Towards Automatic Systems Architecting

Felipe Simon[a,1], Gustavo Pinheiro[a,1] and Geilson Loureiro[b]

[a]Department of Computer Engineering, Aeronautics Institute of Technology, Brazil.
[b]Laboratory of Integration and Testing (LIT), Brazilian Institute for Space Research (INPE), Brazil.

Abstract. This article intends to shed new light on the system design process. We here suggest the possibility of combining simulation features of an executable meta-language called Object-Process Network (OPN) with the descriptive power of well-known modeling languages such as Object-Process Methodology (OPM), Structured Analysis (SA) or SysML. In the Systems Architecture domain, a great issue one always faces is the great number of options to be considered when designing a system. We must keep in mind that modeling the space of options is actually different from modeling the system of interest. The traditional modeling tools allow us to specify a unique solution, when we should consider the whole set of feasible architectures. On the other hand, OPN is able to help architects to assess all these possible configurations but, as a decision-support tool, it doesn't offer the descriptive power OPM, SA and SysML do.

Keywords. Complex Engineering, meta-language, OPN

1 Introduction

The process of designing complex socio-technical systems often involves tasks such as transferring knowledge across different domains and computing parameters of interest. In addition, the space of options to be considered increases as the system being studied becomes more complex. Experience proved essential the use of architectural reasoning techniques when developing such systems. The concept behind these techniques is that reasonable decisions can be made evaluating relevant parameters to that specific system, often related to cost and time issues. Nonetheless, we have had only tools and techniques able to tackle specific needs. No one could handle the main needs in an integrated approach. That is why Object-Process Network (OPN) [5] turns out to be a unique tool in system architecting. It unifies the processes of decomposing the problem (and thus being able to manage

[1] Undergraduate Students (Computer Engineering), Rua H8-C 302, Campus do CTA, Sao José dos Campos, SP - Brazil; Tel: +55 (12) 39477902; Cel: +55 (12) 8134-9758; Email: gusta.pinheiro@gmail.com, felipeeng08@gmail.com

complexity), composing (and analyzing the system as a whole) and eventually evaluating possible architectures.

Essentially, this executable meta-language is a decision-support tool that provides a systematic way of using computational tools to reason about decisions during the early stages of system development.

Meanwhile, system modeling frameworks such as Structured Analysis [2], UML [4] and, more recently, SysML [12] stand as a descriptive method that gives us a static view of the system. This means that they do not give us the chance of simulating or computing parameters. Most customers understand structured methods better than others, though. As communication with stakeholders and users is a main reason for modeling a system, having a final model in Structured Analysis turns out to be a great way of presenting the results yielded by OPN. Thus, the great deal about combining descriptive languages with OPN is, given feasible architectures, we can automatically compose meaningful and communicative models.

Not only having a final model in a descriptive language but also defining the options and constraints that bound our problem into one of these models turns decision-makers experience easier, without great difficulties to understand what is being modeled.

The objective of this research is to propose a new approach to conceive complex systems. As the descriptive and communicative language, we have chosen Object-Process Methodology (OPM) [1], because of its ability to specify both structural and behavioral aspects of a system using a small set of symbols. This article is a first step into the ideal "automatic conception of a system". This final objective would allow us to pick up the best architecture once the functions to be performed and constraints have been set up.

The specific goal of this article is to describe what we now call "The New Approach", explaining the transition between a descriptive language into a decision-support tool (OPN), and eventually translating the results of the simulation process back to the original notation. The methodology that gives support to the suggested approach will be carefully explained in Section 3 of this article. In section 2, we give a brief description of the modeling languages considered in this research.

2 The Current Approach

When developing a new system or enhancing an existing one, the common and current approach adopted is to model the problem and its solution into models in various system description languages, among them SysML, Structured Analysis and OPM. Even though Structured Analysis for instance tries to model the problem (its requirements) aside from implementation aspects, it is inevitable tending to a particular architecture for the problem, rather than considering many possible solutions. These "static" languages do not offer simulation or computation features, they are purely visual.

Currently, decision support tools are completely separated from system architecture modelling tools. So when deciding you do not have instruments for a

common visual understanding of the system architecture and, on the other hand, when modelling you do not have instruments for deciding which way to move forward with required model detailing.

State of the art is modelling frameworks, on one hand, such as OPN providing a full set of combinatorial options and strong decision support, but needing enhancement on visual modelling detailing and, on the other hand, the ones such as Structured Analysis, SysML and OPM, that provide comprehensive and detailed system models but with no decision support capability. In other words, the decision process and solution architecture modelling are done separatedly leading, for example, to decisions with less than available systems information or to detailed models that will never be used.

This section provides a brief review on OPN and OPM, once they will be used in the examples in the following sections.

2.1 Object-Process Network

Studies regarding system architects' current modeling process proved essential the development of a tool that could give support to tasks one always faces when developing a system: enumerating the space of possibilities, given constraints that bound the problem; computing all architectures, based of the options assigned in the first task; and, eventually, evaluating all those solutions, leading us to the preferable ones. Prof. Benjamin Koo's research gave birth to Object-Process Network, a meta-language based on a small set of linguistic primitives that unifies these three tasks in systems architecting. A tool [10] based on Java programming language gives support to this meta-language.

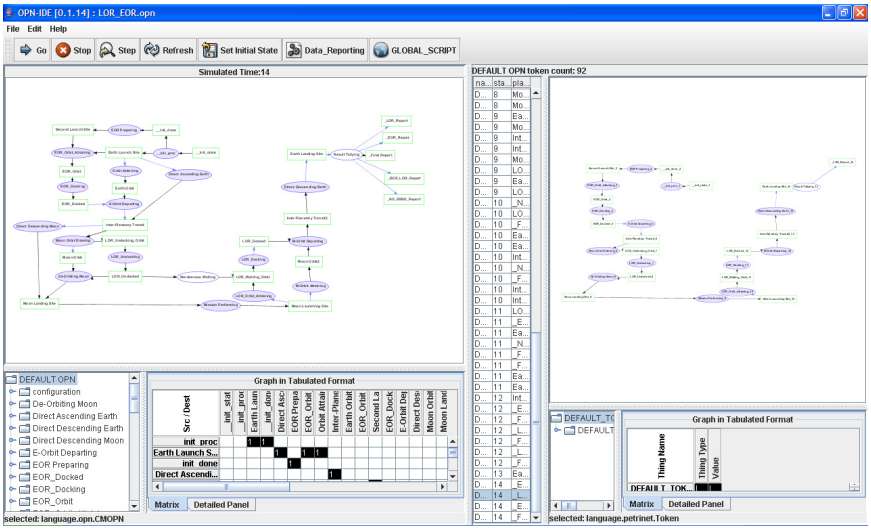

Figure 1. Screen capture of the OPN tool.

An OPN graph is solely composed by processes, objects and links that connect these two entities (only between different types of blocks). A process is defined by functions that modify chosen parameters. Thus, the best architectures can be inferred through analysis of final values of those parameters. Each "thread" in an OPN graph will lead to a different set of "final values". Figure 1 shows on the right an OPN graph and on the left a possible "path" to be followed. For further information on OPN's ongoing research, see References: [5,6,7,8].

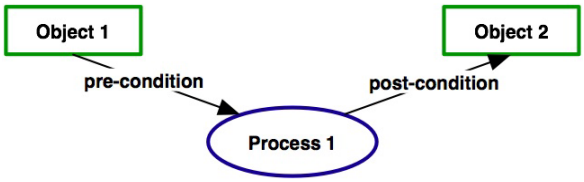

Figure 2. OPN Overview: An object is related to a process through a pre-condition or a post-condition [7]

OPN's highlight is the capability of helping stakeholders making architectural decisions. Currently, OPN does not offer the possibility of converting a series of suggested decisions into an actual model of the system of interest. Besides that, modelling the space of options directly into an OPN graph may become a difficult task as our system becomes more complex.

2.2 Object-Process Methodology

Model multiplicity due to excess symbols has been pointed out as a serious weakness in Unified Modeling Language (UML). A new approach to design information systems was developed in order to overcome the excess notation problem. Object-Process Network uses three types of entities: object, process (modifies the state of an object) and state. Many types of links are available to define which kind of relation the blocks have with each other. These relationships can be either structural or procedural. The first one expresses persistent, long-term relation; the latter one expresses the behavior of the system. Another OPM feature is that it supports graphical and textual models: Object-Process Diagrams (OPD) presents the system through symbols for objects, processes and links; Object-Process Language (OPL) presents the system through sentences. Object-Process Case Tool (OPCAT) [11] is the software developed to demonstrate the applicability of this methodology.

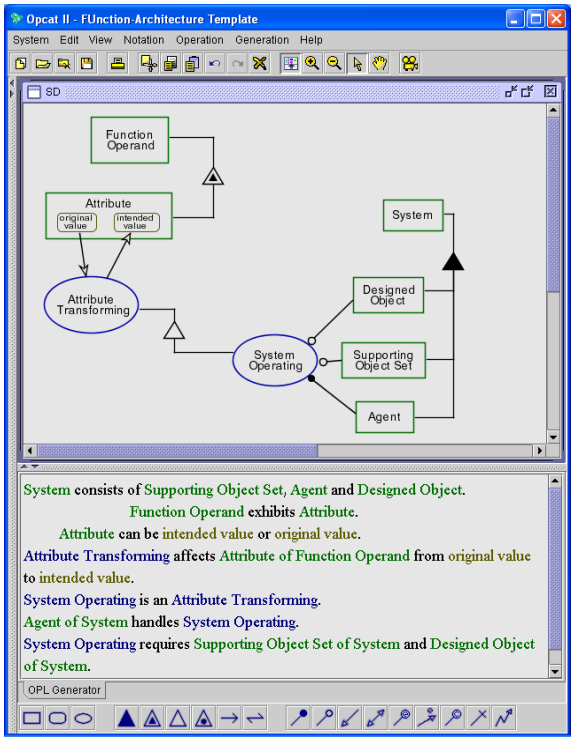

Figure 3. Screen capture of OPCAT [9].

3 The New Approach

The primary objective of this study was to verify the viability of translating a model in traditional modeling tools into an OPN graph. Further studies showed that these languages are not compatible. Actually, OPN has a different purpose than traditional modeling tools. But the association of these two different approaches would lead us to a powerful architecture modelling tool.

Rather then considering a universe of possibilities, currently descriptive languages tend to bias one instance of the system. Nevertheless, each of these languages presents a unique feature. Structured Analysis, for example, allows us to create hierarchy of models that do not depend on their implementations. At each level of abstraction, we could use OPN to help choosing the best architectures to develop. This latter concept can also be applied using OPM as the descriptive language, as we will show with the example to be developed.

All these ideas regarding the union of existing modeling languages gave birth to what we call “The New Approach”. Essentially, it means getting the best of OPN and descriptives languages at once. Figure 4 illustrates “The New Approach”.

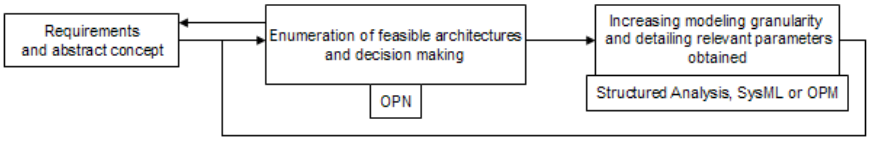

Figure 4. Schema for the proposed approach.

Its first main idea is that we have to define initial and final states related to the system. For example, were we to apply this concept to the Apollo Program the initial state should be "Man on Earth" and the final should be "Man on the Moon". In order not to have infinite possibilities, we have to define the "boundary conditions" of the system. They are the main direct obstacles that hinder the change from an initial to a final state. One may ask: How are we to overcome these "obstacles"? For each of them, a new "subsystem" is designed, allowing these obstacles to be overcome. For an instance, what hinders the man going from Earth to the Moon? The first of the restrictions would be "Distance". For this specific restriction, a given number of subsystems (options) is available.

This process can continue, if for this subsystem to work properly new boundary conditions have to be added. We should stop iterating when the subsystems generated are "minimum blocks", in the sense that there are no more boundary conditions and there is no need to zoom into them. This idea is schematized in figure 5. It is what we will start calling "The General Concept" for The New Approach.

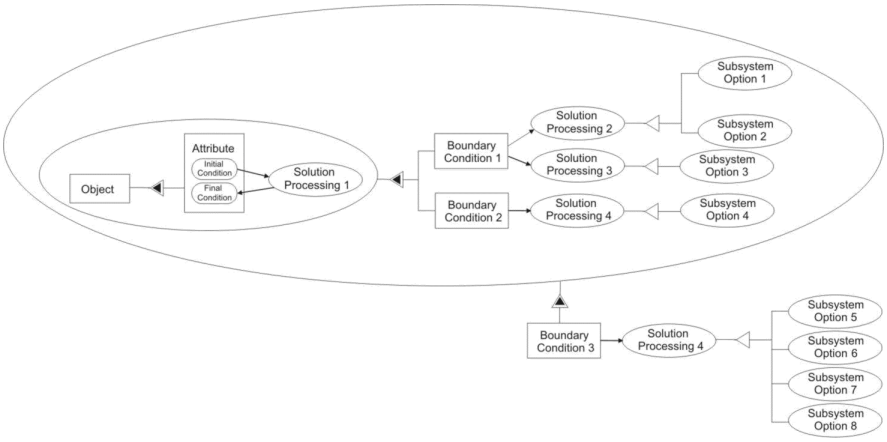

Figure 5. OPM model for The General Concept.

One could argue that the concept here applied does not allow us to make high-level decisions, we would be always obliged to go down into lower levels (generating the "new subsystems"). That's not true. The problem is that starting to

make decisions at top-levels means having fewer parameters to evaluate the architectures. Imagine that we are starting to develop a complex new system, and there are three mainstreams (and completely different) to be followed. Unless this is not a completely new system, we would not have parameters to infer best architectures without going down to lower levels on each of the mainstreams. Simulating at a top level with only macro-parameters could be deceiving. That is why we suggest continuing the iteration process until "minimum subsystems" are modeled. Defining a subsystem as "minimum" is a purely a matter of convenience or availability. For instance, if it is possible to infer parameters that describe this minimum system (for example, from an available database), then there is no need to continue iteration process for it. In other situations, it can be more convenient to develop an entire model for a subsystem and only then it will be possible to describe it in terms of system's main parameters. Note that Subsystem Option 1 and Subsystem Option 2 are regarded in our model (figure 5) as minimum subsystems.

The essential and first step that will govern the development of the rest of the model is the definition of initial and final states and the conditions related to the central problem. The examples below (figure 6 and figure 7) show familiar systems and the definition of the change of states linked to the macro problem (developed with OPCAT II – Object Process Case tool [11]).

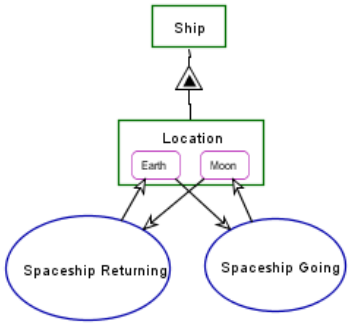

Figure 6. Definition of Change States for the system "Spaceship from Earth to the Moon".

Figure 7. Definition of Change States for the system "Coffee Machine".

The information presented in The General Concept (figure 5) is not sufficient yet to simulate in OPN. This decision-support tool requires parameters that will be modified by the functions and later evaluated. The next model presents these necessary parameters. They are linked to their respective processes through dashed lines. These parameters are related to a process through functions that describe it.

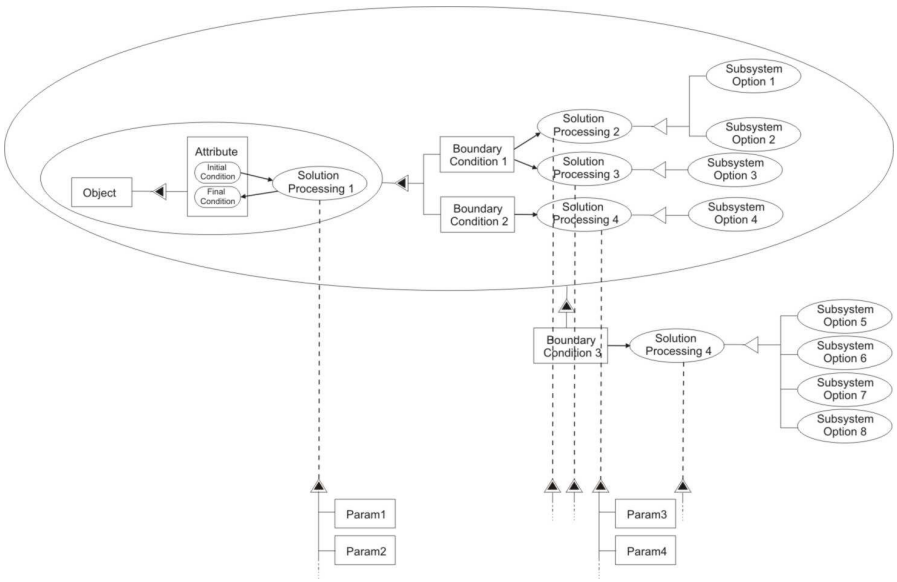

Figure 8. OPM model for The General Concept with parameters to be simulated.

Now that we have options and parameters (related to the options via functions), an OPN model can be created. Based on minimum requirements, the simulation process will eventually point out feasible architectures (figure 9). These requirements could be risk, time, cost etc.

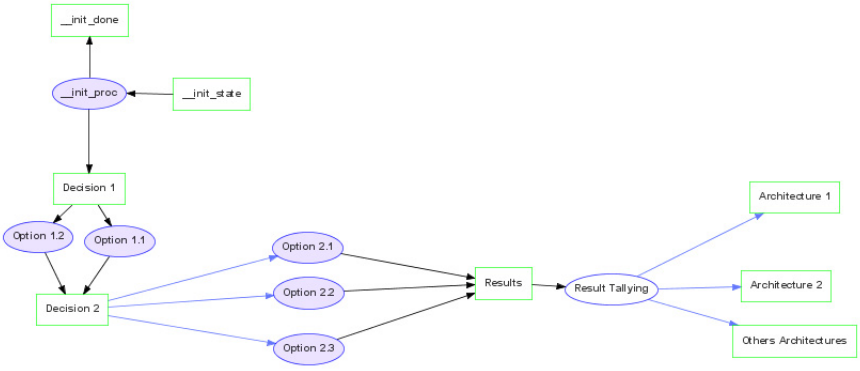

Figure 9. OPN model to be simulated.

The results of the simulation process let us derive a solution for the system to be built (judged to be amongst the best ones).

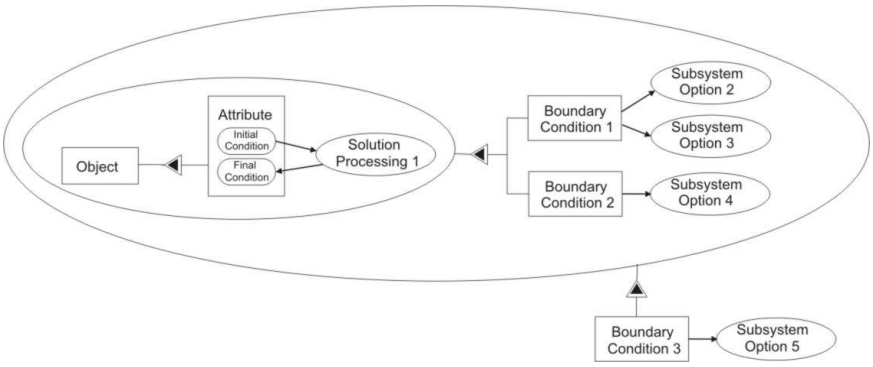

Figure 10. OPM model derived from the simulation process.

4 Study Case - Market of Sodas

The concept behind the suggested methodology was minutely described in the last topic, but no concrete application for a complete system has been presented yet. One could still argue about the unfeasibility of the method to a complex system. "The Market of Sodas Example" is a proof that we can actually develop new systems or improve existing ones adopting this approach. The example to be here presented focuses on logistics issues for the distribution of soda. We will look for best solutions for the problem "Delivering a Soda for a Customer".

The OPM Model in Figure 11 shows the definition of the states and conditions for a market of sodas. Note how the core of the problem (change of states) is attached to boundary conditions such as *Distance*. Following these "restrictions" requires subsystems that "solve" them. This is the first iteration process. The model keeps iterating until "minimum subsystems" are modelled such as *Car*, *Train*, *Airplane*, etc. In the next page, we present the model with parameters to be evaluated (figure 12). New notation has been added to the original set of symbols in OPM. The dashed lines in figure 11 for example, represent the relationships between the subsystems in the actual system, letting us derive an OPM model for the system given the results from OPN (thus, it is essential having all these links signalled).

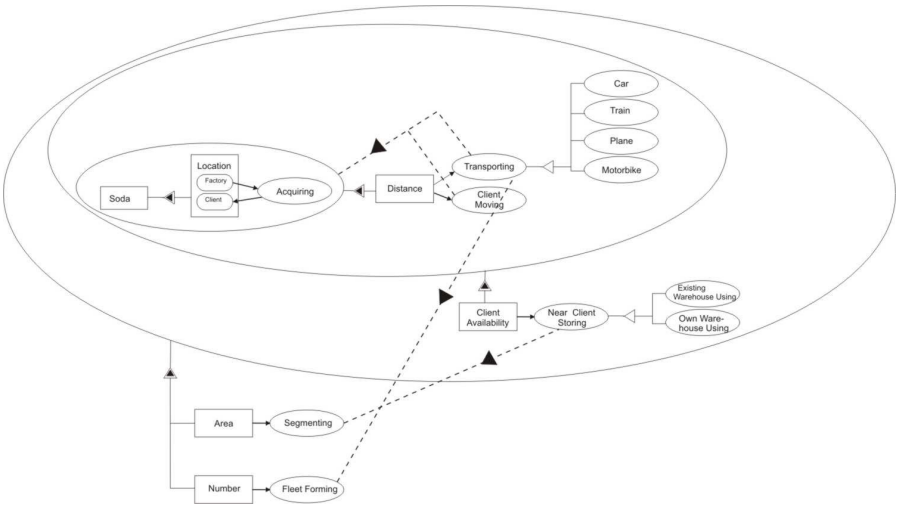

Figure 11. OPM model for logistics of a soda market.

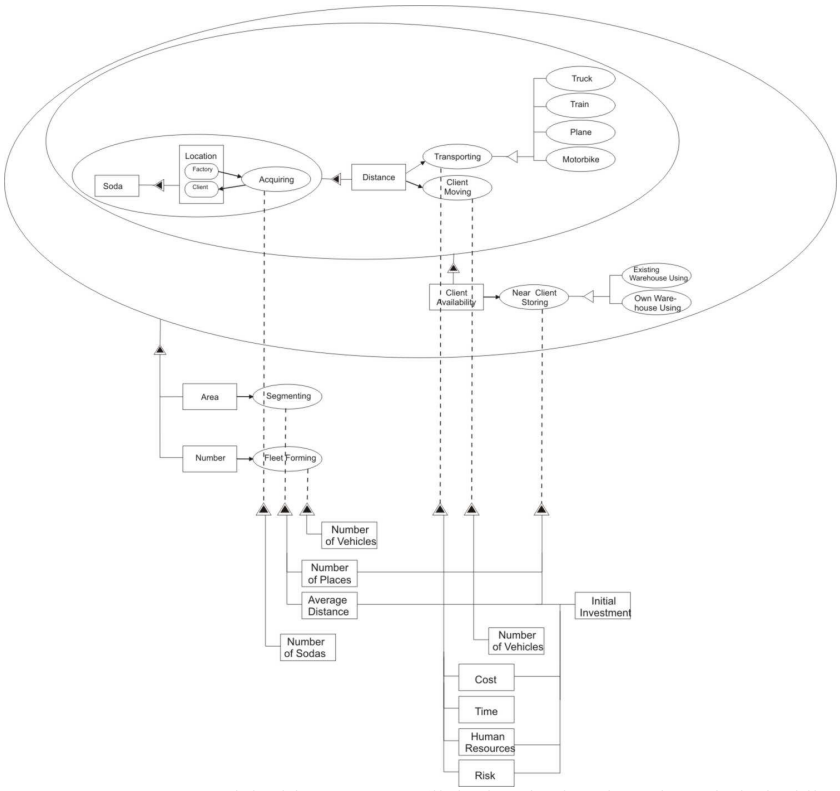

Figure 12. OPM model with parameters (linked to the functions through dashed lines).

Based on the information obtained in the first process, we can now build an OPN model that can enumerate solutions for our problems based on constraints we have defined (Figure 13).

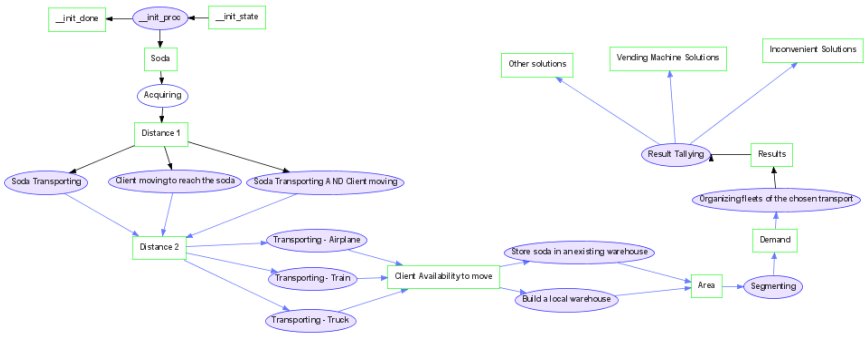

Figure 13. OPN model generated

As stated before, OPN stands as a decision-support tool. Thus, its reasoning features will lead us to a variety of models of the desired system. For our example, after simulating the options of architecture, we can now build instances of systems using OPM. Given a series of decisions made through the simulation process, we can now model them in OPM (figure 14). In figure 8, that would be replacing all the "Solution Processing" processes by the selected "Subsystem Option".

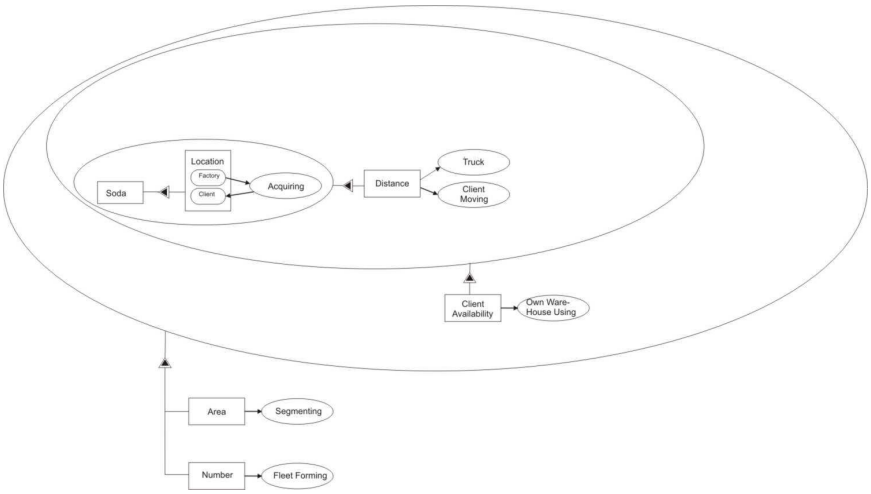

Figure 14. Modeling the decisions using OPM.

The last step of the composition of this system is, based on decisions made, to model the actual system. This can be systematically done erasing all the boundary conditions, their links and not selected options and turning the dashed lines in figure 12 into continuous lines.

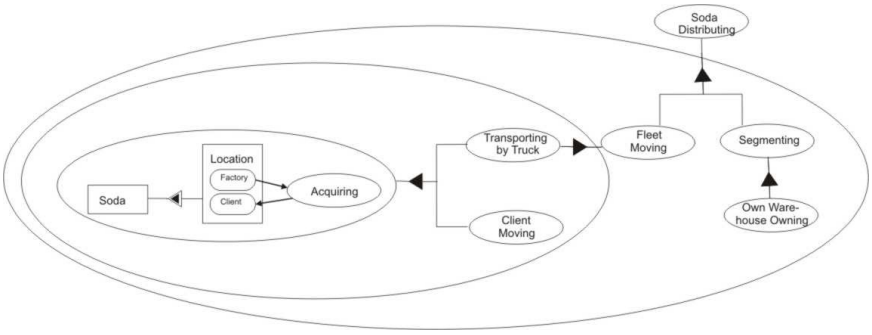

Figure 15. OPM model for the chosen solution/architecture.

5 Further Development

As a next step towards the final goal of mechanizing the conception of systems, further research need to be done. Some questions remain unanswered. How to decide to which extent we should model in order not to spend great effort modeling a solution that will never be developed? When composing and architecture model using different notations for various parts of the model, how to make the overall model make sense? How to integrate different models if they are written with modelling notations at the convenience of the system architect?

6 Conclusions

In this study we presented an innovative approach for complex systems development. The applicability of this new methodology that can automatically generate architectures for a system was shown through the study of the logistics in a soda market. Besides the prospect of mechanizing the conception of systems, this approach would allow us to identify new solutions traditionally discarded by past-experienced based constraints. For an instance, The Apollo Program was constrained in the 60s by time and risk. The solution then was going via Lunar Orbit. New space exploration vision systems are very much constrained by cost. Most cost-effective solutions may point out to a flight directly to Lunar Ground, for example. Certainly, changing initial constraints will change the space of solution options. Using such methodology means that we will not have to start all the decision process over again if constraints change. Further studies are expected in order to answer the current problems with this new approach.

7 References

7.1 Book Titles

[1] D. Dori, Object-Process Methodology: A Holistic Systems Paradigm: Springer-Verlag, 2002.
[2] Yourdon, Edward. Modern Stuctured Analysis. Prentice Hall, 1989.
[3] Derek J. Hatley, Imitiaz A. Pirbhai. Strategies for Real-Time System Specification. Dorset House, 1988.
[4] G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language User Guide. Addison-Wesley Professional, 2005.

7.1 Thesis

[5] B. H. Y. Koo. A Meta-language for Systems Architecting. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2005.

7.2 Papers

[6] W. L. Simmons, B. H. Y. Koo, and E. F. Crawley. Architecture generation for Moon-Mars exploration using an executable meta-language. In Proceedings of AIAA Space 2005, 30 August - 1 September, Long Beach, CA, 2005.
[7] W. L. Simmons, B. H. Y. Koo, and E. F. Crawley. Space systems architecting using meta-languages. In 56th International Astronautical Congress, 2005.
[8] W. L. Simmons, B. H. Y. Koo, and E. F. Crawley. A Computational Method for Mapping The Decision Space of the Lunar Exploration Program. In 57th International Astronautical Congress, 2006.
[9] D. Dori, I. Reinhartz-Berger, and A. Sturm, "Developing Complex Systems with Object-Process Methodology with OPCAT", presented at Conceptual Modeling - ER 2003, 2003.

7.3 Internet References

[10] http://opn.mit.edu/index.php/OPN_Quickstart. Accessed on: Sept. 1st 2006.
[11] http://www.opcat.com. Accessed on: Sept. 1st 2006.
[12] http://www.omgsysml.org