Copyright © 2005 by ABCM

SPECIFICATION AND ANALYSIS FOR AUTOMATED FLEXIBLE MANUFACTURING

José Reinaldo Silva
Mechatronics Engineering Department
Escola Politécnica
Universidade de São Paulo, Brazil
reinaldo@poli.usp.br

Eston A. Santos
Mechatronics Engineering Department
Escola Politécnica
Universidade de São Paulo, Brazil
eston@usp.br

Tiago Stegun Vaquero
Mechatronics Engineering Department
Escola Politécnica
Universidade de São Paulo, Brazil
tiago.vaquero@poli.usp.br

Abstract. *The great effort to achieve effective automated manufacturing systems that could also be integrated and flexible lead to the need of a systematic process of shop floor and manufacturing process design with a complete life cycle, that is, where the initial design phases were not neglected. Frequently, the phases concerned with requirement elicitation and analysis were not considered when a technical based product system or even the manufacturing environment is concerned and the process start with a problem definition and goes directly to the design phase. In this work we try to show that an incomplete life cycle is not an answer, neither for the complexity of designing an integrated and flexible manufacturing system nor to face the informality of the first steps of the life cycle of a net production system. On the contrary, the superposition of project phases, as defined by Unified Process practically condemns the incomplete life cycle to finish with incomplete target systems. We propose a way to treat the initial phases of the process, particularly in what concerns production systems, to be represented by XML (as an internal language in the design environment) and to capture and analyses requirements by translating Use Cases, or any of the UML diagrams, to XML and after that to an Extended Petri Net approach called GHENeSys, where it could be properly analyzed based on the desired properties of the net system that results. A specific software tool was produced to handle the process of translation from UML representation to classical or extended Petri Nets approach such as GHENeSys, called itSIMPLE.*

Keywords*: Requirement Engineering, Specifications, Integrated and Flexible Systems, Petri Net, UML, XML, Automation*

1. Introduction

There is a growing demand for disciplined methods of development in modern Systems Engineering, especially for automated systems, which includes not only the critical problem of managing the fusion of hardware and software but also the requirements for the software system.

Conceptual and practical schemas appear frequently in the literature, some of which are associated with software development techniques, or connected with specific areas of Engineering, or even with the knowledge acquired in the development of specific artifacts. However, in spite of the consensus that the early phases of the development process are the most important to assure that the target system works properly, still, most of the tools to support system development address the design and implementation phases.

The present paper syntheses the recent developments made by the authors and other collaborators in studying the engineering design process, particularly in the early phases of the process, what leads to the Requirement Engineering process as well as to the generation of sound specifications.

In the initial phase of our work we focus on the process of integrating generic viewpoints, that is, meaning that we search for a process that does not depend on the type of the artifact that is being designed. The proposal for that was to translate Use Cases to classical Condition/Event Petri Nets, through a process that starts with a transforming normal sequential Use Cases into a Tagged Use Case representation. That tagged version reinforces sequential flow, deviations, branches and conditions and could be immediately translated to a C/E Net (Silva and Santos, 2003). Now we adopt the thesis that a good engineering design process could be a metaphor for a successive translation of representations since a semi-formal one at the beginning up to a formal design model, which could finally be implemented.

Following this "interpretation between (sequent) theories" approach, would necessarily raised a problem which is how to transpose the early phase of design, which is naturally informal, to come up with formal specifications, just extracted from the analyzed requirements. Silva and Santos (2003) propose a cycle of refinements starting with elicited requirements in at least three different viewpoints (the sponsor, the final user and the developer). The vertical integration, between shop floor and the business management support the need for some consideration with what was called *stakeholder*, or substitute it for a more complex net of interests and objectives for products and systems. The worldwide supply chain approach of EPC-Global is just an example of that.

Thus, for the early phase we need to start with an informal or semi-formal representation, and UML composed of Use Cases and all its diagrams is a good candidate. However we still have to provide the requirements in a representation sound that could be used to analyze the target system and that also could lead to formal specifications. So, what we really need is a complete environment that could handle all the necessary transformation from informal to formal specification. Such environment must have as input informal data associated to characteristics and properties embedded in formal rationalities from the target system. These characteristics represent the user requirement of the proper artifact, indeed. This can be viewed as a meta-planner.

In order to make the proper data transportation and language semantic transfer we need a meta-language (that could also be friendly to handle with). XML is being such a language, once it is the natural substitute for EDI (Electronic Data Interchange) for one side (the data transfer side), and is also meta-language, in the sense that could be used to talk about other language. That later property allows XML to capture the semantic of a statement without being committed with the syntax of the representation language and by its dialect extensions, to make it possible to transfer that semantics to other language representation (if it is consistent).

We present an environment where requirements are elicited and represented using Use Case and UML diagrams, that could be enough to proceed to the analysis of the requirements, if the target system (or product) is no complex. However if the target system is complex, integrated and flexible a huge number of dependencies would come up from those initial requirements and a translation to a more formal language is necessary. That could be achieved by first translating the UML representation to XML and then, with the addition of the proper dialect definition (in XML) we could get to the target representation. If the problem is only planning, as in robot planning, we could make the transfer from UML to XML and from XML to PDDL (Planning Domain Definition Language). If the problem is modeling production processes we should transfer the representation to Petri Nets or one of its extensions, in order to analyze the requirements.

This proposed environment is called itSIMPLE (Integrated Tools Software Interface for Modeling Planning Environments) (Vaquero et al., 2005), where XML is used as an intermediary language in the process of transforming input requirements in its proper language either to engineering analyzes or to compose a formal specification. In order to present the itSIMPLE proposal we will start with a brief presentation of the principal features of XML and its growing use to engineering systems. Following, we present the problem of viewpoints and its integration need to have a good start for a system project. In the next section we will present the representation translation, first from UML to XML and after the translation from XML to GHENeSys using an adapted dialect from the PNML (the dialect of XML that represents Petri Nets). A short example for automation is shown with the problem of the intelligent lift.

2. The XML representation

XML was created by the WWW Consortium in 1996, and had its first version available in 1998. It is part of the effort to create a meta-language capable to add semantics to its preceding markup language, the HTML. It also allows the exchange of data and the integration of different data types, appearing as a good substitute for the current EDI system. In fact the language was derived from the Standard Generalized Markup Language (SGML) to improve flexibility to exchange of transfer documents over the Internet.

The principal element in XML is the possibility to define DTD's, that is, Document Type Definitions, which can be linked not only to the format of the target document as with its intended semantics. Thus, XML can define notations at the same time that express some data that could be represented in that same notation. This feature was explored in the current version we have for the problem taking a structured transference between informal requirements and more formal semantics.

Another attractive aspect of the problem is its spread use in manufacturing, principally to problems associated with production planning and scheduling. The current pressure to have the manufacturing process monitored by modern Information Systems lead the PSLX Consortium to redefine the APS (Advanced Planning and Scheduling) in a new specification of 2004, where XML had an important whole. More than the problem nature, the motivation here is the possibility to define important documents such as the BOM (Bill of Materials) or the Process Sheet.

The proposal for the use of XML in this work is more ambitious and more concerned with its capability as meta-language. Here, the key point is to provide an information modeler (DeRose, 2000), able to integrate different representations as depicted in the next section, while keeping the semantic of the target artifact or system being

Copyright © 2005 by ABCM

described. Such a situation is pretty common during the process of transference from the informal requirement elicitation to the more formal specification phase.

A similar problem has been pointed by (Qiao et al., 2003) in the semantics transference between representations of a product or systems related to MCM (Mass Customized Manufacturing). In MCM, the challenge is to produce a representation that enclose and manage information (more than data) related to the design and the simulation of the target artifact. Again, the idea is to bring what is normally represented in different documents and in different "languages" to the same representation, similarly to the problem of representing the same artifact with different documents and diagrams, as we have in the requirements of production systems.

3. Requirements Engineering Verification based on Viewpoints

As Kotonya and Sommerville (1996) states, "requirements reflect the needs of customers and users of a system. They should include a justification for this system, what the system is intended to accomplish, and what design constraints are to be observed". This work aim to verify if the user's requirements of an engineering system are accordingly those constraints, based on the different viewpoints of the users.

An effective requirements engineering method must have (Kotonya and Sommerville, 1996):

- The precision of definition of its notation
- Suitability for agreement with the end-user
- Assistance with formulating requirements. This can be viewed in terms of two aspects:
  - How the notation organizes the requirements knowledge structure for the system.
  - How the notation affords the separation of concerns.
- Definition of the system's environment
- Scope for evolution.
- Scope for integration. This can be viewed in terms of requirements approaches and types of requirements:
  - There is no single requirements approach that can adequately articulate all the requirements of a system both from the developers' and users' viewpoints.
  - Non-functional requirements tend to be related to one or more functional requirements.
- Scope for communication.
- Tool support.

The previous work of Silva and Santos (2003) proposed the verification of requirements based in Use Case description and after this, validate the modeling with the involved *stakeholders* with a future tool support.

The differences between stakeholders' viewpoints must be considered by the modelers by using different approaches in elicitation tasks. For example, the sponsor thinks in your business process strategies, while the final user priorities could be in the daily tasks.

The strategy now relay in the proposal that engineering design process could be a metaphor for a successive translation of representations of the system being modeled. The different viewpoints could be represented by different UML diagrams (OMG, 2001). Table 1 shows the proposed relation between the stakeholders' viewpoints and the best UML diagram to represents it.

Table 1. Viewpoints x UML Diagrams.

Stakeholder Viewpoint	UML Diagrams
Sponsor	• Use Case • Activity Diagram
Final User	• Use Case • Activity Diagram • Sequence Diagram
Developer	• Use Case • Activity Diagram • Sequence Diagram • State Chart Diagram • Class Diagram • Deployment Diagram • Component Diagram • Collaboration Diagram

Thus, the requirement elicitation would come out with a set of diagrams, each one with the viewpoint belonging to one of the classes of users in the left column of Table 1. Naturally, all these diagrams are supposed to describe

properties of the same artifact and its contents are either the same or complementary (but not contradicting) to the meaning in the others. In (Silva and Santos, 2003) only Use Cases were considered, once they are the most commonly used. However, in practice, that is not possible to use only Use Cases to elicitation requirements.

The normal elicitation process does not predict a check of consistence among the different diagrams, to see if they compose a "sound" requirement set. Therefore, if we just transfer all the diagrams to Petri Nets to validation we can get a strange set of graphs, which become very difficult to analyze. Once they are not sound Petri Net schemas, a verification of isomorphism is not really possible, what would be equivalent to checking of consistence.

It is clear that the integration of requirements should be started before the translation of the whole set of requirement diagrams to Petri Nets, and that crashing incompatibilities should be solved in that phase, where it is easier a negotiation or even clear understanding of the user or stakeholder needs. If the integration of the requirements is done when transferring the information to Petri Nets, that also means crossing the board to the development viewpoint and taking the chance of neglecting important information.

Based on some experiments in the process of elicitation and documentation we decide for an early integration of the requirement viewpoints using UML as a basis for the new representation. The meta-language capability of XML fits perfectly with integration needs, and also has an additional advantage which is the possibility to have its contents transferred to any other schema or specification language. For instance, for some planning problems (which also need complete life cycle), the praxis is to describe problems and domains using a special language called PDDL (Backus, 2003) (Boddy, 2003). In that case PDDL is not an executable language and does not have many resources to verify the requirements. Another possibility would be to transfer to more sound specification languages like B (Abrial, 1996) or even its executable derivation.

Table 2 shows a classification of several of the most used languages used to formal specification today and some properties considered important, such as the possibility to express concurrency, the executability, that is, the possibility of being used to verification by running it contents, as well as the use of variables and the possibility to represent non-determinism (Frappier, 2000). Notice that Petri Nets, as well as Object-oriented Petri Nets are executable (meaning that to apply a token player to it) and so is B.

Table 2. Properties of several of most use language for formal specification detaching executability (Frappier, 2000).

Method name	Concurrency	Executability	Usage of variables	Non-determinism
Action Systems	NO	YES	YES	YES
B	NO	YES	YES	YES
CASL	NO	YES	YES	NO
Cleanroom & JSD	NO	YES	YES	YES
COQ	NO	YES	YES	YES
Estelle	YES	YES	YES	NO
LOTOS	YES	YES	YES	YES
OMT & B	NO	YES	YES	YES
Petri Nets	NO	YES	NO	YES
Petri Nets with Objects	YES	YES	YES	YES
SART	YES	NO	NO	YES
SAZ	NO	YES	YES	YES
SCCS	YES	YES	YES	YES
SDL	YES	YES	NO	YES
UML	YES	NO	NO	NO
VHDL	YES	YES	YES	NO
Z	NO	YES	YES	YES

That is very important, for the task of verifying requirements, the basis of the supporting tool called itSIMPLE (Vaquero et al. 2005). For this reason, the next section presents the itSIMPLE, as part of the engineering design framework, especially to validate requirements and specifications.

4. An Integrated Tool for Modeling Planning Environment

The UML is a powerful modeling language that can be used to increase the expressiveness of the planning domain and problem models. It has also another important feature: It can easily export to XML language.

From XML, the integrated tool can export information to Petri Nets and to PDDL for backward compatibility. The Petri Nets are used to analyze the planning domain about its dynamic feature and life cycle. Petri Net has formal model to provide necessary tools and techniques for dynamic evaluation analyses and also for validation of a domain. In addition, some patterns can be extracted from the Petri Nets of planning domain models that can guide a planner to choose one planning technique over another for a specific domain. The concept of the tool is shown in Figure 3.

The XML has specific tags that encapsulate all features of the UML model. Class, operator, association, state, object are examples of tags in the XML files.

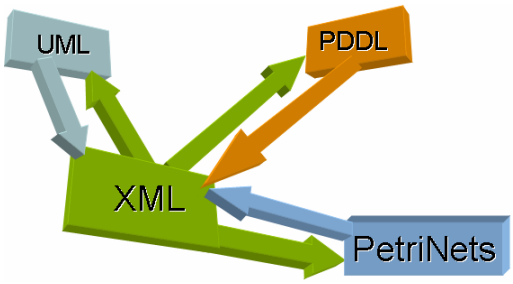

Figure 3. The integrated modeling tool

The exportation to Petri Nets depends on the structure of the XML file. In (Bray *et al.*, 2004), it's possible to read an XML specification and create a Petri Nets graph.

4.1. The itSIMPLE Tool

In order to model and handle all these tools, a software interface has been developing. Figure 4 shows a screenshot of the software in construction. It calls itSIMPLE (Integrated Tools Software Interface for Modeling Planning Environments). This software allows the user to create a UML model of a planning domain by defining the classes, subclasses and objects of the planner, agent or environment classes.

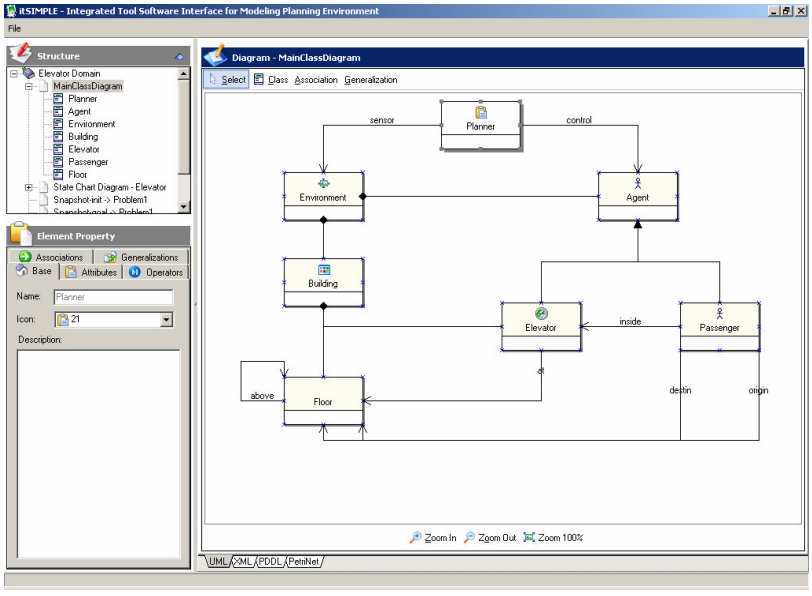

Figure 4. itSIMPLE with the class diagram of the classical elevator problem

The software saves the UML model in a XML file and permits the user navigates in the UML model, XML file or PDDL description freely. A model in Petri Nets could be generated from the Use Case representation and diagrams following the work of Silva and Santos (2004). In the present case, XML is taken as an intermediary language, making the same description available also to be translated to PDDL or other suitable representation.

As an example let us take the problem of an automated lift (the intelligent lift) and its partial definition using even Class Diagram or an Use Case Diagram, as shown in Figure 5 and Figure 6.

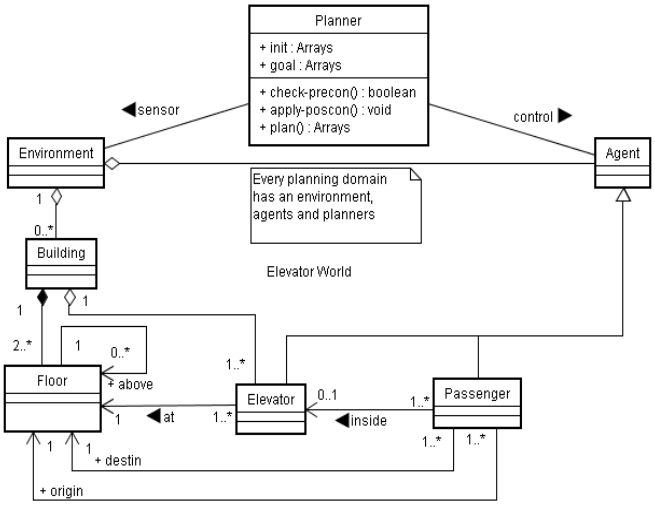

Figure 5. UML Class Diagram for the intelligent elevator problem

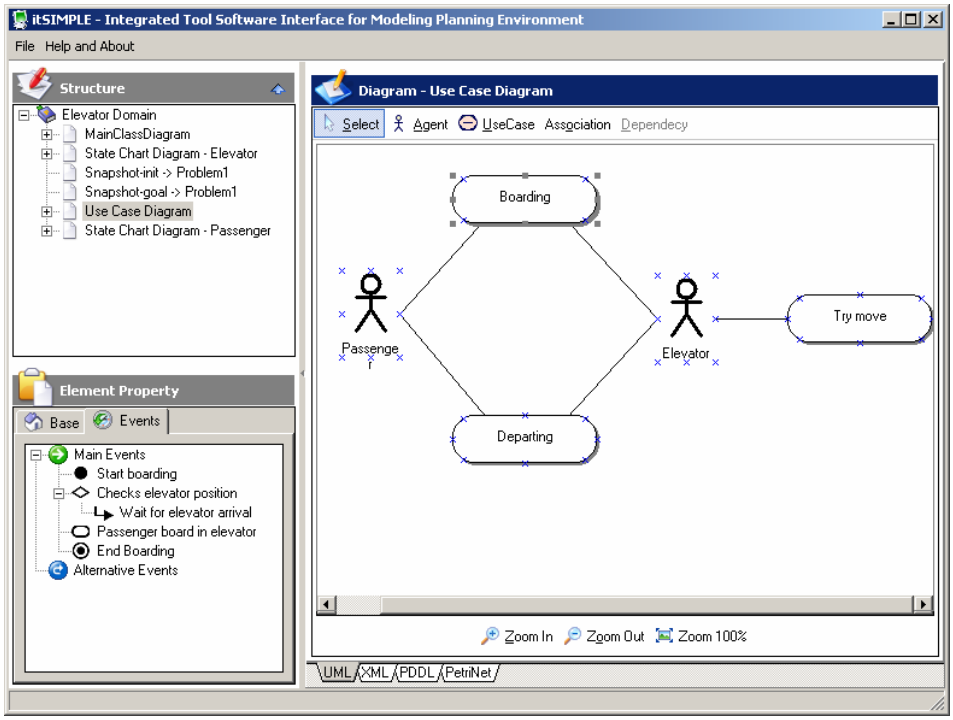

Figure 6. Use Case Diagram for the elevator problem using the itSIMPLE tool.

Using itSIMPLE it is possible to translate these requirements into a XML representation (Figure 7) that could be used as a basis to a further translation in a Petri Net. Here the output of itSIMPLE, besides PDDL, is a special document with the appropriated format to be held by another software application called GHENeSys (General Hierarchical Enhanced Net System), which is an object-oriented net extension (DelFoyo and Silva, 2003) as shown in Figure 8. The choice of an object-oriented net could be justify by Table 2 (Frappier 2000), where it could be seen that among the most used specification languages that admits the use of variables, what can make the early phase of design process a lot easier.

5. Conclusions and further work

We presented a plan (a set of actions) to systematically transform informal requirements into sound and formalized specification, to a general set of projects, including (principally) the specification of integrated and flexible systems. As expected, it could not be an automated, totally computable or formal process, but a systematic approach that could be performed with the aid of computers, using software as itSIMPLE, developed previously to support the life cycle of (artificial intelligence) planning systems.

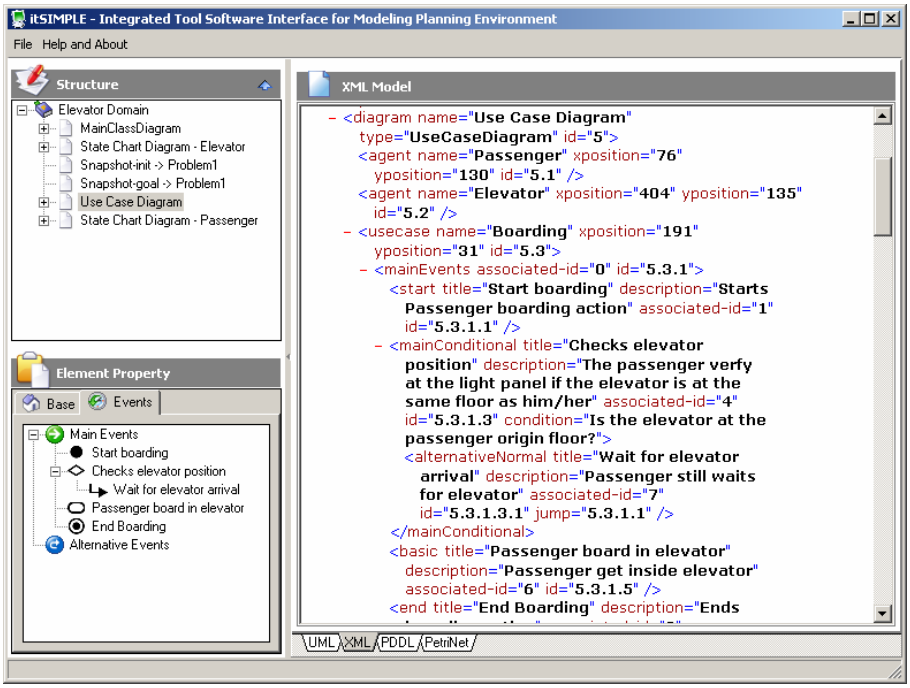

Figure 7. Translation of the Use Case of Figure 6. in XML

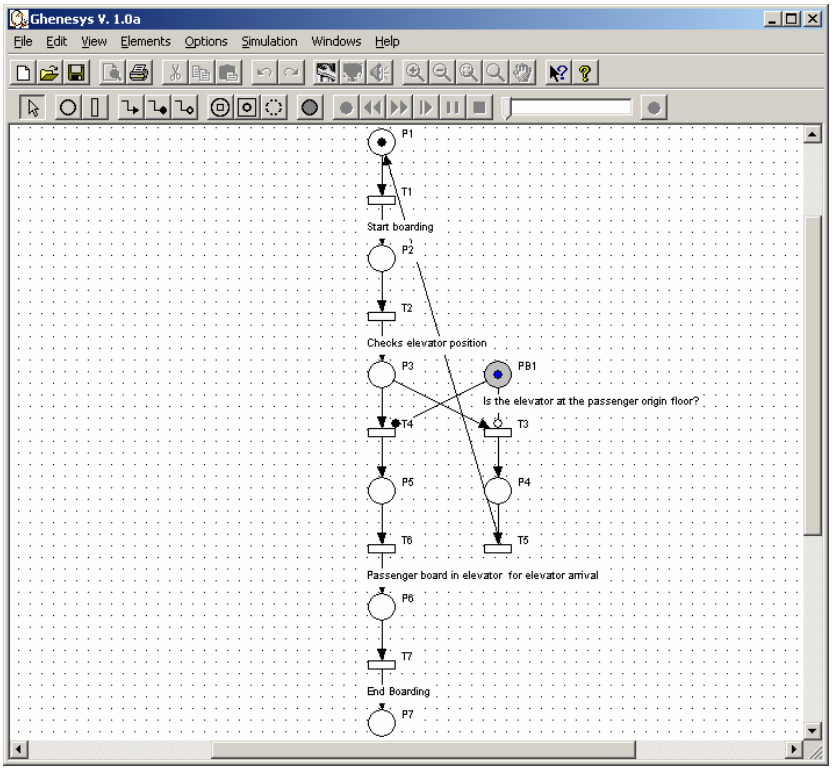

Figure 8. Final translation to an extended object-oriented Petri Net, running in the GHENeSys application.

The integration of requirements before translating that in a more formal specification schema such as Petri Nets has been highlighted and XML was proposed as a suitable intermediary language by its properties, principally as meta-language. However, a computational process that takes care of consistency verification among all UML diagrams where not envisaged, since UML is a semi-formal language and there is not a sound way to treat that verification besides appealing to heuristic methods based on Artificial Intelligence. That will be left for a further work.

Also, it is possible to do the same process of transforming the requirement specification into Petri Nets of even PDDL (in the case of a pure planning problem) using other executable language. We mention in the paper the possibility of using B, a formal language to specification that is growing and spreading its use in the aeronautic industry. That will be also a further work.

6. Acknowledgements

We thank to CNPq for partially financing with scholarships two of the authors of this work.

7. References

Abrial, J.R., 1996, "The B Book: Assigning Programs to Meaning", Cambridge University Press, UK.

Bachhus, F., 2003, "The Power of Modeling – a Response to PDDL 2.1", Journal of Artificial Intelligence Research, v. 20, pp. 125-132.

Balzer, R. and Goldman, A., 1979, "Principles of Good Software Specification and their Implications for Specification Languages", Proc. Specifications of Reliable Software Conf., Cambridge, Mass., pp. 58-67.

Boddy, M., 2003, "Imperfect Match: PDDL 2.1 and Real Applications", Journal of Artificial Intelligence Research, v. 20, pp. 133-137.

Bray, T., Paoli, J., McQueen, C.M., Maler, E., Yergeau, F., 2004, "Extensible Markup Language (XML) 1.0 – Third Edition", http://www.w3.org/TR/REC-xml/.

D'Souza, F.D. and Wills, A.C., 1999, "Object, Components, and Frameworks with UML – The Catalysis Approach", Addison-Wesley. United States of America and Canada.

DeRose, S. J., 1999, "XML Linking". ACM Computing Surveys, vol. 31, no. 4es, December.

DelFoyo, P.M.G. and Silva, J.R., 2003, "Towards a Unified View of Petri Nets and Object Oriented Modeling, Proceedings of COBEM 2003, ABCM, São Paulo.

Edelkamp, S. and Hoffman J., 2004, "PDDL 2.2: The Language for Classical Part of the 4th International Planning Competition", Technical Report, Fachbereich Informatik and Institut für Informatik. Germany.

Frappier, M., Harbrias, H., 2000, A Comparison of the Specification Methods, Univ. de Sherbrook, Quebéc, Ca., http://www.dmi.usherb.ca/~spec/A3-comparison-v4.pdf, consulted in March, 10, 2005.

Green, T.R.G., 1989, "Cognitive Dimensions of Notations", In A. Sutcliffe and L. Macaulay (Eds.) People and Computers V. Cambridge, UK: Cambridge University Press, pp. 443-460.

Jacobson, I., Booch, G. and Rumbaugh, J., 1999, "The Unified Software Development Process". Addison-Wesley.

Kiritsis, D., Xirouchakis, P. and Gunther, C., 1998, "Petri Net Representation for the Process Specification Language - Part 1: Manufacture Process Planning", CAD CAM Laboratory, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland.

Kotonya, G., Sommerville, I., 1996, "Requirements engineering with viewpoints", Software Engineering Journal, v.11, n.1, p.5-18.

Murata, T., 1989, "Petri Nets: Properties, Analysis and Applications", In Proceedings of IEEE, v. 77, n. 4, pp. 541-580.

OMG, 2001, "Unified modeling language specification: version 1.4", http://www.omg.org/uml.

Qiao, G., Riddick, F., McLean, C., 2003, "Data Driven Design and Simulation System Based in XML". Proceedings of the 2003 Winter Simulation Conference, National Institute of Standards and Technology. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

Silva, J. R. and Santos, E. A., 2003, "Viewpoint Requirement Validation Based on Petri Nets", In 17th International Congress of Mechanical Engineering, Sao Paulo.

Silva, J. R. and Santos, E. A., 2004, "Applying Petri nets to requirements validation" In: IFAC Symposium on Information Control Problems in Manufacturing, INCOM'04, Salvador, p. 1-8.

Vaquero, T., Tonidandel, F., Silva, J.R., 2005, "The itSIMPLE Tool for Modeling Planning Domains". Proceeding of ICAPS (International Conference on Advanced Planning Systems), AAAI, Monterey, CA.

8. Responsibility notice

The authors are the only responsible for the printed material included in this paper.