition – Use intermediate object to absorb force: no contact at support

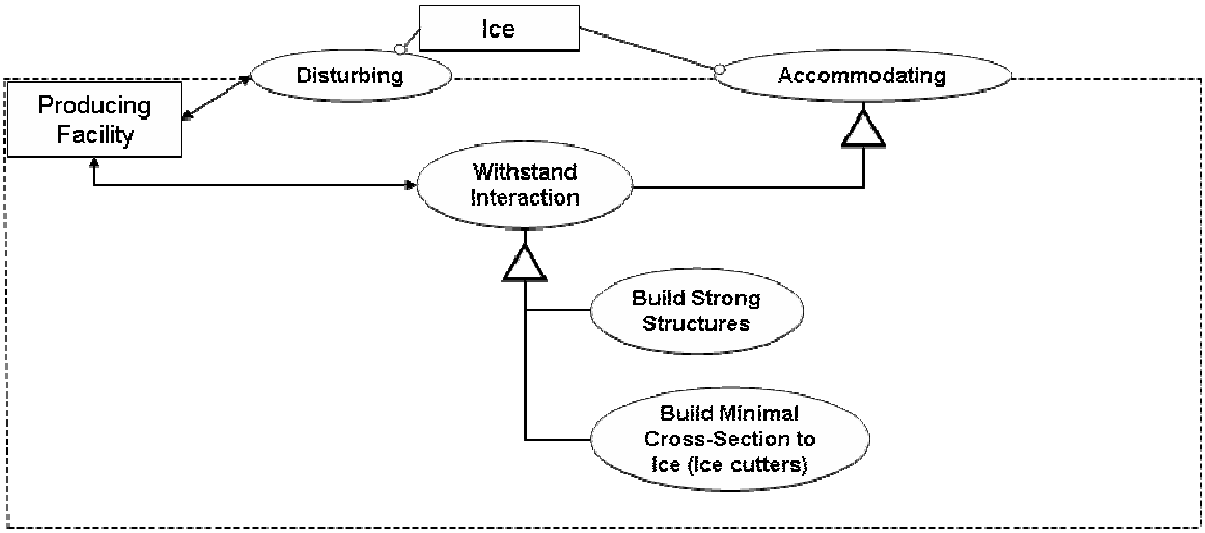


FIGURE 21: Ice protection specific solutions – level 2 decomposition – Withstand interaction

The next step was to use those specific possible solutions to tailor a specific solution for each of the structure subsystems mentioned earlier. We chose to do that using OPN due to its ability to span the entire gamut of solutions for each of those cases and find the one that offers the best value.

Incorporating ice protection methods in OPN

This phase is aimed at find all the possible permutations of ice protection solutions based on the specific possible solutions found earlier. The rationale is to use the specific solution list to generate all the different possible combinations that might give ice protection to the specific system. The architect will be able to use those permutations to tailor a specific solution for each of the structural subsystems.

In order to perform that phase we made some assumptions. The first is that specific possible solutions can be merged. The underlying assumption is that often those solutions do not guarantee 100% protection against ice. In those cases a combination of solutions can increase the overall protection level of the subsystem structure. For example, breaking the ice and heating it might work well together. The ice could be heated and then broken or vice versa. Calculating the overall value of that combination of solutions can be difficult, since in many cases it is not a simple sum of the individual protection levels. For example, heating the ice before breaking it might cause the breaking activity to be more or less efficient than just breaking it as a stand-alone activity.

The second assumption is that not all the possible solutions can be merged due to physical limitations; for example, the structure cannot be moved to land and at the same time move underwater. Furthermore, we assumed that intermediate objects can be used either to absorb force or to minimize it; thus “Use intermediate object to absorb force” and “Use intermediate object to minimize force” cannot coexist. Our last assumption was that there is no value in combining solutions that deteriorate (or even cancel) each other’s effect. For example, moving the structure underwater cancels the effect of breaking the ice.

The following figure illustrates the way these assumptions can be incorporated into the model. The selection of the protection method became a set of selections, each dealing with a different kind of protection. At each selection the options are to select the specific protection or not to select it. If the protection is selected it is added to a portfolio of solutions. Additionally, solutions that cannot work together are implemented in parallel (as in the case of "Use intermediate object to absorb force" and "Use intermediate object to minimize force").

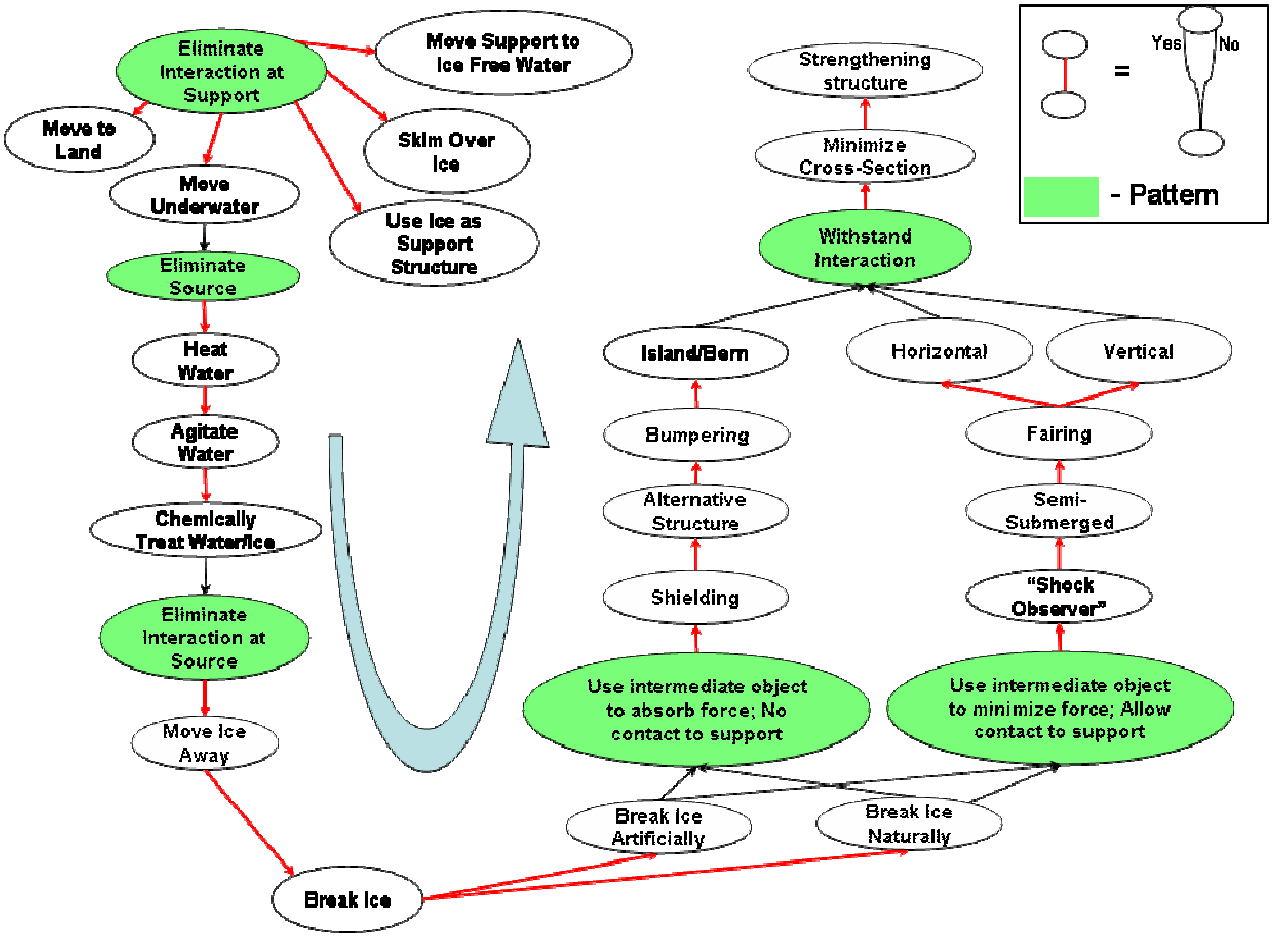


FIGURE 22: Ice protection - multi solution selection

The following table summarizes all the possibilities of those combinations. A '+' sign at the intersection of two solutions means that the two solutions could be merged.

TABLE 10: Possible combinations of ice protection solutions

	Heat water	Chemically treat water/ice	Agitate water	Move Ice away	Break Ice Naturally	Break Ice artificially	Move support to land	Move support to Ice free Wat	Move support underwater	Skim support over ice	Use Ice as support structure	Horizontal Fairing	Vertical Fairing	Shock observer	Semi submerged	Shielding	Bumpers	Alternative structure	Build artificial Island/Berm	Strengthen structure	Minimize cross-section
Heat water																					
Chemically treat water/ice	+																				
Agitate water	+	+																			
Move Ice away	+	+	+																		
Break Ice Naturally	+	+	+	+																	
Break Ice artificially	+	+	+	+	+																
Move support to land																					
Move support to Ice free Water																					
Move support underwater																					
Skim support over ice																					
Use Ice as support structure																					
Horizontal Fairing	+	+	+	+	+	+				+	+										
Vertical Fairing	+	+	+	+	+	+				+	+	+									
Shock observer	+	+	+	+	+	+				+	+	+	+								
Semi submerged				+	+	+						+	+	+							
Shielding	+	+	+	+	+	+				+	+	+	+	+	+						
Bumpers	+	+	+	+	+	+				+	+	+	+	+	+	+					
Alternative structure	+	+	+	+	+	+				+	+	+	+	+	+	+	+				
Build artificial Island/Berm					+	+						+	+	+		+	+	+			
Strengthen structure	+	+	+	+	+	+				+		+	+	+	+	+	+	+	+		
Minimize cross-section	+	+	+	+	+	+				+		+	+	+	+	+	+	+	+	+	

These combinations can be further extracted to include more than two possible solutions. The rationale is to cover all the possible cases where three or more solutions merged together will offer higher value than only one or two solutions merged. For example, breaking the ice might be combined with shielding and strengthening the structure to offer the highest protection and highest overall value. The general rule is that if solution A can be combined with Solution B and Solution B can be combined with Solution C, then a trio of solutions A, B and C is also possible.

Following that, calculating the number of possible permutations reveals that there are over 10,000 possibilities, which is higher than the possible number we can analyze. In order to reduce that number without losing the integrity of the model, we made additional assumptions. We defined the maximum number of different solutions that can be

combined into one solution as 3 and we eliminated the solutions that had the lowest probability to be implemented (based on estimation of the technical feasibility). Those assumptions reduced the number of permutations for that model to 665, which is within the system analysis capabilities.

The modified structure looks as follows (solution marked with Red indicates low probability solutions that where left out of the model):

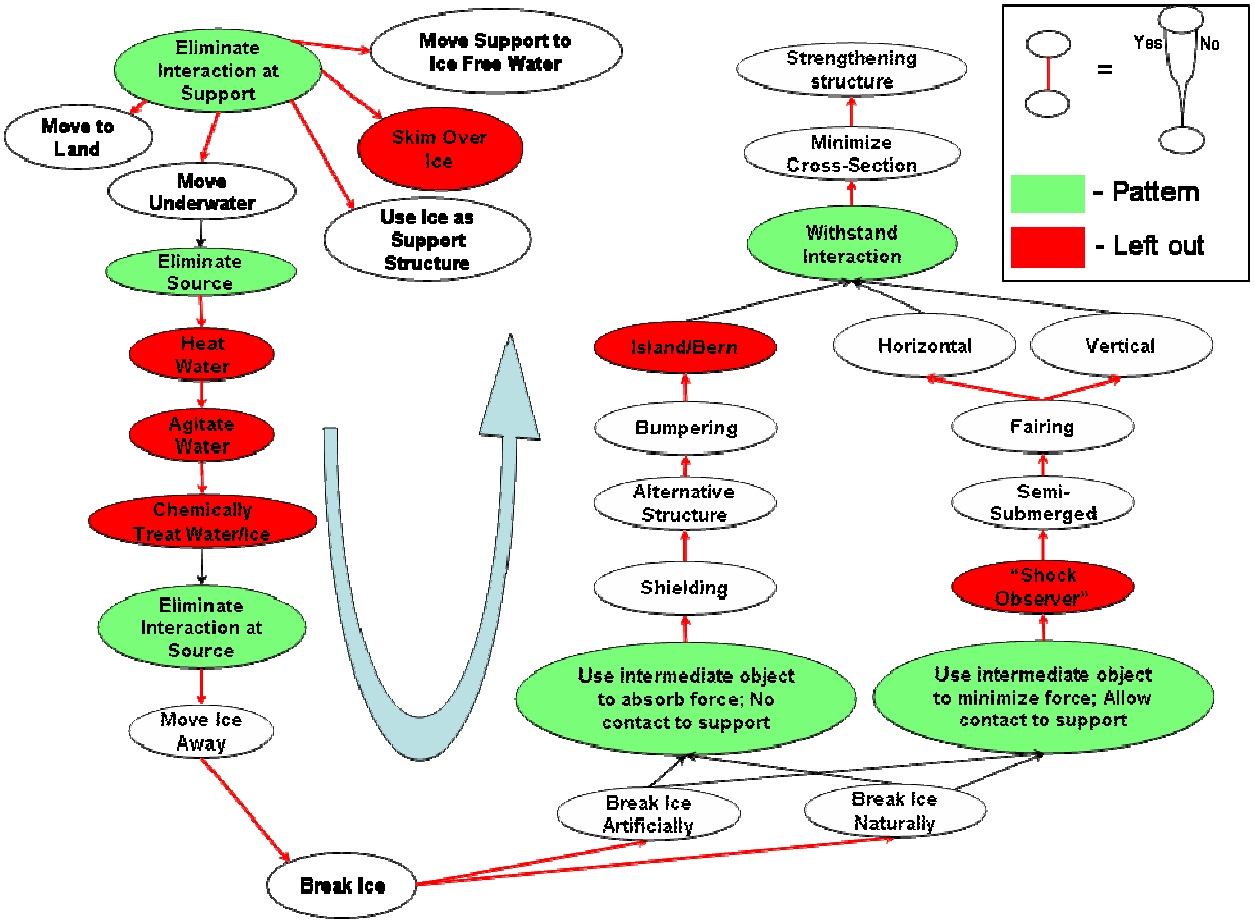


FIGURE 23: Incorporating multi-solution selection into a reduced model[12]

[12] The OPN model incorporating that model can be seen at appendix B.

4.4 Conclusion

This chapter presented a structured way to generate “out-of-the-box” solutions. It started from the highest generalization and dived into the specific problem. That approach offers the advantage of exploring the entire gamut of possible solutions to a problem while not being bounded by existing practices. The outcome of that process is a list of permutations – in our case, all the possible ways to protect the oil exploration system from ice.

Having all those permutations at hand allows the architect to examine each in order to select the best solution for each of the subsystems. It can be done by defining the value formula for each of the structural subsystems and then running the model to find the best solution for each of the structural subsystems based on that formula.

5 How to deal with a non-fixed boundary

5.1 Introduction

Finding the best architecture to offer value to stakeholders can be affected by the definition of the system boundary. Often a lean system can offer greater value for the invested resources than a comprehensive one. One example is an oil exploration system where there are several ways to bound the system:

1. Lean system - bounded very early right after the separating phase of the oil mix into its products. In that case an "outside the system" entity will take care of moving the products from the separating facility and passing it to storing/distribution centers as a mid-point to distributing it to customers.
2. Comprehensive system - placing the boundary after moving the products to the final customer. That means the transportation and storing subsystems as well as distribution etc. become one of the processes the system needs to support.
3. Mid size system– anywhere between a lean system and a comprehensive system. One possibility is to define the system boundary after transferring the separated oil products to a storing facility (thus incorporating storing within the system). Another possibility might be to define the system boundary after transferring the product to the distribution center etc.

The decision where to put the system boundaries depends on several considerations. One is the strength of the relationship between the different subsystems. An architect should consider the number and complexity of interfaces between the system and the outside world. Changing the system boundary can increase or reduce both the total number of interfaces and their complexity. Another consideration is the value each subsystem creates for the stakeholders vs. the effort associated with incorporating it. For example, effort can be measured by cost and complexity, whereas value can be measured by political power. I selected political power to demonstrate that value is not determined

only by the technical relevance of the subsystem to the entire system (like steering subsystem to a transportation system). It could also create non-technical value.

Having these two considerations in mind, the architect can decide which subsystems (and as a result of that, functionality) to incorporate within the system boundary. Often it becomes an iterative process: The architect needs a system and stakeholder model in order to check whether each subsystem should be within the system boundary, thus, an estimated model needs to be built in advance. The architect can then check if it has the best boundaries and, if needed, correct them. The process of finding the right boundaries can end up being a process of iteration between estimation and evaluation till finding the right system boundaries.

That raises a fundamental question, relevant to both OPN and OPD, which is how to represent a changing system boundary, because in those methods a process can either be in or out of the model. It cannot dynamically move to the other side of the system boundary. The current practice is to put all the possible processes inside the OPN and OPD. Those who might move outside of the system boundary get an additional form beside those physically feasible, which is called Null. This form gets a value of zero for every relevant attribute or parameter. Picking this form simulates a situation of leaving the process outside of the system boundary. While this solution actually allows generation of architectures, it raises other problems. The system cannot be treated as a "black box" – since the system boundaries are not fixed, there is more than one function that can be the last. That creates a situation where an interfacing system needs to dive into the system (probably one level down) in order to understand what process (or sub-system) it needs to connect with. Furthermore, when examining both the OPN and OPD, it is not clear which process can be the last one before the system boundary. This section will propose a way to deal with this issue in OPD and OPN and will discuss its advantages.

5.2 Proposed Solution for a non-fixed boundary – change of ownership

So far a “thing,” whether it is a form or a function, could be either within or outside the system boundary. The proposal is to have “things” on the boundary layer, representing change of ownership. That means the owner of the process changes from within the system to outside the system. There are some theoretical and practical advantages to this kind of implementation. That method causes the system boundary to be represented as an entity within OPN and OPD. That will allow the architect to “include” the system boundary in the mathematical modeling of the system and thus give him the mathematical framework to decide on the system boundary. Moreover, the proposed method makes the interfaces to the outside world much easier. The change of ownership entity gives one point of contact solution for the outside systems that try to interact with the specific system. The inside of the system can thus be considered as a “black box” for the outside systems. Interfaces from within the system will also become easier. Inside subsystems will have one point of contact to submit their products to regardless of the forms that will be selected to be part of the system. Another advantage is in visualization. It will be much easier for an outside viewer to distinguish where the system ends. The following section presents an example of implementing change of ownership in OPD and OPN.

5.3 Change of ownership – Specific example

As was mentioned at the beginning of the chapter, there are different ways to bind an oil exploration system. Those different options are summarized in the figure below:

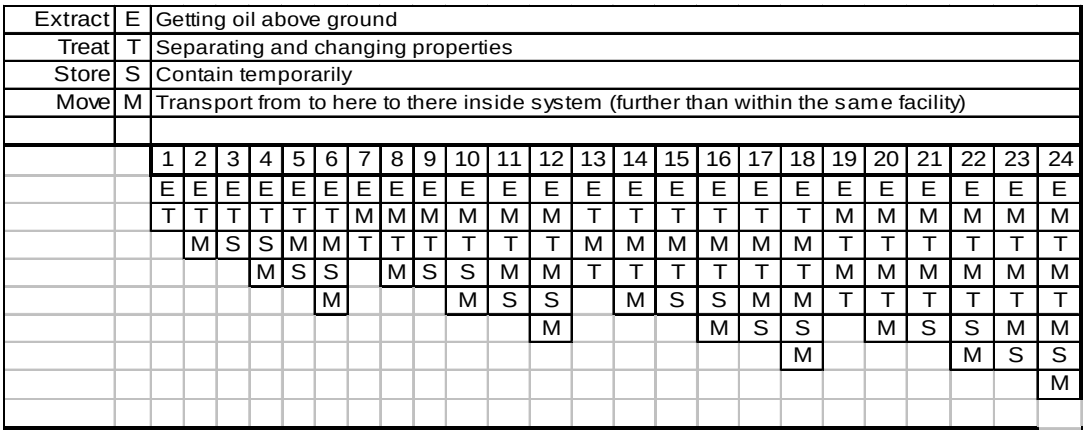


FIGURE 24: The different ways to bound an oil exploration system

As can be seen, different functions can be the last. It could be Treating, Moving or Storing. The way to deal with the floating boundary is by adding a function that represents the function of changing ownership. In our case, we called it Exporting:

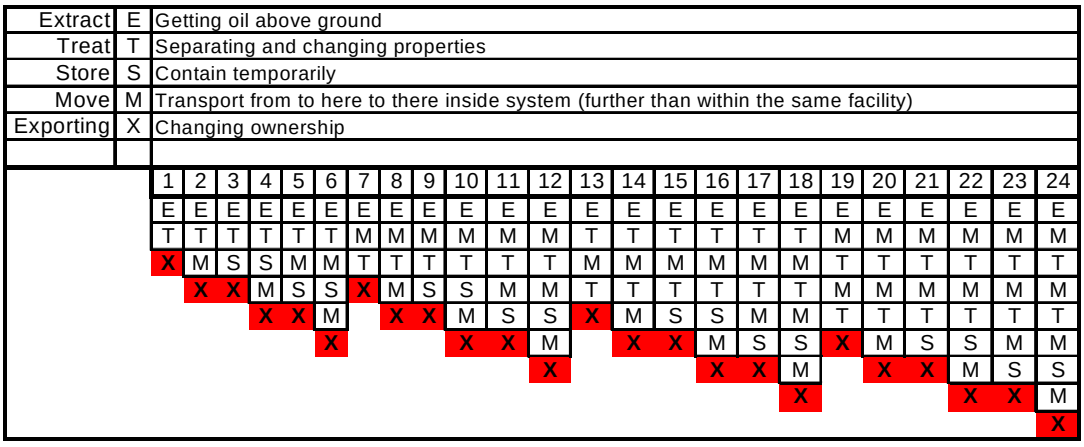


FIGURE 25: System boundary using changing of ownership

The implementation in OPD looks as follows:

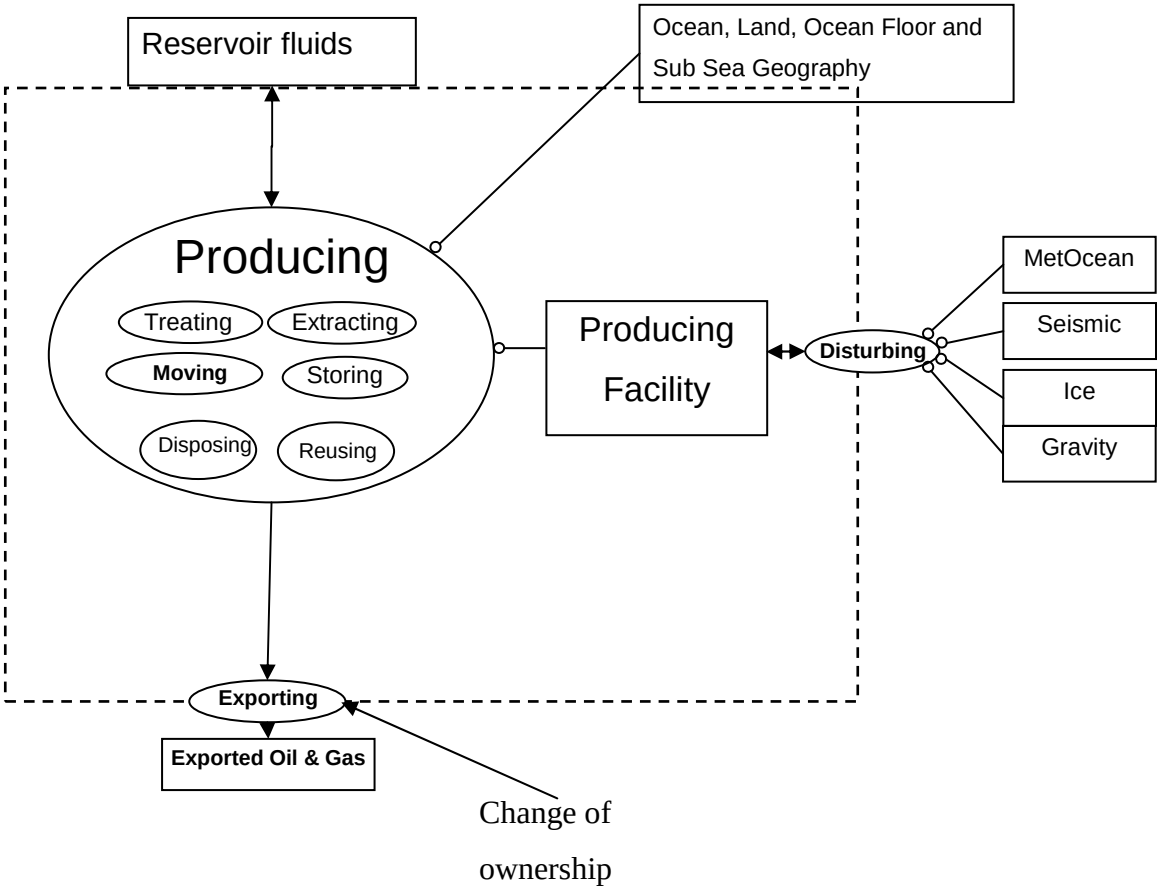


FIGURE 26: Change of ownership – Solution-neutral level 1 OPD model

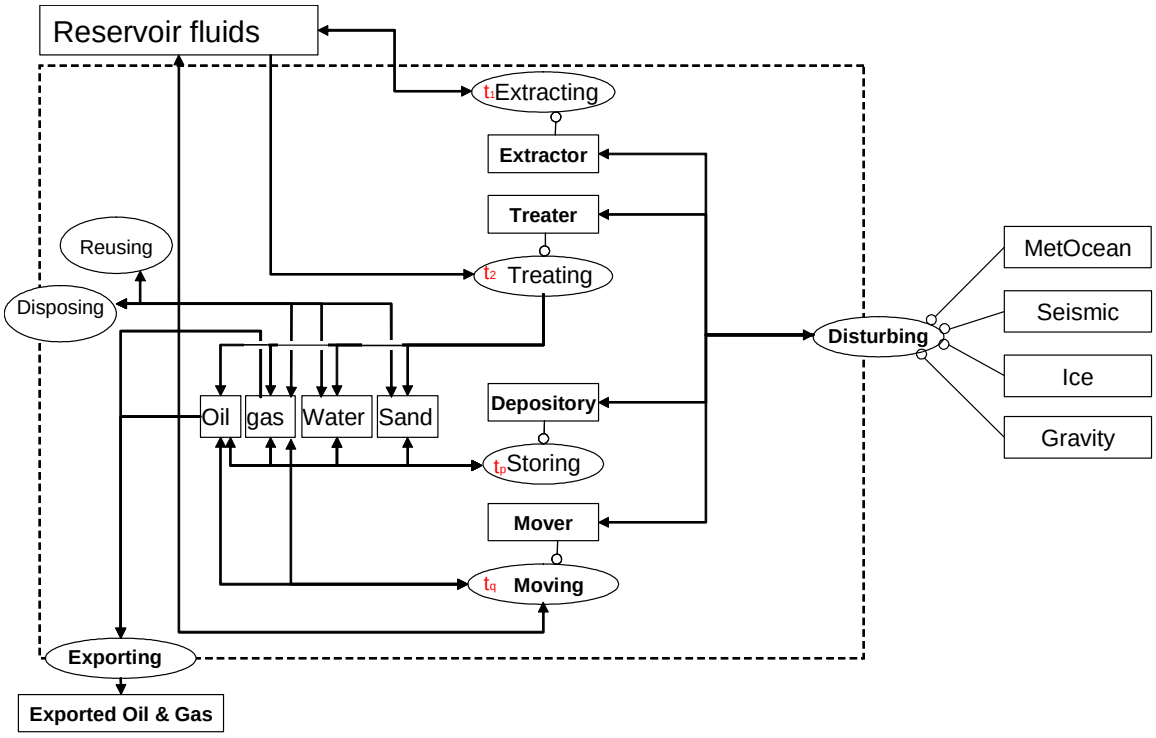


FIGURE 27: Change of ownership – Solution-neutral level 2 OPD model

And the implementation in OPN looks as follows:

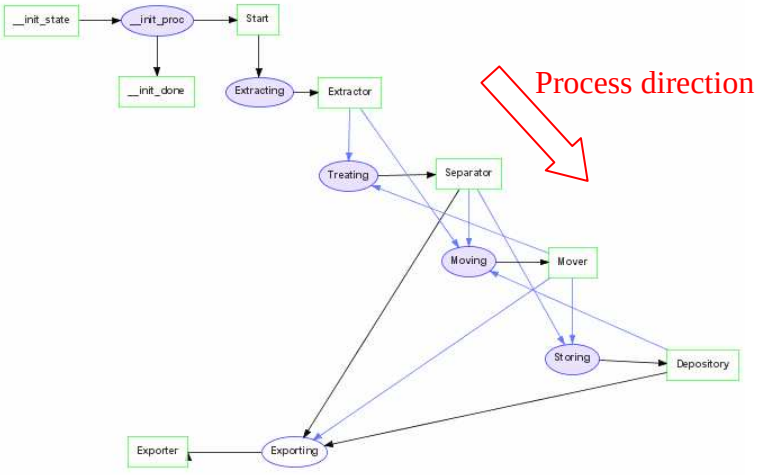


FIGURE 28: Change of ownership – OPN model

5.4 Conclusion

This chapter offers a possible solution to deal with floating boundary in OPD and OPN by incorporating the boundary into the system model. This is done by defining the system boundary as a “change of ownership” process. That definition will give subsystems within the system and interfacing system one point of contact for processes that cross the system boundary. Another advantage is the ability to model the system boundary and to incorporate it in the value calculation in OPN.

6 Coupling/decoupling possibilities between Stakeholder's model and Object-Process model.

6.1 Introduction

As discussed, systems should be measured by the amount of overall value they create for all their stakeholders, taking into account the relative weight of the different stakeholders. A basic question is how to connect the stakeholder's and functional models in order to find the best architecture (the best set of forms and the right context) that generates the highest value to stakeholders.

This chapter presents four levels of possible connections, going from the easiest to implement to the hardest. The predicted ability of each to measure overall value will also increase as the implementation complexity increases.

1. Two separate models: Stakeholders and Object-Process. These models will be minimally connected and only with human interpretation of analysis up to this point. The Stakeholder model will yield the most important factors to consider in evaluating the different architectures. Those factors will be used to rank and screen the different architecture permutations that the Object-Process model will yield after calculating all the possible permutations.

2. Some coupling between the models. The Stakeholders model will be used (in addition to generating a selecting criteria) to generate a set of rules that represent value generated parameters. Those rules will then be incorporated into the Object-Process model. That way, the screening process of the architecture permutation will occur during the permutation-creation process. Additional value is that the permutation-creation processes can be altered (before running it) to focus on process that generate more value (for example, performing treating twice).

3. Adding a stakeholder's evaluation model at the end of the process. The result will look as follows:

   Stakeholders model 1 → Object-Process model → Stakeholders model 2

   The first Stakeholder model will calculate a set of rules that will be used to screen out non-valuable process permutations. The second stakeholder's model will be used to calculate the actual value of each of the remaining permutations. The input for that model will be the permutations along with their important attributes (price, duration, etc.). The output will be the architecture/s that generates the highest amount of value (or a ranking of all the architectures). That value will be calculated by dynamically running the model on each of the architectures.

4. A complete coupling of the Stakeholders and Object-Process models that may run as one model, calculating the best architecture (or ranking all the different permutations) "on the fly."

6.2 Two separate models

At this level the Stakeholder and Object-Process model are physically disconnected. The system architect will be the one to make the connection in order to find the value in each of the architectures that the Object-Process model generates. The system architect will use the following algorithm:

1. Build separate Stakeholder and Object-Process models.
2. Use these models to find characteristics that affect the value gained by the stakeholders. There are several ways to utilize the model to get those characteristics. This topic is currently being studied by Professor Edward Crawley's research group. In general the system architect should select those characteristics that he can easily alter utilizing the architecture. For example, there is usually a strong relation between selected forms and the overall cost of the system. On the other hand there

is usually a weaker connection between the forms of the system and the political power of the stakeholders.

3. Use those characteristics to evaluate the different architectures permutations generated by the Object-Process model. This valuation occurs after running the Object-Process model and there are two possible ranking operations:

   i. Screening out the architecture that does not answer a threshold level – for example, screening out all the architectures that do not satisfy a minimal safety level.

   ii. Ranking architectures by the value they create for a specific characteristic or for a collection of characteristics (for example weighted average). The following figure demonstrates such a ranking where the different architectures are ranked according to the overall NPV they are expected to generate.

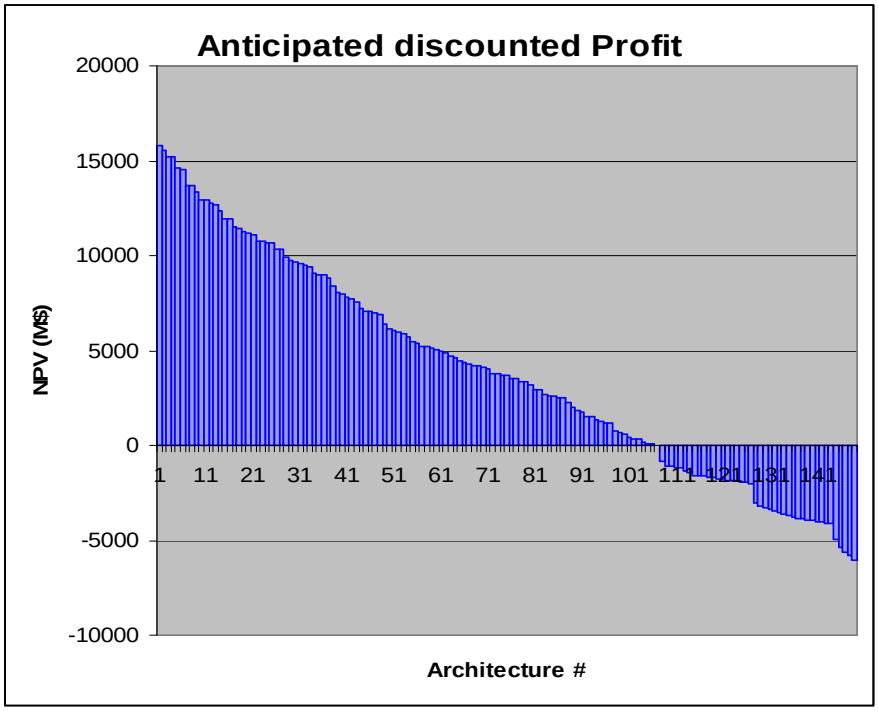


FIGURE 29: Ranking architectures based on important characteristics

The following figure summarizes this possible connection between the Stakeholder's model and the Object-Process model:

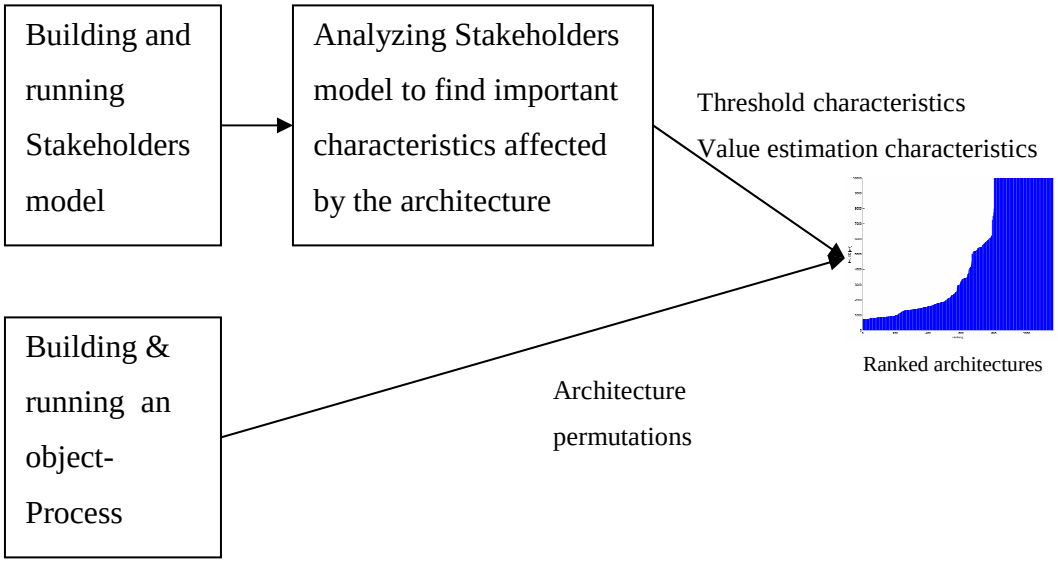


FIGURE 30: Schematic of a two separate model processes

6.3 Some coupling between the models

In this option there will be a connection between the Stakeholder's model and the Object-Process model while the Object-Process model is generating the architectures permutations. The Stakeholders model will be used (in addition to its role in defining a selecting criteria) to generate a set of rules that represent value-generated parameters. Those parameters will be incorporated into the Object-Process model as threshold parameters. That way, the screening process of the architecture permutation will partially occur during the permutation creation process. The immediate benefit is that some of the "bad" architectures will be screened out during the architecture generation process, which will increase the overall process efficiency (since the system will waste less resources on those "bad" architectures).

There are two important points to consider. The first is that this option does not change the previously mentioned serial nature of the overall process. The Stakeholders model will have to be executed before the Object-Process model in order to identify the important characteristics. The change is in the time stamp where those characteristics will

be incorporated into the Object-Process model. The second point is that the connection between the models is still manual, because the connection is unpredictable. Neither the characteristics nor their desired threshold can be estimated prior to running the Stakeholder's model.

The following figure summarizes this possible connection between the Stakeholder's model and the Object-Process model:

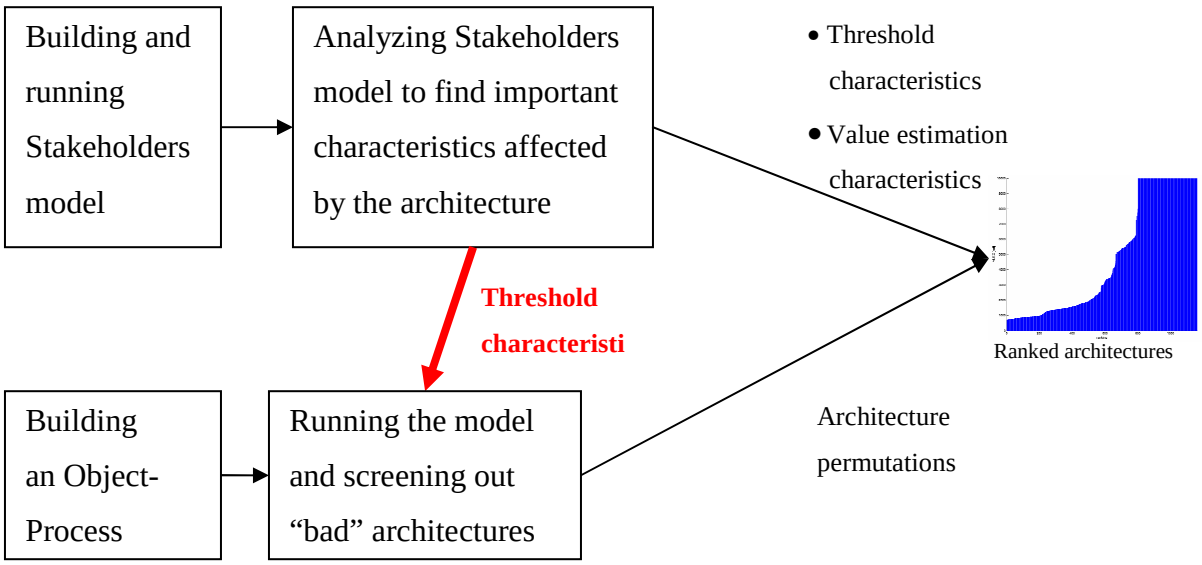


FIGURE 31: Schematic of Some coupling between Stakeholders and process models

6.4 *Adding a stakeholder's evaluation model at the end of the process*

This option will allow a better estimation of the "goodness" of each of the architectures generated by the Object-Process model. The problems in that regard in the previous options were that often it is not easy to find the value-creating characteristics, either because of the high complexity of the Stakeholder's model or because different stakeholders perceive those characteristics differently (subsystem supplier and integrating company will probably associate opposite value to the subsystem's cost).

Additionally, sometimes defining a specific characteristic as the sole metric to evaluate architecture is not enough from several reasons. There could be a more complex connection between the architecture and value. For example, we can look at a system where schedule is a possible value characteristic. The stakeholders in that case do not value schedule linearly. One stakeholder will prefer a system that will be built in less than five years (he is indifferent to how much less than five years) whereas another stakeholders will value schedule exponentially (such that 50% reduction in building time will increase his value by 200%) and so on. Coming up with value formula in those cases can be difficult especially when there are many stakeholders and many possible characteristics. Moreover, in many cases the value characteristic is actually a weighted average of several characteristics. This is possible when the relative weight of each of those characteristics is fixed. There are also cases where it is not true (for example when schedule becomes extremely important when building time passes X years).

In order to solve this problem, a new stakeholder model will be created. This new model will be used to value each of the permutations created by the Object-Process model. The algorithm for utilizing that model will be as follows:

1. The first Stakeholder model will calculate a set of rules that will be used to screen out non valuable process permutations and to find the value characteristics.
2. The Object-Process model will generate all the possible architectures, screening out the non-valuable architectures on the fly. Additionally, it will calculate the value characteristics.
3. The second Stakeholder's model will be used to calculate the actual value of each of the remaining permutations. The input for that model will be the permutations along with their value characteristics (price, duration, etc.). The output will be the architecture/s that generates the highest amount of value (or a ranking of all the architectures). That value will be calculated by dynamically running the model on each of the architectures.

The following figure summarizes this possible connection between the Stakeholder's model and the Object-Process model:

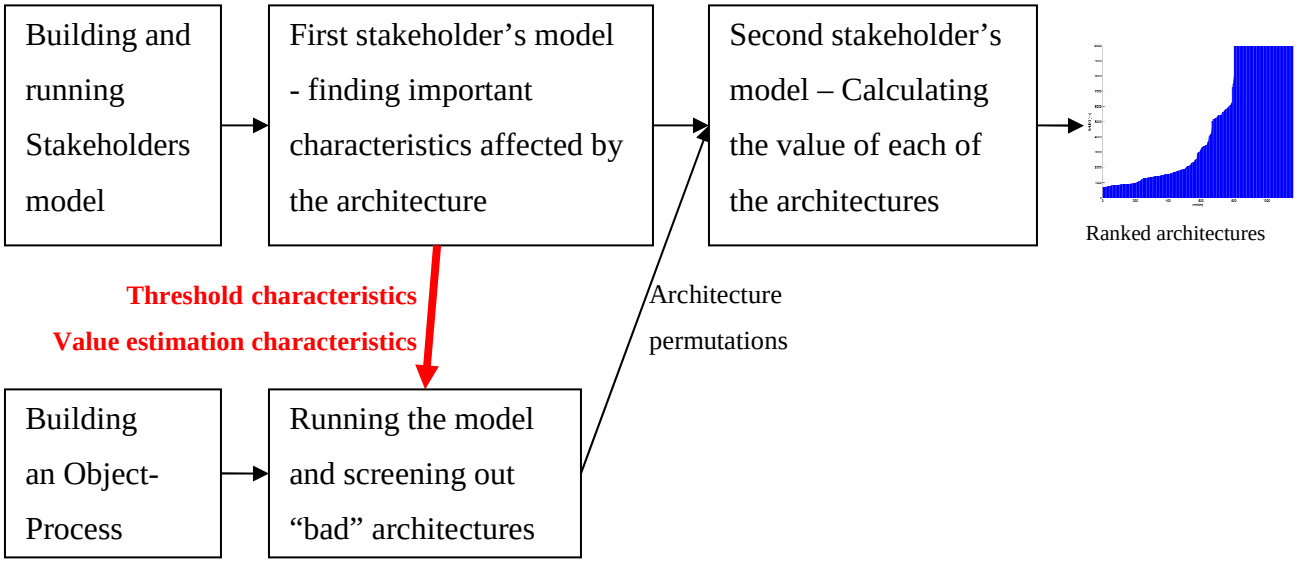


FIGURE 32: Schematic of a process with two stakeholders models

6.5 A complete coupling of the Stakeholders and Object-Process models

In this option, both the stakeholder's and Object-Process models will run together as one model. It is hard to describe the exact characteristics of this option before learning about the practical limitations of the previous options but I can still discuss the issues that this option will address as well as the new options it will enable.

First it will allow a dynamic creation of architectures. Each stage of the architecture creation will be valued against the Stakeholder's model on–the-fly. That will allow the system to make ad-hoc decisions. For example, if a particular form appears to generate high value, the architecture generating system might decide to alter the possible permutation path and to further explore different options associated with that form (for example the use of two forms for each function).

A major problem that option will solve is the high computational time of complex models. This model will reduce the number of permutations generated since all the "bad"

architectures will be screened out on–the-fly (whereas earlier only part of the “bad” architectures were screened out).

Moreover, a smart algorithm incorporated into the model will allow a smart search of the optimal architecture. Those algorithms will basically be optimization algorithms (due to the discrete nature of the architecture they will probably be based on genetic algorithms) that will search the optimal architecture without searching the entire range of architecture. There are several benefits to that approach First, it will increase the system architect’s ability to explore complex systems – the number of permutation is currently the main parameter that limits OPN, and thus complex systems with a large number of expected permutations are simplified in the translation to OPN. An optimization algorithm by nature reduces the number of permutations by focusing on those that lead to the highest value architectures. Second, it will allow a sensitivity analysis – a byproduct of the optimization process is the ability to easily generate a sensitivity analysis. This analysis can help in a cost-effective analysis or act as a base for an isoperformance analysis [14].

6.6 Conclusion

This chapter presents four different options to connect Stakeholders and Object-Process models. While most of the current models utilize the first two options, there is a lot of value to be gained by expending those models to the third and four options. The value is both in allowing a more complex model and in improving the way those models reflect reality and predict the architecture’s performance.

7 Conclusion

Albert Einstein said, “Everything should be as simple as it is, but not simpler.” This thesis was aimed at allowing the system architect to simplify the complexity of the model while not decreasing the ability of those models to represent complexity and complex systems. It investigates existing modeling methods – OPD and OPN [1], [2] and proposes techniques to improve their ability to represent complex systems.

Specifically, four topics are reviewed in the research part of this thesis. The first suggests an algorithm to implement an iterative process in OPN. That algorithm allows a dynamic examination of the possibility to use more than one form to perform a specific function. The second topic suggests a framework that uses a top-to-bottom approach to facilitate generation of “out-of-the-box” solutions to technical problems. The third topic suggests a method to deal with a non-fixed boundary in the architecting phase such that subsystems either within or outside the system keep interfacing with the same object regardless of the actual boundary of the system. The final topic deals with the coupling and decoupling possibilities between the stakeholder’s model and the Object-Process-model.

These four topics, aside from being all related to OPN, are part of an overall solution that will allow OPN to explore a range of solutions much larger than was defined by the architect as an input. The “out-of-the-box” framework allows OPN to suggest solutions of forms that the architect did not think about or was not aware of, while the iterative process increases that capability by allowing OPN to explore all the possible combinations of those solutions. That will “break” the connection of one form to one function – checking the possibility to achieve a certain functionality using a set of forms instead of only one.

The floating-boundary method in its turn will allow OPN to investigate the boundary of the system to find the best set that maximizes overall stakeholder’s value. That will generate a new set of possible solutions where some of the initial functionality could be left outside of the system boundary.

The final piece of this overall solution is to implement increased coupling between the Object-Process model and the Stakeholder's model. The value of the previous topics will increase as it will become easier and faster to measure the overall value of each system, subsystem and form to the system stakeholders.

I believe that this overall solution will allow OPN to increase its ability to deal with a bigger set of issues. An example for a use of that kind of implementation is in portfolio optimization – finding the best future strategy for a company by optimizing its future portfolio. A possible future strategy for a specific company might be to build many new products (in OPN, it will be many new functions). Another option could be to heavily rely on a current product, creating few new ones (in OPN, that means few new functions) or anything in between. Beside this necessity to change the number of functions (that can be solved using the non-fixed boundary and loop methods) that kind of optimization will probably also require a strong connection between the two models of possible portfolios and stakeholders.

Another possible implementation is in economic research looking for a preferred economic strategy. In this kind of research, there are many possible tools that the architect can use (like different monetary tools) by themselves or in combination with other tools to achieve the best results for a specific economy. Those results heavily rely on the behavior of other (basically their stakeholders). Using loops, changing boundaries and strong coupling of models, the architect can offer a great deal of value in that regard.

Achieving that kind of functionality in OPN requires additional research. There is room for additional research at the topic level, for example I believe that the method to generate out-of-the-box solutions is still not robust enough. It can not offer automatic value to any kind of problem, and it still requires heavy human interaction. Another area for research is in finding ways to create models that better reflect reality. One example that was mentioned earlier was the ability to increase the OPN complexity by introducing optimization algorithms that will replace the creation of the entire set of permutations. Another possible area is incorporation of real-option analysis to evaluate future value of forms that are added to the system.

Appendices

APPENDIX A: OPN symbols

As mentioned, OPD uses a set of symbols to describe possible relations and links between objects and processes. This section presents those relations and an example of a model built in OPN.

Decomposition/Aggregation - describes a relationship between a whole and its parts. The symbol used for that is: ▲

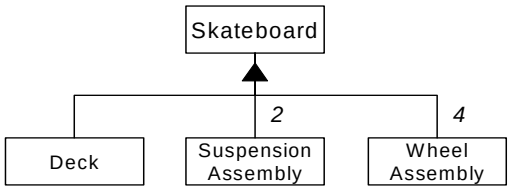


FIGURE 33: OPD example of Decomposition/Aggregation [1]

Characterization/Exhibition – describes the relationship between an object and its features or attributes. It is important to note that some attributes can be states [1], which is a situation in which the object can exist for some positive duration of time.
The combination of all the states describes the possible configuration of the system throughout the operational time. The symbol used for that is: ⧋

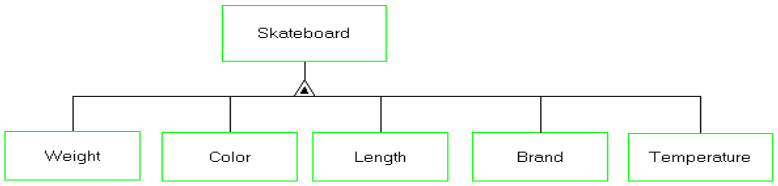


FIGURE 34: OPD example of Characterization/Exhibition [1]

Specialization/Generalization – describes the relationship between a general object and its specialized forms.

The symbol used for that is:△

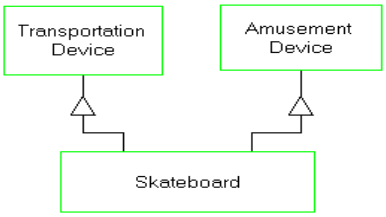


FIGURE 35: OPD example of Specialization/Generalization [1]

Instantiation - describes the relationship between a class of things and instances of the class.

The symbol used for that is: ⧋

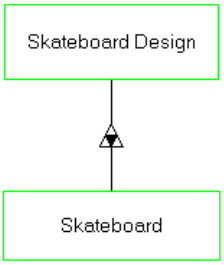


FIGURE 36: OPD example of Instantiation [1]

Another set of symbols is used to describe the possible links between the objects and processes:

- **P changes O** (from state A to B).
- **P affects O (affectee)**
- **P yields O (resultee)**
- **P consumes O (consumee)**
- **P is handled by O (agent)**
- **P requires O (instrument)**
- **P occurs if O is in state A**

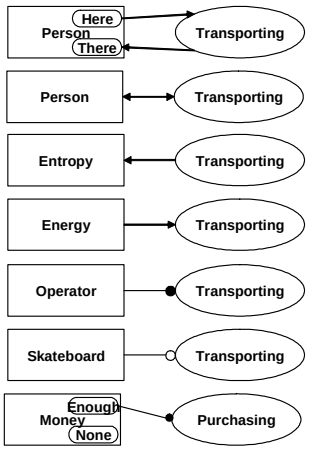


FIGURE 37: OPD symbols for links between objects and processes

Example of a model built in OPN

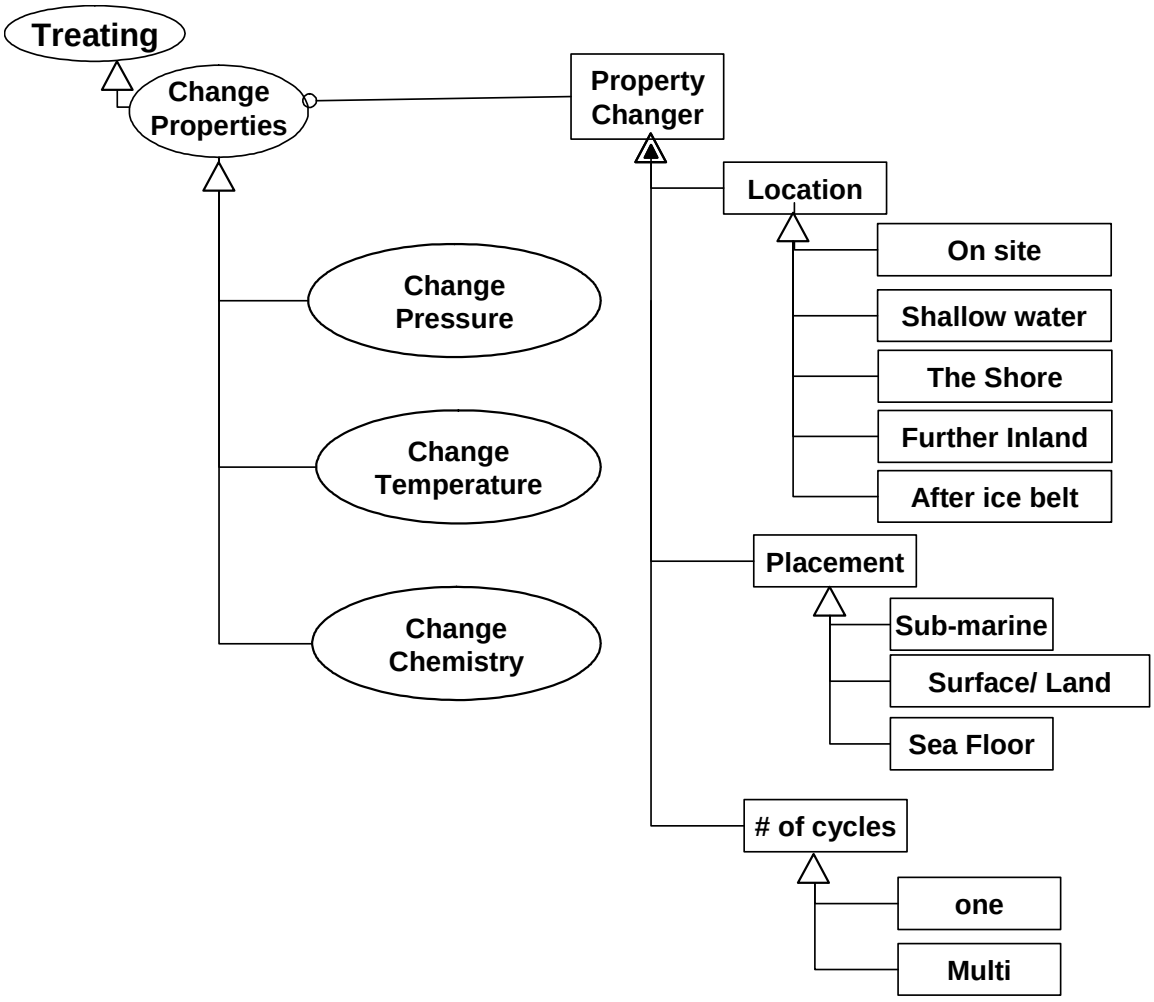


FIGURE 38: Example of a model built in OPD

This example demonstrates how the treating process is represented in OPN. It can be specialized to Change Properties, which in turn can be specialized into Change Pressure, Change Temperature and Change Chemistry. A Property Changer is the instrument used to change property. It is characterized by its location (horizontal), placement (vertical) and number of treatment cycles.

APPENDIX B: An OPN representation of Ice Protection solution system

This section presents the implementation of the ice protection model in OPN:

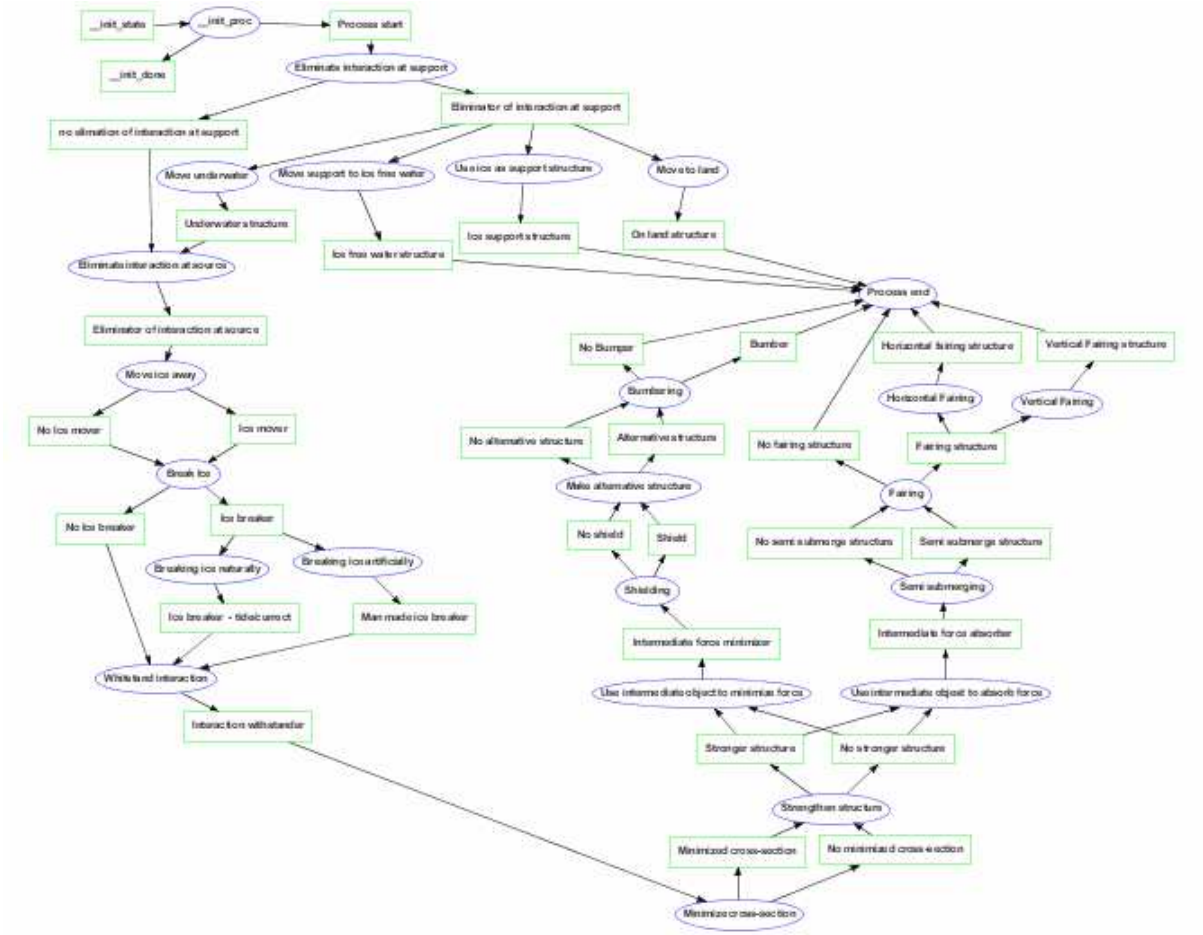


FIGURE 39: An OPN representation of Ice Protection solution system

APPENDIX C: Formulation of the oil exploration system loop example

Following is the set of equations that describe the system and the expected value. The legend that describes the used abbreviations appears at the end of this section.

$$Years\ of\ \Pr oduction = TLT - TTBS$$

For TTBS we assume that the treating system and extracting system can be built in parallel. Thus:

$$TTBS = Max(TSB, ESB)$$

Furthermore, the possible production per year is the minimum between the treating and extracting capacity:

$$PPY = Min(TPY, EPY)$$

The total cost is the sum of the treating and extracting system costs:

$$Total\ Cost = TSBS + ESBS$$

And profit is defined as the NPV over the leasing period:

$$\Pr ofit = NPV\left(building\ the\ sysetm\ priod\right) + NPV\left(Extracting\ oil\ period\right)$$

$$\Pr ofit = -\sum_{n=0}^{TTBS} \frac{Yearly\ Cost}{(1+r)^n} + \sum_{n=TTBS+1}^{TLT} \frac{Yearly\ Income}{(1+r)^n}$$

$$\Pr ofit = -\sum_{n=0}^{TTBS} \frac{Total\ \cos t \,/\, TTBS}{(1+r)^n} + \sum_{n=TTBS+1}^{TLT} \frac{PPY \times PPB}{(1+r)^n}$$

Legend:

TLT – Total Leasing Time
TTBS – Total time for Building The System
TSB – Treating System Building Time
ESB – Treating System Building Time
PPY – Production Per Year
TPY – Treating Per Year Production
EPY – Extracting Per Year Production
TSBS – Treating System Building Cost
ESBS – Extracting System Building Cost
PPB – Price Per Barrel
r – discount rate

APPENDIX D: List of possible solutions to the oil exploration loop problem

Following is a list of all the possible combinations to build extracting and treating subsystems. For each solution there are the different numbers of forms selected as well as the expected discounted profit. The options are ordered from the most profitable to the lowest.

Solution #	# of Large Extracting forms	# of Medium Extracting Forms	# of Small Extracting Forms	# of Large Treating Forms	# of Medium Treating Forms	# of Small Treating Forms	Profit (M$)
1	0	1	1	0	0	2	15837.63
2	1	0	1	0	1	1	15530.68
3	0	2	0	0	1	1	15244.01
4	0	2	0	0	0	2	15240.50
5	1	0	0	0	1	0	14627.50
6	1	1	0	0	2	0	14509.30
7	1	1	0	0	1	1	13719.11
8	0	0	2	0	0	2	13661.13
9	0	1	1	0	1	1	13376.60
10	0	1	0	0	1	0	12966.03
11	1	0	1	0	2	0	12928.93
12	0	1	0	0	0	1	12768.24
13	0	2	0	0	2	0	12647.98
14	1	0	0	1	0	0	12385.60
15	1	1	0	1	0	1	11950.68
16	1	0	2	0	0	3	11942.75
17	0	0	2	0	1	1	11509.18
18	0	1	2	0	0	3	11417.41
19	0	0	1	0	0	1	11283.64
20	2	0	0	0	2	0	11147.36
21	0	1	1	0	2	0	11067.61
22	1	1	0	0	0	2	10786.67
23	2	0	0	1	1	0	10750.22
24	1	1	1	0	1	2	10707.77
25	1	0	1	1	0	1	10640.88
26	0	2	0	1	0	1	10363.43
27	0	0	3	0	0	3	10350.83
28	1	0	2	0	1	2	9884.13
29	0	0	1	0	1	0	9725.78
30	1	0	0	0	0	1	9702.13

Solution #	# of Large Extracting forms	# of Medium Extracting Forms	# of Small Extracting Forms	# of Large Treating Forms	# of Medium Treating Forms	# of Small Treating Forms	Profit (M$)
31	0	1	0	1	0	0	9575.34
32	0	0	2	0	2	0	9487.24
33	1	1	0	1	1	0	9429.48
34	0	1	1	1	0	1	9053.63
35	2	0	0	0	1	1	9031.21
36	1	1	1	0	0	3	8954.13
37	0	1	2	0	1	2	8801.40
38	1	0	1	1	1	0	8362.90
39	0	2	0	1	1	0	8100.79
40	0	0	3	0	1	2	7969.40
41	1	1	1	0	2	1	7822.38
42	0	0	2	1	0	1	7743.83
43	1	2	0	0	1	2	7568.76
44	1	0	2	0	2	1	7211.74
45	0	0	1	1	0	0	7066.89
46	0	1	1	1	1	0	7042.15
47	0	2	1	0	2	1	6958.12
48	2	0	0	0	0	2	6907.13
49	0	1	2	0	2	1	6356.23
50	1	2	0	0	0	3	6158.69
51	2	0	1	0	2	1	6013.43
52	0	0	2	1	1	0	5975.57
53	2	0	0	2	0	0	5849.00
54	0	0	3	0	2	1	5745.60
55	2	0	1	1	0	2	5449.20
56	2	0	1	0	1	2	5367.68
57	1	1	1	1	0	2	5240.17
58	1	2	0	1	0	2	5203.28
59	1	1	1	0	3	0	5121.77
60	0	3	0	1	0	2	5003.37

Solution #	# of Large Extracting forms	# of Medium Extracting Forms	# of Small Extracting Forms	# of Large Treating Forms	# of Medium Treating Forms	# of Small Treating Forms	Profit (M$)
61	1	1	0	2	0	0	4993.49
62	1	0	2	1	0	2	4840.34
63	1	0	2	0	3	0	4712.83
64	0	2	1	1	0	2	4594.43
65	0	2	1	0	3	0	4476.02
66	1	0	1	2	0	0	4382.85
67	2	0	1	0	0	3	4267.00
68	1	0	3	0	0	4	4211.05
69	0	1	2	1	0	2	4185.49
70	0	2	0	2	0	0	4137.98
71	0	1	2	0	3	0	4067.08
72	0	0	3	1	0	2	3785.65
73	0	1	3	0	0	4	3758.80
74	1	1	2	0	0	4	3738.72
75	0	0	3	0	3	0	3658.14
76	0	0	4	0	0	4	3542.71
77	0	1	1	2	0	0	3527.34
78	2	1	0	0	2	1	3394.51
79	2	1	0	1	1	1	3389.19
80	2	0	1	1	1	1	3173.10
81	1	2	0	1	1	1	2936.94
82	0	0	2	2	0	0	2916.71
83	1	1	1	1	1	1	2720.85
84	2	1	0	0	1	2	2598.04
85	2	1	0	1	0	2	2592.73
86	1	0	2	1	1	1	2504.76
87	0	3	0	1	1	1	2484.68
88	0	2	1	1	1	1	2268.60
89	0	1	2	1	1	1	2052.51
90	0	0	3	1	1	1	1836.42

Solution #	# of Large Extracting forms	# of Medium Extracting Forms	# of Small Extracting Forms	# of Large Treating Forms	# of Medium Treating Forms	# of Small Treating Forms	Profit (M$)
91	2	1	0	0	0	3	1801.58
92	1	1	2	0	1	3	1516.79
93	1	0	3	0	1	3	1477.19
94	1	2	1	0	1	3	1307.16
95	0	2	2	0	1	3	1242.87
96	0	1	3	0	1	3	1203.27
97	0	0	4	0	1	3	1163.66
98	1	2	1	0	0	4	784.00
99	2	1	0	1	2	0	666.84
100	2	0	1	1	2	0	627.24
101	1	2	0	1	2	0	402.68
102	1	1	1	1	2	0	363.08
103	1	0	2	1	2	0	323.48
104	0	3	0	1	2	0	128.76
105	0	2	1	1	2	0	98.92
106	0	1	2	1	2	0	59.32
107	0	0	3	1	2	0	19.72
108	0	2	1	2	0	1	-825.97
109	0	0	4	0	2	2	-1052.36
110	1	0	3	0	2	2	-1084.40
111	0	1	3	0	2	2	-1184.06
112	1	1	2	0	2	2	-1216.11
113	0	2	2	0	2	2	-1305.70
114	3	0	0	0	2	1	-1466.18
115	0	0	3	2	0	1	-1572.63
116	1	0	2	2	0	1	-1604.67
117	2	0	1	2	0	1	-1626.66
118	3	0	0	1	1	1	-1647.71
119	0	1	2	2	0	1	-1694.27
120	1	1	1	2	0	1	-1726.32

Solution #	# of Large Extracting forms	# of Medium Extracting Forms	# of Small Extracting Forms	# of Large Treating Forms	# of Medium Treating Forms	# of Small Treating Forms	Profit (M$)
121	2	1	0	2	0	1	-1758.36
122	3	0	0	2	0	1	-1819.18
123	0	2	1	2	0	1	-1825.97
124	1	2	0	2	0	1	-1858.02
125	3	0	0	1	0	2	-1897.79
126	0	3	0	2	0	1	-1947.62
127	3	0	0	0	0	3	-1986.45
128	0	0	4	1	0	3	-3023.12
129	0	0	3	2	1	0	-3167.87
130	0	1	3	1	0	3	-3302.16
131	1	0	3	1	0	3	-3364.16
132	0	1	2	2	1	0	-3446.91
133	1	0	2	2	1	0	-3508.91
134	1	1	2	1	0	3	-3643.20
135	0	2	1	2	1	0	-3725.96
136	1	1	1	2	1	0	-3787.95
137	2	0	1	2	1	0	-3849.95
138	2	1	1	0	1	3	-3896.42
139	2	1	1	0	0	4	-3911.95
140	1	2	1	1	0	3	-3932.58
141	0	3	0	2	1	0	-4005.00
142	1	2	0	2	1	0	-4067.00
143	2	1	0	2	1	0	-4128.99
144	2	1	1	1	0	3	-4149.63
145	0	0	4	1	1	2	-4974.19
146	0	1	3	1	1	2	-5397.95
147	1	0	3	1	1	2	-5609.83
148	0	2	2	1	1	2	-5821.71
149	1	1	2	1	1	2	-6033.59

APPENDIX E: Comparing parallel and serial building models for oil extraction systems

Chapter Three in this thesis refers to incorporating loops into OPN. This section will broaden the example in that chapter with the target of presenting the difference between parallel and serial building of forms. Parallel building means that if two or more forms are built to perform a certain function, they will be build in parallel, each starting to work as its building is complete. Serial building means that those forms are built one after the other and they all start to work at the same time – after the last form is ready.

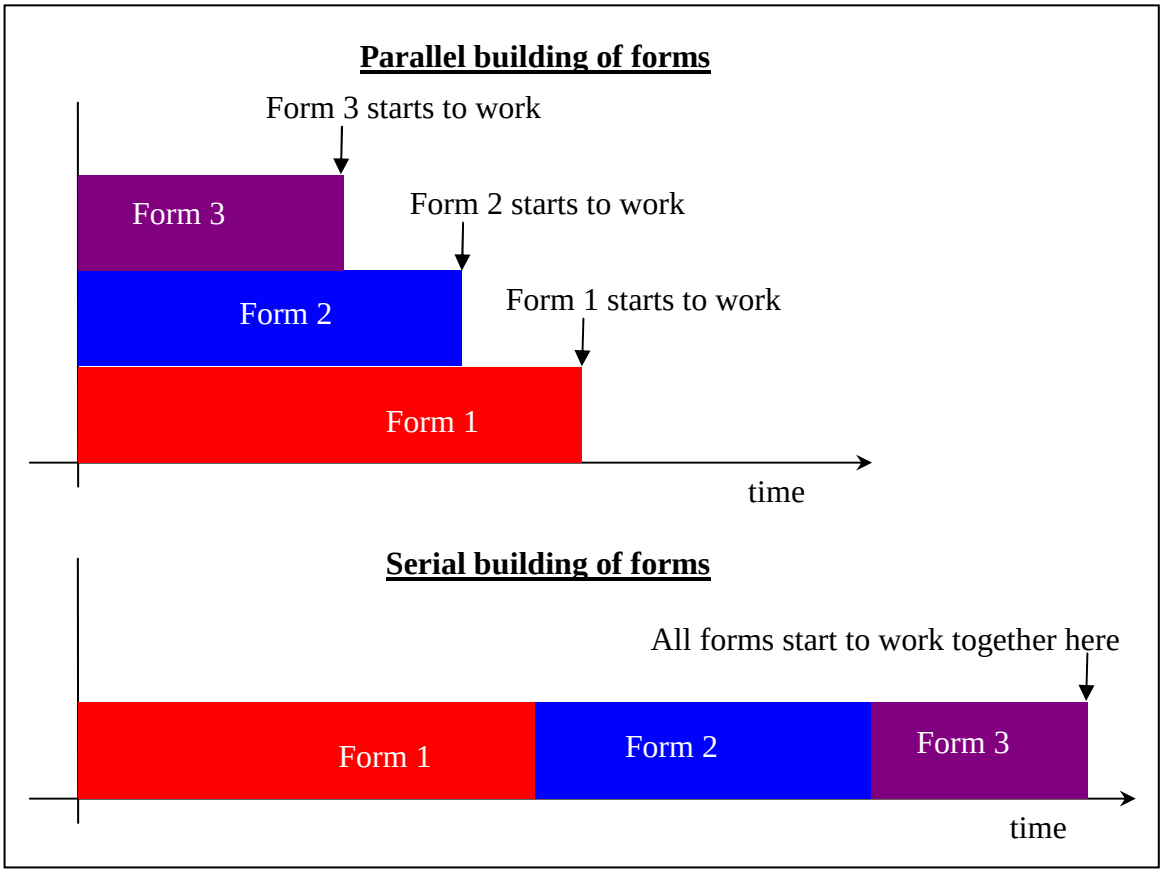


FIGURE 40: Difference between parallel and serial selection of forms

When applying these approaches to the oil exploration example, presented in Chapter Three, each approach yields a different result and a total number of distinct solutions. The serial approach yields 149 distinct solutions whereas the parallel method only 80. The NPV difference is also significant. There is a difference of over $5B between the most profitable solutions of the two methods, as shown in the figure below.

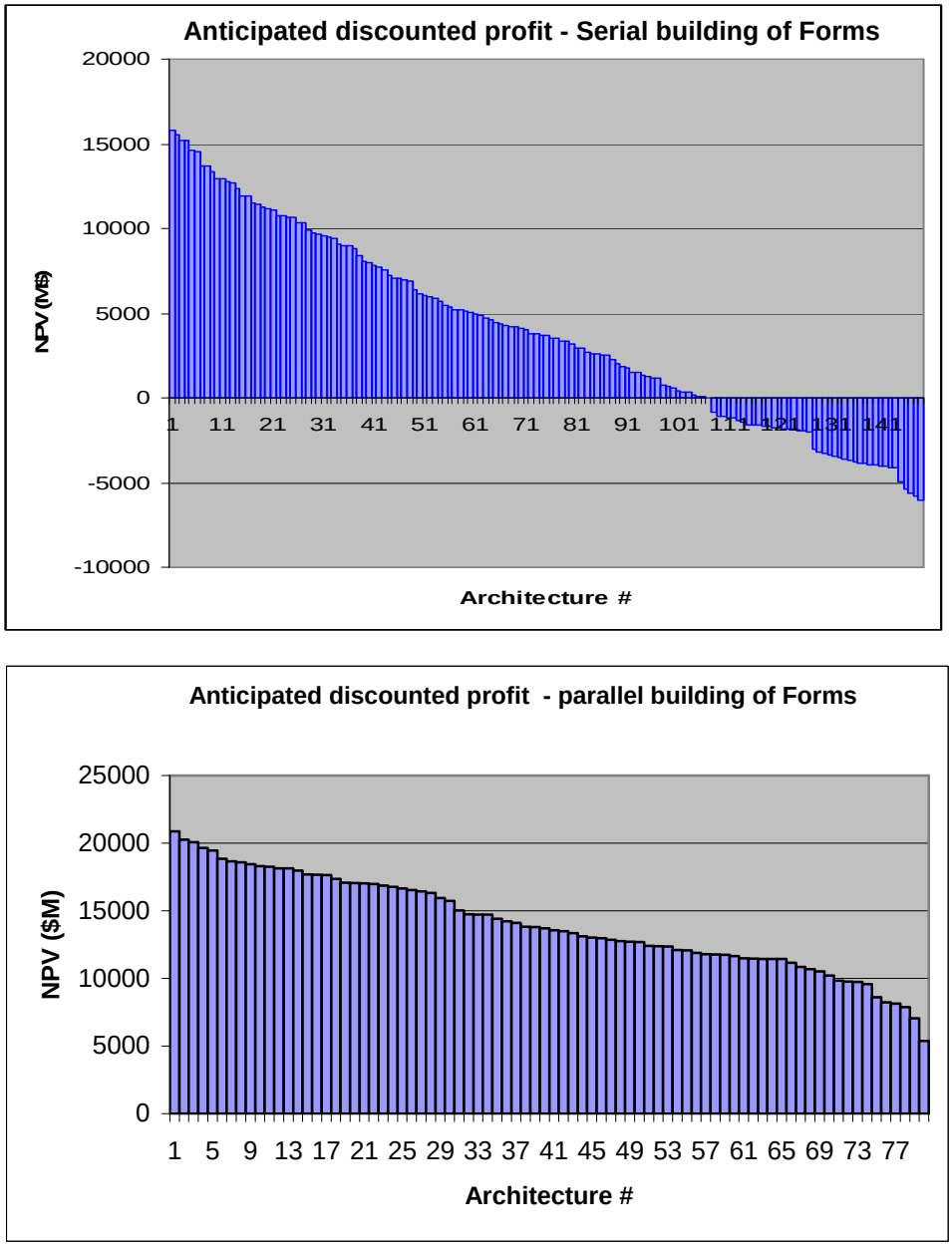
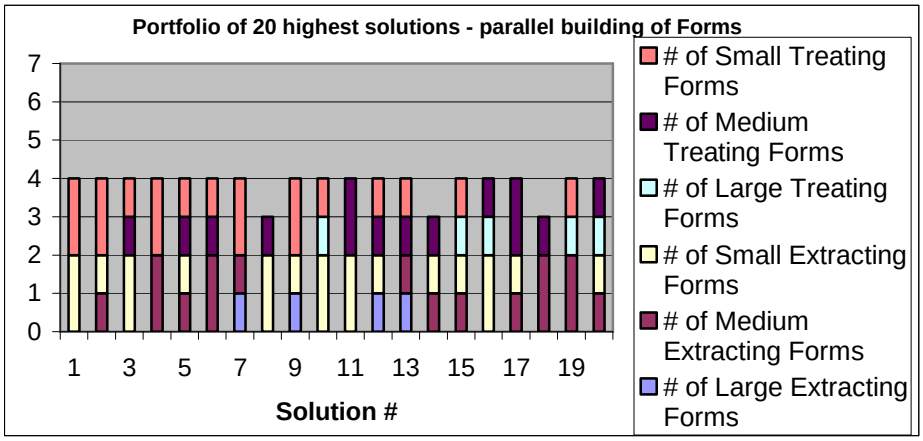


FIGURE 41: Anticipated Discounted Profit - parallel and serial selection of forms



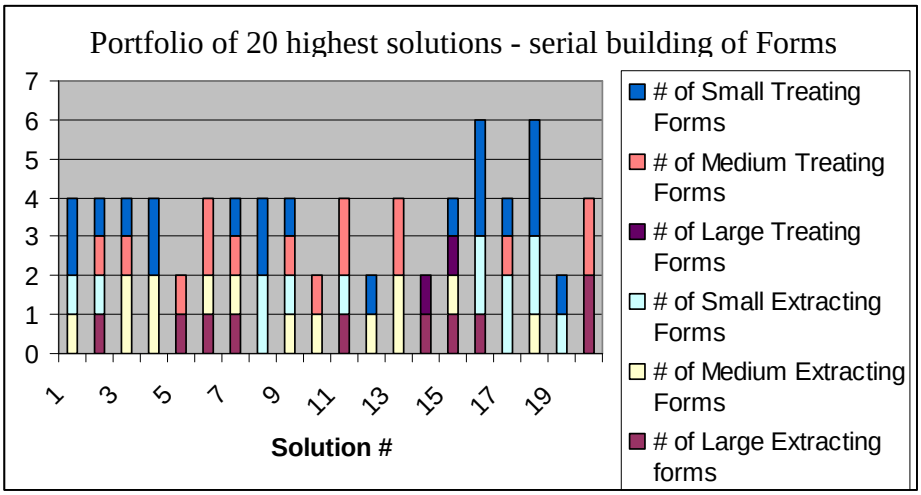


FIGURE 42: Highest Solution Portfolio - parallel and serial selection of forms

Discussion

This problem imposes two possible constraints. The first is the total amount of oil in the field and the second the leasing time. The search for a possible solution (by adding new forms to fulfill a certain function) will be terminated if the existing forms reach the maximum possible field capacity or if there is no sense in creating new forms since reaching the end of the leasing period.

In the parallel approach, the active constraint was the maximum oil capacity in each of the possible solutions, because there was no accumulation of building time (all the forms were built in parallel). Moreover the number of forms for treating and extracting is relatively constant at two forms for each. In the serial approach both

constraint were active, each for different solutions. When many smaller forms were used the active constraint was usually the limit due to the leasing time and when the bigger forms where used the active constraint was usually the maximum oil .That inconsistency in the active constraint is also partially responsible for the inconsistency in the number of forms used for each of the solutions.

Generally, the right approach to the way more than one form should be incorporated into functions depends on the nature of the project. There are projects where most of the forms are built in serial (for example, due to the same resource being needed for all forms), whereas a parallel approach might be adequate in other cases where the start time of operability is important (for example, if the discount rate is high or the total project time is limited – as in the oil example). Additionally, there are cases where a combination of parallel and serial is the best reflection of reality. An example might be a bridge, where some forms (for example foundations) can be built only in a serial way onsite, whereas the other structural forms can be built in parallel offsite but assembled serially onsite. This ability to combine serial and parallel approaches was not implemented into OPN during this exercise. The complexity of that task is not only in the NPV formulation. Some of the parameters act differently in serial than in parallel. Building cost, for example, might be different since some of it is based on fixed costs (for example, the cost of buying the required equipment). Building several forms in serial might split that cost, whereas doing it in parallel will require each form to “pay” for the entire fixed cost. Incorporating that into an OPN model will increase the model’s complexity.

8 Bibliography

[1] E. F. Crawley (2006), System architecture - course notes, MIT

[2] I. Reinhartz-Berger, D. Dori, "OPM/Web – Object-Process Methodology for Developing Web Applications", Annals of Software Engineering 13 pp. 141–161, 2002.

[3] E. Rechtin, M. Maier, The art of systems architecting, CRC, 2002.

[4] D. Dori, Object-Process Methodology, Springer, 2002.

[5] M. J. Kinnunen, Complexity Measures for System Architecture Models, MIT Thesis. 2006.

[6] E. F. Crawley, W. Simmons, Towards a Formalism for System Architecture - From Value to Architecture, MIT, October 2006.

[7] W. Simmons, B. Koo, E. F. Crawley, Space Systems Architecting Using Meta-Language, 56th International Astronautical Congress, 2005.

[8] I. Reinhartz-Berger, D. Dori, Object-Process Methodology (OPM) vs. UML: A Code Generation Perspective, 4th CaiSE/IFIP8.1 International Workshop on Evaluation of Modeling in Systems Analysis And Design (EMMSAD04), Riga, Latvia, 2004.

[9] W. Simmons, B. Koo, E. F. Crawley, Architecture Generation for Moon-Mars Exploration Using an Executable Meta-Language, AIAA space, CA, 2005.

[10] D. Clausing, D. Frey, Effective Innovation, ASME Press, 2004.

[11] V. Fey, E. Rivin, Innovation On Demand, Cambridge University Press, NY, NY, 2005.

[12] H. L. McManus, Space System Architecture - Final Report of SSPARC: the Space, Systems, Policy, and Architecture, Research Consortium (Thrust II and III), MIT Lean Aerospace Initiative, September 2004.

[13] K. T. Ulrich, S. D. Eppinger, Product Design and Development, 2nd edition, New York, Irwin/McGraw-Hill, 2000.

[14] O. L. de Weck, M. B. Jones, Isoperformance: Analysis and Design of Complex Systems with Known or Desired Outcomes, 14th Annual International Symposium of the International Council on Systems Engineering (INCOSE), 2004