INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Extraction of Knowledge from Petri Nets Models in Service-oriented based Industrial Automation Systems

Joel Alexandre Pereira Alves

Relatório de Estágio para Obtenção do grau de Mestre em

Engenharia Industrial

Ramo de Eng. Mecânica

Orientador: Paulo Leitão

Supervisor Empresa: Armando Walter Colombo (Schneider Electric)

November, 2009

Abstract

In last decades significant changes have been noticed in manufacturing environment, moving from a local economy to a global economy and generalized concurrency. Markets expectations for products with high quality at lower costs, highly customized and with short life cycles, lead us to the need to implement complete different strategies to satisfy these requirements.

With the current scenario, technologies have a strong weight in the search of a solution for new manufacturing systems. Service oriented Architectures (SoA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains, which combined with Petri Nets (PN) capabilities of modeling, analysis and simulation and using Web services as communication platform, is one suitable proposal to solve the manufacturing problem and its limitations, granting interoperability, adaptability, flexibility and integration of all the devices that belong to the manufacturing systems.

The main objective of this work is to develop mechanisms for the extraction of knowledge from the Petri nets models to support the execution of important control functions, such as monitoring and decision-making. Other objectives can be referred, like the modeling and the assessment of the Petri net models and the creation and improvement of some tools already existent in the area. This work was done during a stage at the Industrial Automation HUB of the Schneider Electric company in Seligenstadt, Germany.

All these studied and developed mechanisms were applied to an experimental case study, which allowed us to perform the assessment and the validation of the designed Petri nets models.

Resumo

Nas últimas décadas tem-se assistido a mudanças significativas nos ambientes de fabrico, passando de uma economia local para uma economia global de concorrência generalizada. Nos mercados actuais, a expectativa por produtos de alta qualidade a baixos preços, altamente customizados e com ciclos de vida curtos, conduzem-nos para a necessidade de implementar novas e diferentes estratégias para satisfazer estes requisitos.

Neste cenário, as tecnologias têm um grande peso na procura de uma solução para os novos sistemas de fabrico. O Service oriented Architectures (SOA) é um paradigma para organizar e utilizar as suas capacidades de controlo distribuído para ligar diferentes domínios, que combinando com as redes Petri (PN) e aproveitando as capacidades de modelação, análise e simulação, e utilizando os Web Services como plataforma de comunicação, é uma proposta adequada para resolver os problemas dos sistemas de fabrico e suas limitações, garantindo interoperabilidade, adaptabilidade, flexibilidade e integração.

O objectivo principal deste trabalho foi o desenvolvimento de mecanismos de extracção de conhecimento dos modelos criados usando o formalismo das Redes de Petri, para servir de apoio à execução das importantes funções de controlo, tais como a monitorização e a tomada de decisão. Outros objectivos que podem ser explícitos são a modelação e avaliação dos modelos em redes de Petri e a criação e aperfeiçoamento de algumas ferramentas já existentes na área. Este trabalho foi realizado durante um estágio, na unidade de Industrial Automation HUB da empresa Schneider Electric em Seligenstadt, Alemanha.

Todos os mecanismos estudados e desenvolvidos durante o período de estágio foram aplicados a um caso de estudo experimental, o que nos permitiu fazer a sua avaliação e validação.

Acknowledgements

I must first thank my coordinator, Paulo Leitão, for his apparently infinite knowledge, encouragement and patience. This work would not be here today without his guidance and support. His knowledge and insight contributed immensely to the quality of this report.

I wish to thank Armando Walter Colombo from Schneider Electric for accepting this internship and for his support throughout the 6 months that I was under his care and supervision. And also thank him, for believing in my work and giving me the chance to work on this project.

Special thanks are also due to Marco Mendes, my friend. His insight and knowledge was poured into this work through our many discussions on the problems we were facing and the objectives we were trying to accomplish.

Further thanks should be extended to Nataliya Popova who, despite having incredibly busy work schedule, always managed to find time to oversee the progress of the work and provide her support.

I also wish to express my gratitude towards Axel Bepperling, whose technical expertise was matched only by his generosity in sharing both his time and experience, thus motivating me even when it seemed that everything was failing to work properly.

I also want to thank my friends and colleagues, especially Daniel, Nuno, Dorian, and Viktor, in Germany who became a kind of foster family in a strange country, and made Seligenstadt a home away from home.

Finally, I want to thank my family and particularly my girlfriend Joana, for their encouragement and support, and thank my friends here in Portugal, and worldwide, for all the same reasons.

Acronyms

IPB	Polytechnic Institute of Bragança
ECTS	European Credit Transfer and Accumulation System
FP6	European Union's 6th Framework Programme
SOCRADES	Service-oriented Cross-layer Infrastructure for Distributed Smart Embedded Systems
SoA	Service-oriented Architecture
PN	Petri Nets
MAS	Multi-Agent Systems
IT	Information Technologies
XML	eXtensible Markup Language
WSDL	Web Service Definition Language
ebXML	Electronic Business using eXtensible Markup Language
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
OE	Orchestration Engine
DAS	Dynamic Assembly System
RFID	Radio-Frequency IDentification
CDS	Continuum Development Studio
Mutex	Mutual Exclusion Objects
PNC	Petri Nets Composer

Table of Contents

Table of Figures

1. INTRODUCTION

This report was written in the context of the Final Project course of the Masters in Industrial Engineering degree, specialization of Mechanical Engineering, from the Polytechnic Institute of Bragança (IPB). The Final Project course is one year long and worth 42 ECTS (European Credit Transfer and Accumulation System), the firsts six months were focused in the preparation for the internship where the student received formation and bases for the second phase which started on 1st March and lasted exactly five months, until the 30th of July 2009.

The project was mainly done at the Industrial Automation HUB of the Schneider Electric company in Seligenstadt, Germany. Schneider Electric is a French global company founded in 1836. Initially, the company, named Groupe Schneider, focused in weapons and armaments, but nowadays the company completely changes its business in favor of the electrical and controls industry.

For the internship at Schneider Electric, the main initial direction and objectives for the work as well as supervision were made by Dr. Armando Colombo and Professor Paulo Leitão.

The internship activities were focused on modeling, extraction of information and knowledge of the Petri nets to support the monitoring and decision-making in service-oriented automation systems, being framed under the European Union's 6th Framework Programme (FP6) research project Service-oriented Cross-layer Infrastructure for Distributed Smart Embedded Systems (SOCRADES). Its primary objective is to develop a design, execution and management platform for the next-generation of industrial automation systems, exploiting the Service-Oriented Architecture (SoA) paradigm both at the device and at the application level. The SOCRADES consortium is made up of 15 partners from 6 European countries. It is led by Schneider Electric GmbH and includes the major European players in the industrial automation sector (see www.socrades.eu).

1.1. The Problem

Last decades significant changes have been noticed in manufacturing environment, moving from a local economy to a global economy and generalized concurrency. Markets expectations for products, with high quality at lower costs, highly customized and with short life cycles, lead us to the need to implement complete different strategies to satisfy these requirements.

In this worldwide market competition and crises shadow on its minds, companies try to respond to a maximum of potential clients' needs and look to every stimulus of the markets as a business opportunity. The most important factor in this kind of scenario is to give a response at the right time, in the right place granting satisfaction of the client needs and addressing the business opportunities. In this environment, where manufacturing systems have to respond to changes almost instantaneously, this factor could be the distinguishing point between a company with an immediate and short time success and a company with sustained growing and success. These changes, have forced companies to switch from a mass production assembly to a mass customization production. Competitiveness and business advantages between companies, lead them to fight for best strategies and best solutions for answering market expatiations.

The traditional manufacturing systems do not exhibit this capability of adaptation and evolution in terms of production control. In fact, the centralized and hierarchical control approaches present good production optimization but a weak response to changes, being antiquated, difficult and expensive to enhance, maintain and support, mainly because of rigidity and centralization of the control structure.

With the current scenario, technologies have a strong weight in the search of a solution that allows autonomy and intelligence capabilities, agile and fast adaptation to the environment changes, and more robust against the occurrence of disturbances, and easier integration of manufacturing resources and legacy systems.

Service oriented Architectures (SoA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains,

which combined with Petri Nets (PN) capabilities of modeling, analysis and simulation and using Web services as communication platform, is one suitable proposal to solve the manufacturing problem and its limitations, granting interoperability, adaptability, flexibility and integration of all the devices that belong to the manufacturing systems.

1.2. Objective and Contributions

The main objective of this work is to extract knowledge from Petri nets models to support the execution of important control functions, such as monitoring and decision-making. Other objectives can be referred like modeling and assessment of a case study, and creation and improvement of some tools already existent in the area.

Petri nets are a powerful formalism to model, validate and control complex discrete event-driven systems. Based on its powerful mathematical foundation it is suitable to formally and graphically represent the relations, properties and concepts presented in dynamic behavioural systems, as automation and production systems are, such as the concurrency, parallelism, resource sharing and mutual exclusion.

In industrial automation and production systems, especially those that are structured in a distributed and collaborative way, it is crucial to analyze and extract knowledge from the Petri nets –based behavioural models. In fact, the mathematical foundation allows the verification of several properties that will be useful to support distinct tasks, such as the decision-making, conflict resolution, behaviours forecasting and detection of available paths. These specifications can be analyzed and validated using qualitative and quantitative methods, based on algebra theory.

This work aims to study the available methodologies and the specification of new methods for the (automatic) extraction of information and knowledge from Petri nets –based control models. In this work, it will be used High-level Petri nets that are characterized by having additional information associated to their elements and by the step-wise refinement. A special attention is devoted to the service-oriented automation and production systems, which present specific requirements.

As this work is integrated in a European Project called SOCRADES (Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices), some of the proposed objectives and contributions intersect the SOCRADES project objectives, testing, applying and improving some project concepts and methods, and designing new mechanisms to be used in the developed system. Namely, it was contributed to develop the distributed control application for the experimental case study, through the modeling of the devices using Petri Nets, making the assessment of these models and composing these device models to achieve a large system model. In terms of the decision support system, it was used the information extracted from the Petri net models to provide a real-time decision-making based on manufacturing issues.

1.3. Document Organization

This document is divided into seven chapters, with this one being the first.

Chapter 2 introduces the various engineering concepts and technologies used throughout the work, and try to give an overview of each in order to make the work more easily understandable.

In Chapter 3, entitled “Modeling the FlexLink demonstrator using Petri nets”, the FlexLink Demonstrator will be modeled and the created models will be presented. Along the chapter the Petri nets models for each part of the system will be described and explanations about some options will be also presented.

In Chapter 4, entitled “Assessment of the Petri nets models”, the models of the FlexLink Demonstrator presented in chapter 3, will be analyzed and verified using the available Petri nets methodologies.

In Chapter 5, entitled “Composing Petri nets models”, the importance of composition will be emphasized and some examples of possible compositions will be presented. The synchronization and communication between models and engines will also be explained.

In Chapter 6, entitled “Decision support system using the knowledge extracted from Petri nets”, will introduce a decision-making mechanism that uses information extracted from Petri nets models; the analysis of the decision criteria used in this

mechanism is also described. The proposed mechanism is afterwards applied to the experimental case study.

The Chapter 7, entitled “Conclusions and future work”, rounds up the report with the conclusions and points out some future work.

The final pages are reserved for references.

2. STATE OF THE ART

Before the 20th century, the craft production was the dominant type of production, characterized by skilled workers that use general purpose tools to produce exactly what the customers asked for, being the production level close to one-at-a-time. Then, with industrial revolution, the manufacturing systems evolved to mass production assembly, where all the products are equal between themselves, attacking a huge market with a big number of potential costumers that has the same needs. This kind of manufacturing systems was characterized by a centralized and rigid hierarchy, which allowed the production of the same product in large scale using a rigid assembly line to produce a product composed by identical interchangeable parts. Figure 2-1 illustrates represents the result of a mass production assembly, with a high number of final products but without any differences between them.

(a) (b) (c)

Figure 2-1- Evolution of manufacturing systems a) craft production b) mass production c) flexible production

Currently, companies are faced with different requirements, due to the higher expectations of the markets, which are always changing, with customers having more demanding wishes and hoping for new products not only to satisfy their needs, but also innovative and creative new products which create new kinds of needs. The products are supposed to be specific and customized for the customer demands and with high quality at the lower prices, with short life cycles. Companies have to be prepared to react and to anticipate the movements and expectations of the markets

and they can only achieve this objective with flexibility, adaptability, reconfigurability and distributed intelligent manufacturing control systems.

Some companies already find strategies and production systems that allow them to respond to some of the new requirements. For instance, the well known car producer Smart, aims to provide to customers the possibility of customize completely its own car. The customer is able to create and customize his own car with his own specifications in few minutes. The Smart company (http://www.smart.com) gives the warranty that the car is built in short time and exactly with the asked features. This is only possible due its organization that consists in have all the suppliers' chain in a close area to the main production line, creating agreements where boot parts have a lot to win and if someone misses a term is penalized. Basically, Smart ensures that all suppliers have work to do and the suppliers ensure that the asked parts of the cars are available in the exact time and with the exact specifications. This strategy works perfectly, most of all because in the beginning Smart has thought as a city car and in the 90th passed to a fashion car inside the youth class and because of this factor company's look to this car as a good image to have its brands associated.

Figure 2-2 - Examples of variants of Smarts car models

However, this is not enough to address the current customer; although the thousands of variants for the same model, Smart only provides a short number of car models.

By now, most of companies are trying to find similar answers but with different strategies and trying to find a more consistent and embracing to actual requests in terms of manufacturing control and production systems.

Technological advantages are one of the preferred answers to current difficulties caused by evolution of markets and constant change of costumers needs. Actually, we live on technology season, where every day appears new devices and improvements on this area, so companies concentrate many efforts and money in research to find a perfect answer to actual difficulties.

For this purpose, several hardware and software technologies, methodologies and formalisms were lately introduced and applied in the manufacturing domain. In the scope of this work, Multi-agent systems (MAS), Service-oriented Architectures (SoA) and Petri nets assume significant relevance.

2.1. MultiAgent Systems and Service Oriented Architectures

The advances in networked information technologies (IT) verified in recent years are inducing significant socio-economic changes. Intelligent software entities are increasingly acting on our behalf automation processes, protecting sensitive data, filtering useful information, etc. The pervasive nature of network enable devices, its availability and reduced costs, are progressively transforming the internet, which mainly targeted people, into a hybrid mesh of autonomous devices that seamlessly interact with each other and with humans. Only recently have these ideas have the industrial domain, and in the future the survival of an enterprise depends on how well established is its IT infrastructure and how quickly can it accommodate changing requirements [1].

Decentralized production systems are considered organizational structures able to match necessary agility and efficiency necessary to compete in the market. One of the challenges faced by decentralized production systems is to ensure the coordination of heterogeneous decisions of multi-functional populated systems [2]. One of the problems that decentralized production and control tries to solve is the recovery from failures along the manufacturing control process, being the robustness of the systems is crucial. Considering that all part of the system are independent and able to perform decisions, the loss of one unity doesn’t mean that all system have to stop or be re-initialized. In fact, if one of the decision parts is missing, which means that one of the

components of the systems is for some reason stopped or failed, the collaboration between the existing components continues to be executed, finding new production solutions.

The MultiAgent Systems (MAS) and Service oriented Architectures (SoA) paradigms try to present a step further in manufacturing control systems, taking advantage of the newest mechatronics, information and communication technologies, to increase the modularity, flexibility and re-configurability of distributed automation systems. For this purpose, it is used the advantages of their implementation in terms of modeling, design and self-properties to maximize robustness and performance of distributed assembly systems.

2.1.1. MultiAgent Systems (MAS)

A MultiAgent System is a concept originated in the Distributed Artificial Intelligence area, [4] before present a definition for MAS, it is necessary to understand what an agent is. Despite the several definitions and interpretations for agents, it will be used an extension of the definition presented by Jennings and Wooldridge [4]:

*"An **Agent** is considered a software entity situated in a flexible production environment, with enough intelligence that is capable of autonomous control action in this environment and of co-operation relationship by participating in associations agreements with others entities in order to meet its design objectives"[5].*

An agent should be able to act without the direct intervention of humans or other agents, and should have a control over its own actions and internal state. A multi-agent based control system means one in which the key abstraction used for control components, e.g. controller, scheduler, is that of an agent [6].

According to these definitions, a MultiAgent system is a suitable approach to implement distributed production control systems, and can be characterized by decentralization and parallel execution of activities based on autonomous entities, the agents [7].

In the MAS paradigm, agents organize and interact between themselves, in order to achieve a final goal. When an agent has not enough capability to accomplish a task, it looks to another agents and tries to see who has the knowledge, the capability and the ability to help, to finish the work. In this approach one final result is a sum of a group of agents' contribution. To be able to interact, there is a major need of communication and knowledge of others agents skills.

Negotiation between agents requires local decision-making, the agents act upon reasoning on the state of the environment. They decide on which behavior to execute based on information gathered from other agents or on the state of surrounding environment. They are reactive to environment changes and in critical situations allow a quick response.

What are the characteristics of an agent on MultiAgent systems? It is a hard task to find an unanimous response to this question, since some characteristics aren't applied to all types of agents because they depend on the circumstances and application that are intended for the agent. The most common and widely accepted characteristics, and common are [1] autonomy, cooperation, reactivity/proactivity and intelligence/adaptability.

An agent is considered autonomous when is capable of act alone without help of external entities. The agents are cooperative if they are able to interact and communicate with others agents with the objective to reach a common goal. Another important characteristic of the agents is their capabilities to react upon changes in the environment, and the capabilities to evaluate their reactions according to a final goal, and this is what it is called a reactive/proactive agent. An agent is considered intelligent/adaptable when is able to learn and adapt its behavior according to changes in the environment when a better solution is discovered.

In conclusion, MAS is suitable to address the current manufacturing requirements, namely in terms of flexibility, re-configurability and responsiveness. However MAS applications present some problems, particularly in terms of interoperability, which are crucial in distributed and heterogeneous environments, as manufacturing systems are.

2.1.2. Service oriented Architectures (SoA)

Service oriented Architectures are an emerging paradigm with a vast, complex and multidisciplinary fields of application, which was initially applied to business and electronic commerce. A possible definition is [8]:

"*A service-oriented architecture is a set of architectural tenets for building autonomous yet interoperable systems".*

This definition is very limited but it shows at once two of the most important characteristics of SoA systems, the autonomy and the interoperability.

SoA systems can be considered autonomous because there are no direct dependencies between the services and they operate independently from environment.

This definition is very limited but it shows at once two of the most important characteristics of SoA systems, the autonomy and the interoperability. SoA based systems can be considered autonomous because there are no direct dependencies between the services and they operate independently from environment. Interoperability of SoA systems is achieved by rather than detailing the operations performed by the service provider, specifying an interface that describes the services being hosted and the interaction patterns considered.

However, there are other important characteristics that are important to refer, namely the platform independence that consist on having the services described using text-base formats, such as XML (eXtensible Markup Language), WSDL (Web Service Definition Language) and ebXML (Electronic Business using eXtensible Markup Language). These representations are not tied to a particular computer architecture, operation systems, programming language or technology and can be easily decoded by any system. Encapsulation of services is other characteristic found in SoA and consists on self-contained functionalities that are exposed by user defined interfaces hiding unnecessary details. By composing and orchestrating services a very complex level of functionality can be offered through a clean and simple interface. Finally we have the availability of the services that can be explained by the capability of publishing the services in public registries and made available for general use.

The introduction of service oriented principles in manufacturing systems allows the development of distributed, reusable and agile systems, exhibiting more powerful modular, interoperable and reconfiguration mechanisms. The bases of SoA systems, mainly how the processes of publication and discovery of services are performed, are represented, in a simplified way, in Figure 2-3.

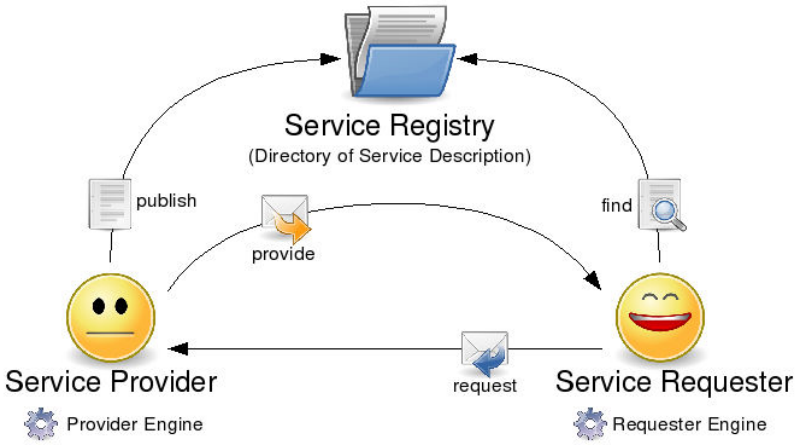


Figure 2-3- “Magic” SoA triangle

Basically, when a service provider wants to offer a service it publishes it in a service registry, which can be invoked later by any intervenient. In the process of manufacturing when a service is needed, means that one service is requested so it is made a search in the service registry seeing if it is someone capable and available to do the predicted task, after this search stars the communication between the services provider and requester. Some technologies used in the choreography and orchestration are WSDL for the abstract description of the services interfaces and access to platform, SOAP (Simple Object Access Protocol) that is responsible for messaging and communication, and UDDI (Universal Description, Discovery and Integration) for registry and discovery of the services.

One example of application of SoA, to solve some of the manufacturing problems is the SOCRADES project.

One of the main targets of SOCRADES project is to create a consistent architecture that can provide the flexibility and dynamicity required by the next generation of manufacturing systems. SOCRADES is fundamentally based on the adoption and integration of the concepts provided by SoA along all levels of the

manufacturing pyramid. Considered at the lowest level of this approach are the devices available on the factory floor, which may offer their complex functionality as service interfaces encapsulated by web standards. This mechanism is very important since it permits the loose integration of different types of devices, independently of the manufacturer or underlying technology. Figure 2-4 shows the SOCRADES control architecture, which, besides services, also includes other components such as Orchestration Engine(s), Orchestrator, and a Decision Support System.

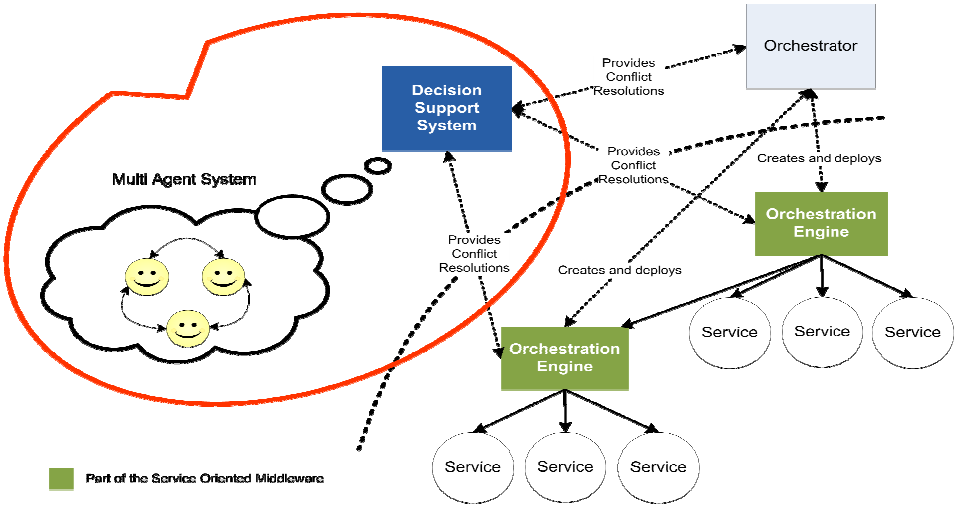


Figure 2-4- SOCRADES control architecture

The manufacture of a specific product depends on a set of production processes that must be controlled according to the production plan. This means that the execution of services representing different production processes must be orchestrated (i.e., sequenced and synchronized) by an OE (Orchestration Engine). According to Figure 2-4, there may be more than one OE available in the production system at a time.

2.2. Petri Nets

Petri nets are a graphical and mathematical modeling tool applicable to many systems, suitable for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams and networks [9].

A Petri net may be identified as a particular kind of bipartite direct graph populated by three types of objects. These objects are places, transitions and direct arcs connecting places to transitions and transitions to places, as it is possible to see in the following figure.

Figure 2-5- Objects in Petri Nets: (a) place, (b) transition, (c) arc connecting a transition to a place

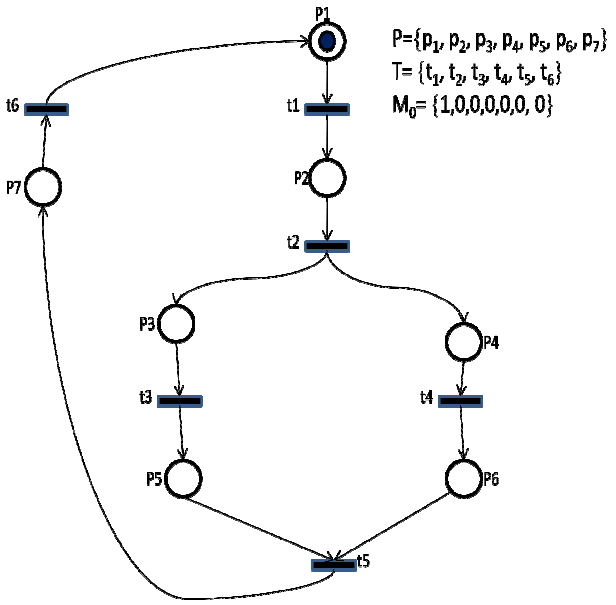
A **place** is represented by a circle and can be interpreted as being a state of the system (provisory state or one waiting stage); it can also be used to represent the availability of a resource / service. A **transition** is represented by a bar and represents one possible evolution in the system, like the execution of an action or the occurrence of one event / call for a service / expose service. An **arc** is represented by an arrow and allows connecting transitions and places, and places with transitions; it isn't possible to connect places with places and transitions with transitions.

Formally a Petri net can be defined by the tuple {P, T, I, O}, where [8]:

- $P = \{p_1, ..., p_m\}$ is a finite set of places.
- $T = \{t_1, ..., t_n\}$ is a finite set of transitions.
- $I: (P \times T) \rightarrow N$ is an input function that defines directed arcs from places to transitions. Each element of I represents the weight of the input arc from the place p_i to the transition t_j.
- $O: (P \times T) \rightarrow N$ is an output function that defines directed arcs from transitions to places. Each element of O represents the weight of the output arc from the transition t_j to the place p_i.

In order to study and simulate the behavior of the modeled system, in terms of its possible evolutions, each place can hold either none or a positive number of tokens represented by a black dot. The presence or absence of a token can be seen like a condition of true or false, or the number of resource presents in the system. If a Petri

net holds one or more tokens it is called marked, see Figure 2-5. The initial marking of the Petri nets model is represented by M_0.



I=

	t1	t2	t3	t4	t5	t6
p1	-1	0	0	0	0	1
p2	1	-1	0	0	0	0
p3	0	1	-1	0	0	0
p4	0	1	0	-1	0	0
p5	0	0	1	0	-1	0
p6	0	0	0	1	-1	0
p7	0	0	0	0	1	-1

Figure 2-6-A marked Petri net and incidence matrix

In the illustrated Petri nets model, the token is on the place p_1, and the system can evolves to another state if the firing conditions are granted.

The graphical representation has a correspondent mathematical representation that is based on the incidence matrix which defines all the possible interconnections between places and transitions in a Petri nets model [9]. Figure 2-5 also illustrates the incidence matrix, I, for the exemplary Petri nets model. The use of matrix equations allows representing the dynamic behavior of Petri nets.

As it was said before, a place is provisory state or a waiting stage, in the system presented the token is on place p_1, the system can evolutes to another state if the firing conditions are granted, in this case the only transition that can be fired is t_1, because it is the only that have an arc connecting it to a place with a token when this transition is fired the token moves to place p_2. For instance, the transition t_5, is connected with places p_5, p_6, and p_7. Places p_5 and p_6, can be seen like inputs or preconditions and p_7, as an output, to be able to fire the transition t_5, the preconditions have to be granted, this means that the input places have to own tokens. When a transition is fired the actual marking is changed, see figure 2-6.

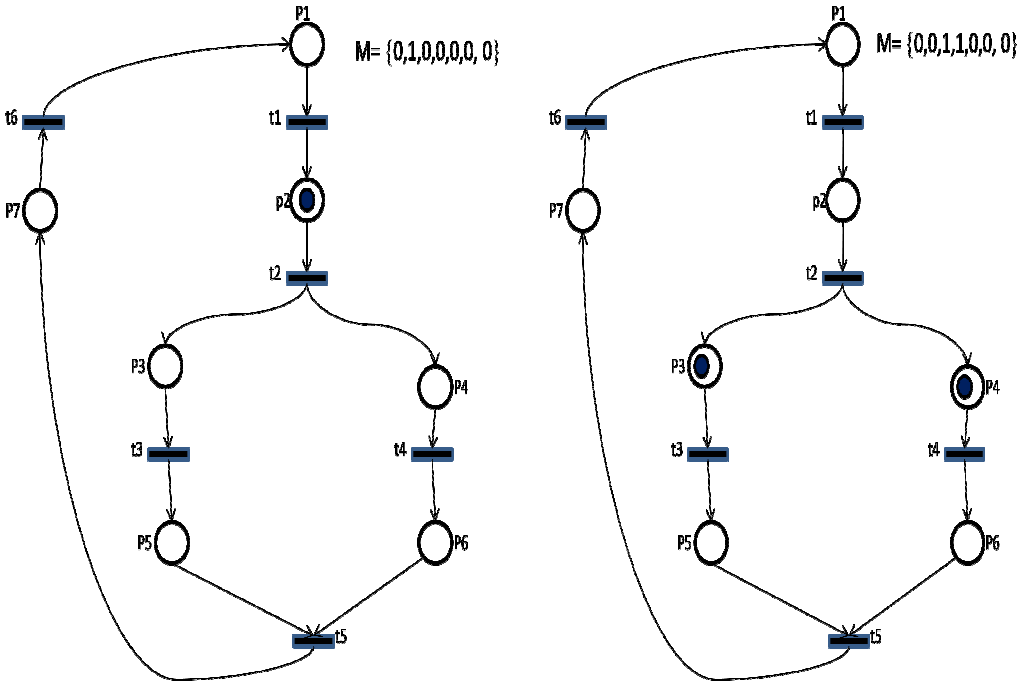


Figure 2-7-Evolution of a Petri net

When the preconditions are granted, the system is able to evolutes, in the figure above are two example of the evolution in the system and who the marking in the Petri net changes with this evolution.

Petri nets as mathematical tools process a number of properties. These properties, when interpreted in the context of the modeled system, allow the system designer to identify the presence or absence of the application domain specific functional properties of the system under design. Two types of properties can be distinguished: behavioral and structural properties. The behavioral properties are those which depend on the initial state, or marking, of a Petri net. One the other hand, the structural properties depend of the Petri net structure and topology, and not of its initial marking of a Petri net [10]. The main properties of a Petri net model are:

- **Reachability** - indicates if all the states of the Petri nets model are reachable from a different marking, and if the path of fired transitions to achieve that objective is the predicted one.
- **Boundedness and Safeness** - identify if the modeled system has some overflow in the places, i.e. a limited number of tokens in each place. The

importance of this property is related with the use of the places, in some situations of modeling, as a location of storage information and control. The Safeness is a particular way of boundedness, where all the places can only receive one token.

- **Conservativeness** - a Petri net is considered safe if the number of tokens is conserved during the evolution of the system. From a structural point of view, this can only happen if the number of input arcs to each transition is equal to the number of output arcs. But in real systems sometimes it doesn't happens because some actions need synchronization, combination of resources, etc.; so the Petri net is said conservative only if is possible to associate weights to places allowing for weighted sum of tokens in a net to be constant. This is very important because allows to analyze if the system can evolve in a consistent way.

- **Liveness** - a Petri net is called live if for all the possible markings of the net, exist the possibility to fire at least one transition. This property prevents the existence of deadlock; if a net isn't live it is provable that the system crashes.

- **Reversibility** - allows the modeler to know if the Petri net is capable to return to the initial marking. This property is important because if a Petri net is reversible the system is able to recover from failure states, and assumes that doesn't exist deadlocks.

- **Repetiteness** - this property allows the modeler to know if the Petri net is repetitive, i.e. if the system can execute its behavior several times.

All these properties may be extracted from the analysis of the Petri nets model and calculated using the mathematical foundation of the Petri nets, e.g. using algebra theory. In fact, the mathematical foundation of the Petri nets is based on matrix equations, namely the incidence matrix, which defines every possible interconnection between places and transitions in the net, so it is clear that the structure of the Petri net stays properly reflected on this mathematical representation [See 9, 10 and 11].

The T-invariants and P-invariants are two concepts that are related to the incidence matrix. The T-invariants represents the firing counts of the corresponding transitions which belong to a firing sequence transforming a marking M_0 back to M_0 and the P-invariants can be explained intuitively in the following way: the nonzero entries in a P-invariant represent weights associated with the corresponding places so that the weighted sum of tokens on these places is constant for all markings reachable from an initial marking M_0 [10].

T- and P-invariants constitutes valuable information to support the decision-making: the analysis of P-invariants allows verifying mutual exclusion relationships among functions and resources, and the analysis of the T-invariants allows the identification of work cycles (i.e. the alternative paths to evolve). The quantitative analysis can be performed by means of the simulation of the temporized Petri nets models, allowing the verification of the system compliance with specified performance indexes, such as throughput and resource utilization, and the development of optimization strategies.

The knowledge extracted from the Petri nets models, especially the P- and T-invariants, have a proper meaning in the physical system:

- Co-related discrete states of composed resources belonging to, at least, one *state-invariant* (P-invariant in the Petri nets terminology). The set of state-invariants and their linear compositions represents all possible configurations of hardware resources that are able to expose a service.

- Co-related exposed services belonging to, at least, one *service-invariant* (T-invariant in the Petri nets terminology). The set of service-invariants and their linear compositions represents all possible service coordination paths that the system is able to expose.

Focusing our attention on the T-Invariants, it's possible to extract very important information from it, information that can be used on decision, production plans, T-flows of production, etc.

For more information about the Petri nets properties and mathematical foundation and about the algorithms that allow reaching the pretended results see for example [9, 10 and 11].

3. MODELING THE FLEXLINK DEMONSTRATOR USING PETRI NETS

In our lives, modeling is an essential task in the development and testing of complex systems. In the automation domain, and particular in flexible manufacturing systems, models are usually used to represent systems, given a global view of the functionality of the system, allowing simulating and finding errors that without visualization were impossible to be detected. In such systems, exhibiting important characteristics of concurrency, asynchronous operations, deadlocks, conflicts or resource sharing [12], it is important to use a formal modelling tool that have the capability to validate the behavioural characteristics of these event-driven systems, and also to analyze other important aspects, such as the deadlock detection and the performance analysis. As referred in the previous chapter, Petri nets formalism is suitable to address this challenge.

Model is an abstract representation (formal or experimental) of a portion of reality, intended to promote its understanding [13]. The advantage to model the reality or a system is that it can be made more explicit, simpler and easier to manipulate than the reality it is supposed to represent [14]. Usually, models are created to allow that the creator demonstrate the main idea of a creation and allow others persons that aren't familiars with that type of creations to understand and visualize them. Other application of modelling is the search for errors and designing problems.

Modelling is highly used in several areas, how is possible to see on the definition. Depending in the area that is used, modelling can be seen differently, e.g. the physic laws or models, but the main idea is always there, that consist on imitation of reality.

At this stage, the aim is to model, using Petri Nets, the functionality of the FlexLink demonstrator, which is localized in the facilities of Schneider Electric Automation GmbH in Seligenstadt, Germany, under the objectives of the SOCRADES project.

3.1. Case study description

The case study corresponds to the FlexLink® Dynamic Assembly System (DAS) 30, illustrated in, which is a transport system that comprises several unidirectional and cross conveyors arranged in a closed-loop configuration, and two lifters connecting the upper and lower systems. Figure 3-1 shows a real picture and a virtual representation of the system.

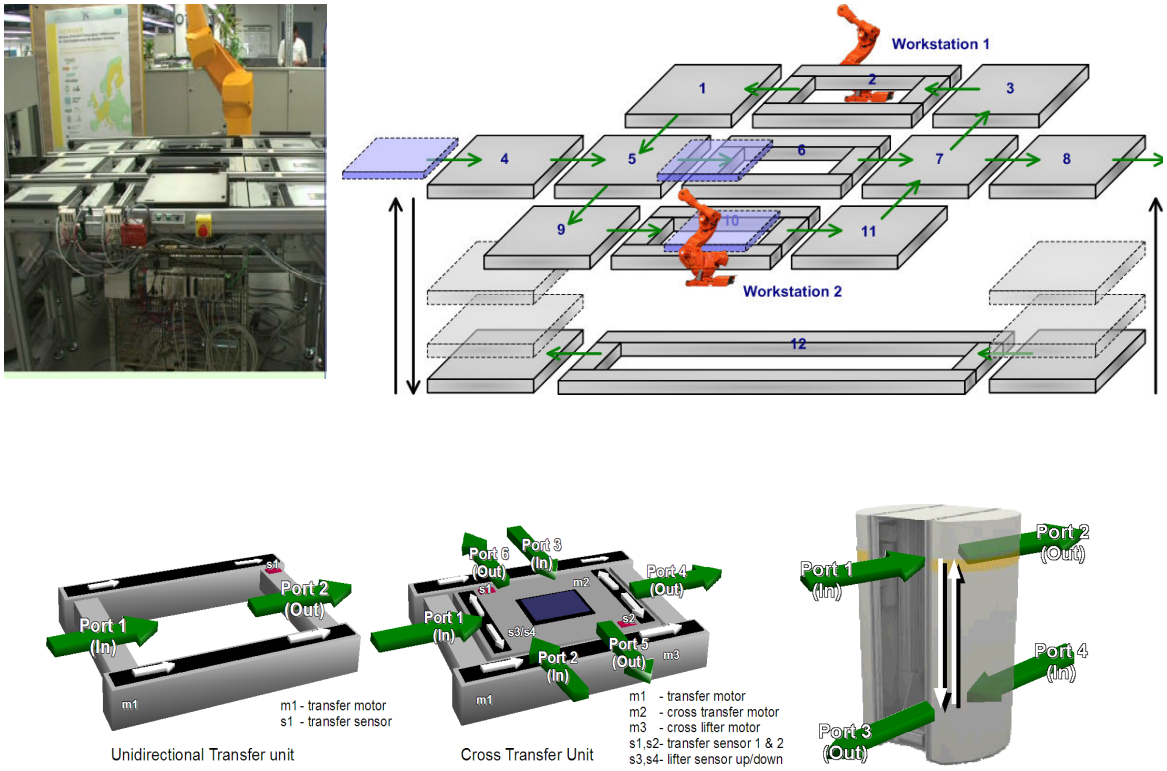


Figure 3-1 - Layout of the demonstrator and types of transfer units [14]

The following table summarizes the characteristics of each transfer unit belonging to the central part of the transfer system.

Unit Id	*Type*	*RFID*	*Work Station*	*Multiple I/O*
1, 3, 9, 11	Cross	✖	✖	✖
2, 10	Unidirectional	✔	✔	✖
5, 7	Cross	✔	✖	✔
6, 12	Unidirectional	✖	✖	✖
4, 8	Lifter	✔	✖	✔

The lower transfer unit (12) has the same behavior as the normal unidirectional transfer unit (e.g. the unit 6), but it is physically longer. Lifter units are identified by the units 4 and 8 in Figure 3-1. Besides being the interface between the upper and lower part of the system, they are also responsible for transferring pallets into and out of the factory cell.

The pallets enter in the system through the lifter 4 and are conveyed using alternative paths to achieve the two workstations associated to the transfer units 2 and 10. These transfer units have the possibility to halt the pallet during the required amount of time for the execution of the operation. At the last the pallets are routed outside through the lifter 8. Each transfer unit has a RFID (Radio-Frequency IDentification) reader/writer for identifying the pallets and transmitting information to them.

3.2. Models of the FlexLink Demonstrator

The methodology used in this work uses the service-oriented principles and High-level Petri nets as formal language to describe and execute the structure and predictive behavior of the case study (see [10] for information about the type of High-level Petri nets used). It uses a bottom-up approach, which consists in:

- Modeling the behavior of resources like robots, machines and transport components, using High-level Petri nets. The models represent all possible discrete states of such a resource and also all the manufacturing functions that this resource is able to expose as services, e.g. move-piece, pick-part and transfer-pallet. The modeling approach generates a set of resource´s discrete states that fulfill basic properties, such as boundedness and conservativeness [9], and a set of resource´s exposed services that fulfill basic properties, such as repetitiveness and liveness.

- The resources models are composed into a coordination model. This task follows the same rules of configuring a required resource's layout, i.e. taking into account the competition, concurrency and shared resources behavioral

relationships, among others. The result is a High-Level Petri nets model of the whole factory.

The pallets enter in the system through the lifter 4 and are conveyed using alternative paths to achieve the two workstations associated to the transfer units 2 and 10. These transfer units have the possibility to halt the pallet during the required amount of time for the execution of the operation. At the last the pallets are routed outside through the lifter 8. Each transfer unit has a RFID (Radio-Frequency IDentification) reader/writer for identifying the pallets and transmitting information to them.

So, the models created for the experimental case study are based on the functions and movements allowed in the demonstrator. So, the cross transfer unities that in reality can have 6 ports, 3 for input tasks and 3 for output tasks, in the designed models only have the active ports. One active port is basically a port that is used in the system allowing a movement of a palette from a conveyer to another; for instance, the unity 5 illustrated in Figure 3-2 have 6 ports (i.e. Port1 – In, Port 2 – In, Port 3 – In, Port 4 – Out, Port 5 – Out and Port 6 – Out), but only 4 ports are active during its operation (i.e. Port1 – In, Port 3 – In, Port 4 – Out, and Port 5 – Out).

The model illustrated in 3-2, was created to represent the unidirectional conveyers 2, 6, 10 and 12 from Figure 3-1 of the demonstrator layout and the cross unities 1, 3, 9 and 11 from the same figure, that only have two active ports. This model only allows two actions: conveying in a palette from the port 1 and conveying out from the port 2. These actions are static, i.e. it is impossible to change the direction of the movements of the palettes without changing the model's structure.

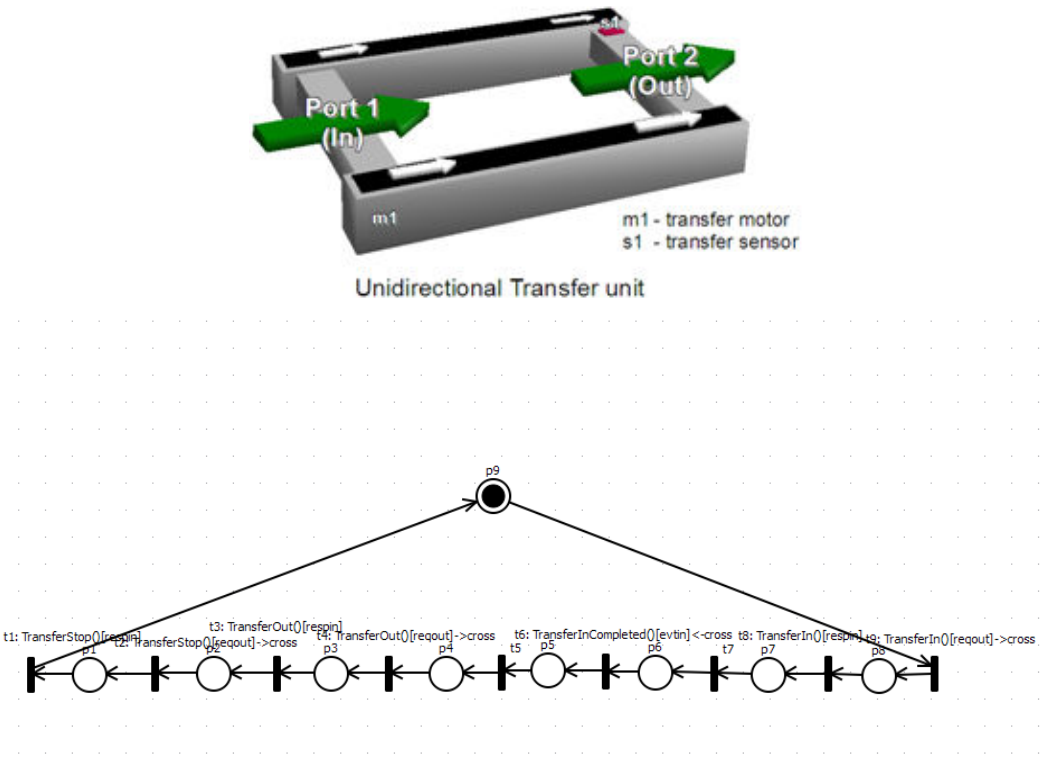


Figure 3-2 – Petri nets models for unidirectional conveyers and cross transfer unities with two active ports

This model is constituted by nine transitions and nine places. The place p_9 represents the availability of the resource. As it is possible to be seen by analyzing the model, it comprises two different parts. The first one associated to the input port, responsible for the transfer in operations, that is constituted by the transitions t_6, t_7, t_8 and t_9, where three of them already complete functionality installed t_6, t_8 and t_9, are ready to be composed using Petri Nets Composer tool (this is familiar to all of the created models), the other transition t_7 was intentionally let there without any logical installed, meaning that when this transition is fired doesn't occurs any action in the system, and it was intentionally placed there because it is supposed that the models need more logical attributes, and in future if someone intend to use or improve this models they are ready for it.

The second part of the model, representing the port responsible for the transfer out operations, is constituted by 5 transitions, namely t_1, t_2, t_3, t_4 and t_5. The transition t_5 has no logic installed but it is needed to synchronize the transfer in and transfer out

operations between different models. The others transitions, i.e. t_1, t_2, t_3 and t_4, have logic installed and are responsible for transfer out operations. It is considered here that stopping the conveyer is part of transfer out operation; the transitions t_1 and t_2 are responsible for this action.

The next model, illustrated in Figure 3-3, represents the cross unity 5 and 7 of the demonstrator layout which has four active ports: two are inputs ports, responsible for the transfer in operations, and the other two are outputs ports, responsible for transfer the out operations.

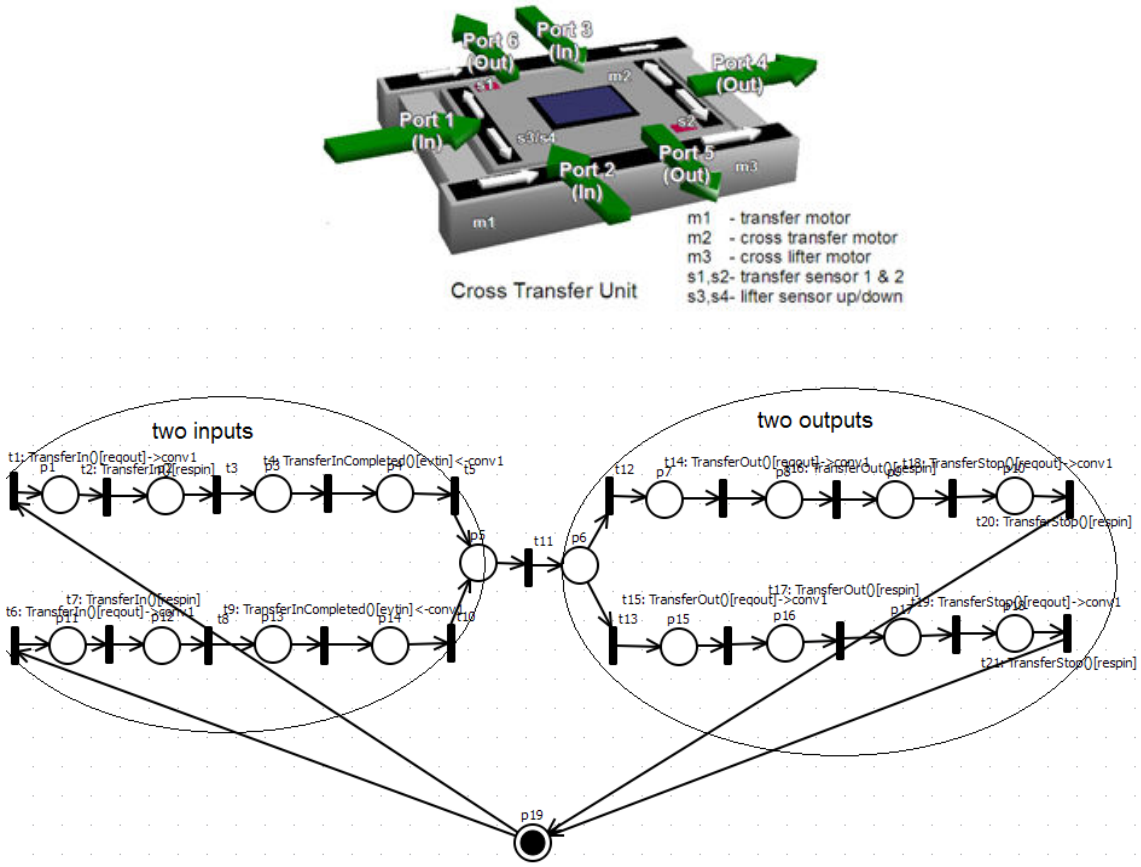


Figure 3-3 - Petri nets model for cross transfer unities with four active ports

The cross unity model is constituted by 21 transitions and 19 places. The place p_{19} represents the availability of the resource. This model has the same configuration of the unidirectional conveyer model, but is a little bit more complex because of the number of active ports, and the increased number of possible movements as a consequence of the cross transfer operations possibility.

A palette can be transferred in from two possible ports: one constituted for transitions t_1, t_2, t_3, t_4 and t_5, and another constituted for transitions t_6, t_7, t_8, t_9, t_{10}. The model can only receive palettes from one of the ports at the time, because when one of the ports receive one order to transfer in a palette, the token that represents the availability of resource is consumed and don't allow that the other port to receive a palette.

After the palette is received on the cross unity, there are two possible ports to convey the palette out, one constituted for transitions t_{12}, t_{14}, t_{16}, t_{18} and t_{20} and another constituted for transitions t_{13}, t_{15}, t_7, t_{19} and t_{21}. At this moment, it is needed to create a conflict situation, to the decision support be called and decide which one of the ports will be used to transfer out the palette; for this purpose, the places p_5 and p_6 and transition t_{11} are included in the model. The decision support for conflict resolution situations will be deeply explained in next sections.

The lifters models were supposed to have two active ports, but it was established that all the ports should be defined because this lifter system is prepared to be connected to similar systems. The Petri nets model illustrated in Figure 3-4 represents the lifters 4 and 8 of the demonstrator layout.

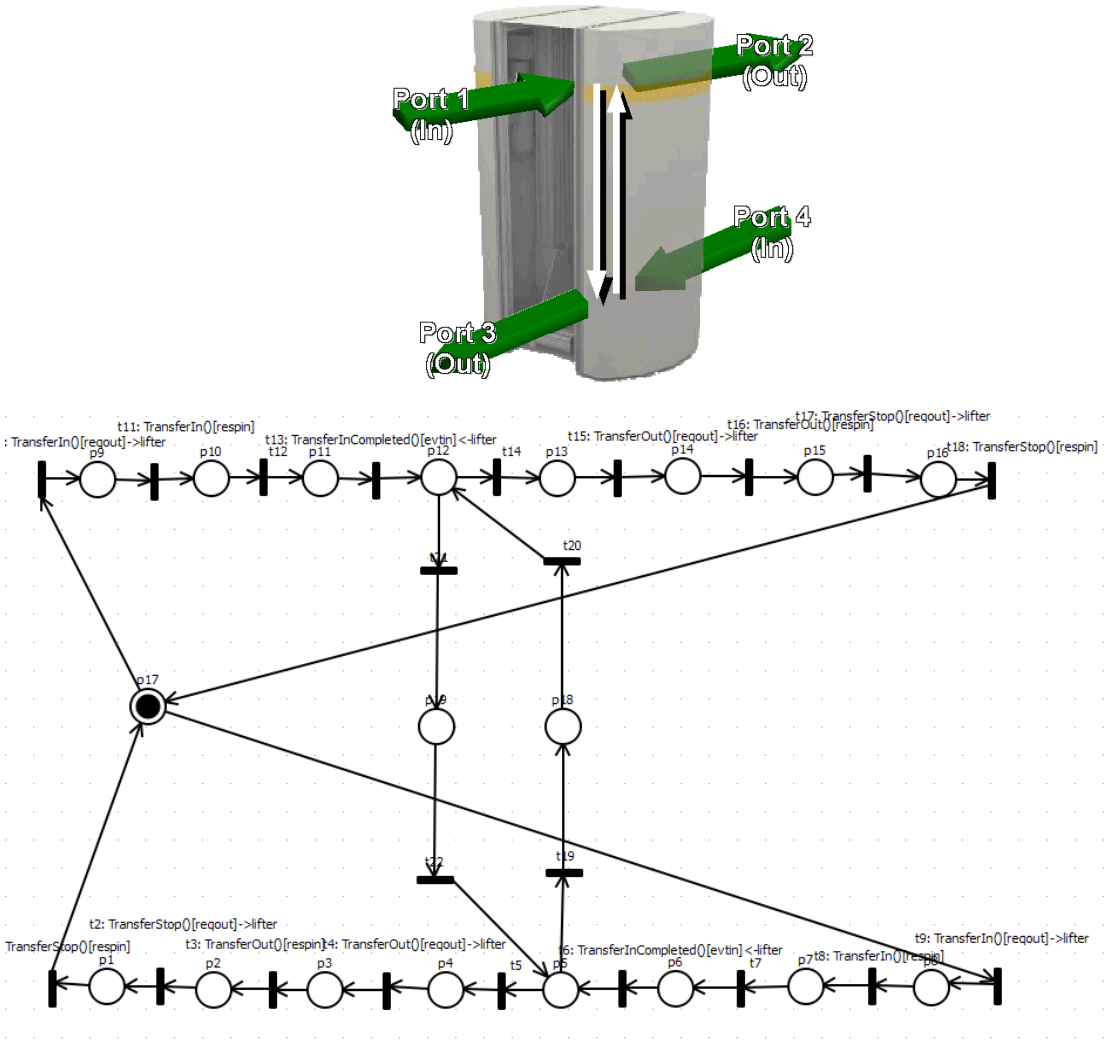


Figure 3-4 - Petri nets model for the lifter unities

The lifter model is constituted by 22 transitions and 19 places. The place p_{17} represents the availability of the resource. The model may look complex but in fact it is simple and easy to understand. Looking to the model, it is possible to see one upper region and a lower region; if the central part is disconnected it is obtained two parts similar to unidirectional unities. So, the upper region has two ports, one input (left side of the model) constituted by transitions t_{10}, t_{11}, t_{12} and t_{13}, and other for output (right side of the model) constituted by transitions t_{14}, t_{15}, t_{16}, t_{17} and t_{18}; the lower region also has two ports but they are exhibited in an opposite way, the input (right side of the model) constituted by transitions t_6, t_7, t_8 and t_9, the output (left side of the model) constituted by transitions t_1, t_2, t_3, t_4 and t_5.

As explained before, the lifter is the part of the system where the palettes get in to be conveyed to the workstations or get out the system after be worked. In the presented model the entrance of palettes is made from port 1 that is located in the left of the upper region, and the exit is in the lower region also in the left.

The central part of the model has 4 transitions and 2 places, and can be considerate like a composition of the models, i.e. the connection between the two regions. The central transitions have no logic installed, but they are crucial to model the conflicts that exist; associated to the transitions are a decision support system that decides which way the pallets should take.

In this model it is supposed that more logic will be added to the transitions. When a palette calls for a transfer in the lifter and the platform isn't in the exact position to receive it, a problem occurs. The logic to be add will be responsible to put the platform in the right position and only after this the transfer in will begin. It is possible to solve this problem in a simple way because the ports are very well defined and when one of the ports is activated the system knows exactly where the platform was to be and it is always the same routine when a port is activated, for instance, if a palette ask for access in the base, we know exactly that the port that allows entrance in the bottom of the lifter is the port located in the bottom right because the other is responsible for exit of palettes.

The model is prepared to be connected to others similar systems by the lifters, using all existing ports. Note that in our demonstrator the lifter only have two active ports, one in the upper region that connects to the conveyers in the central part and other in the lower that connects both of the lifters that is made by a unidirectional conveyer.

Now, that the models are created, the next step is to perform the assessment of these models and simulate them to search errors and see if the models are representing correctly the intended reality.

4. ASSESSMENT OF THE PETRI NETS MODELS

Due to the strong mathematical background that is behind the Petri nets theory, the models were formally analyzed and validated [15-16]. Basically, two types of analysis can be made: the qualitative and the quantitative analysis. The qualitative analysis, based on the structural analysis of the matrix representation of the graph model, allows the verification of the structural and behavioral properties of the High-level Petri nets model, extracting conclusions about the operation of the system, such as the existence of conflicts and deadlocks, the bounded capacity of resources, and possible control sequences [15]. The quantitative analysis, are based on statistical methods, using timed-transitions or associating statistics distributions to transitions that allow a better representation of the intended reality, given the opportunity of extracting information about the performance of the Petri net model.

After having modeled the case study system with Petri nets, it will be used the major strengths of Petri nets to support the analysis of properties and problems associated with the modeling tasks. The assessment of the models assumes a major importance, especially because it allows the designer engineer to learn more about the models and with this knowledge be able to see if the models correspond to the intended behavior and corresponding structurally to intended reality. In this section, it will be done the assessment of the developed models, analyzing the behavioral and structural properties, already explained in a previous section.

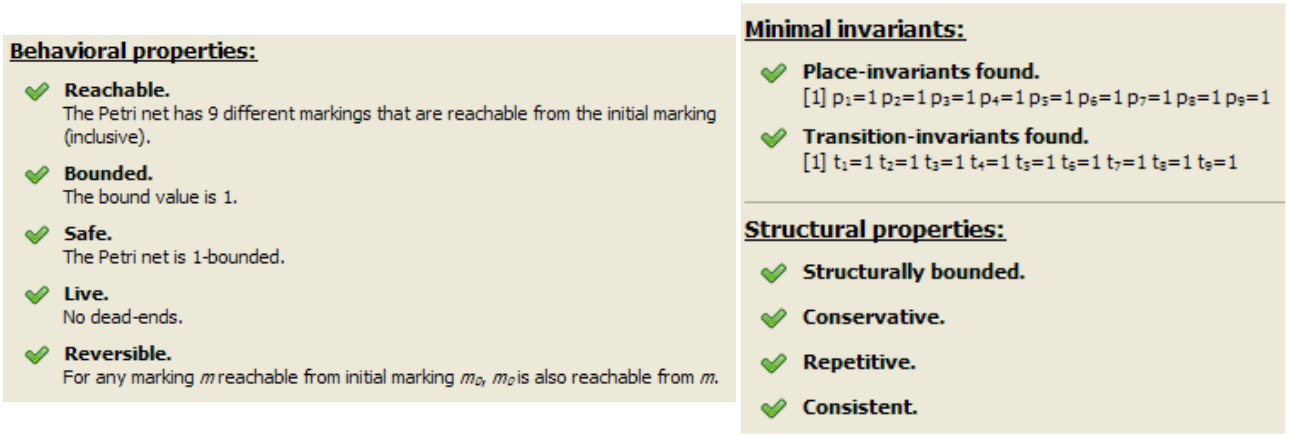
The assessment procedure will be done using the CDS (Continuum Development Studio) tool [17], which provides a computational platform that offers the capability to analyze and simulate Petri nets models. All the designed Petri nets models were structurally and behavioral analyzed, and simulated. However, in this work, only the assessment of one Petri nets model will be described, the unidirectional conveyer model, to exemplify the procedure.

The first step is to classify the Petri nets model, verifying if it is a valid one (i.e. if all places and transitions are connected and it does not present any structural

problem, e.g. the existence of two places connected each other) and if it is marked. As illustrated in Figure 4-1, the Petri net model for the unidirectional conveyor is an ordinary and marked Petri net, consisting of 9 places and 9 transitions, being all places and transitions connected. This initial analysis allows concluding that the model is valid to behavioral and structural analysis and for simulation execution.

Figure 4-1-Classification of the Petri net model

The next step in the assessment of the Petri nets model is to run a structural and behavioral analysis, as illustrated in Figure 4-2, addressing the properties referred in the section 2.2, and using proper algorithms for this analysis [9, 10].



(a) (b)

Figure 4-2- Behavioral and structural analysis of the Petri nets model

From the results obtained, it is possible to see that all the behavioral and structural properties analyzed were granted by the model, namely the reachability, boundedness, safeness, liveness and reversibility. This means that:

- All the places are connected and are able to be reached at least one time.

- Doesn’t exist over flows in the places, this indicates that it is impossible to have more than one palette at time in each unity.

- There are no place in the system that crashes the system avoiding him to come back for is initial state, i.e. in the system after a conveyer be used it can be reused as times as the user want. And the system is able to restore is availability state or initial marking, is capable of recovering from errors or failures.

Additionally, it is also possible to extract the P- and T- invariants of the analyzed model. In fact, the Petri nets model has the following invariants:

- A P-invariant, { p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9}, which means that the palette have to pass all this places to be conveyed from the input port to output port, i.e. Left to right side of the unidirectional conveyer, place p_9 is the place that indicates that conveyer is available for another operation.

- A T-invariant, {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9}, which means that this path of transitions or actions have to be fired to complete a cycle, i.e. for a palette be conveyed for this unidirectional unity all this transitions have to be fired, when the last transition is fired the token returns to place p_9.

The information extracted from the Petri nets models using the behavioral and structural analysis will be used latter in the section of the decision support system.

After achieved the confirmation that the models were correctly constructed, the next stage is related to the simulation of the model, which is crucial to understand the functionality of the model, allow to verify if it represents the system specifications. This simulation was granted by one tool of the CDS which allows the token game performed in the Petri nets.

During the simulation of the individual Petri nets models for the case study system, some conflicts were found. The decision points/conflicts in the complete transport system model represents a fork in the paths of the palette, upon which it can either continue straight on to the end lifter, or turn in the direction of one of the two

workstations. The decision to be taken at this point is related to decide which workstation should perform the operation and consequently which the path should be chosen. In the Petri nets model representing a cross transfer unit, the decision points represent a fork in the paths, at this point the palette have to choose which port take to the system evolution continues as planned.

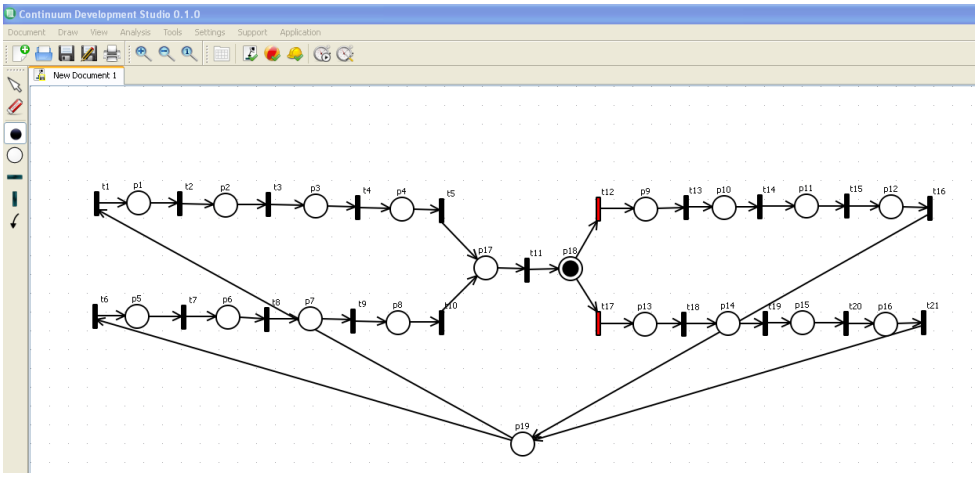


Figure 4-3- Conflicts in the Petri nets model for the cross transfer unit

In the cross transfer unit, it was find one conflict that is derived from the presence of two output ports in the model. After a palette is conveyed into the transfer unit there are two possibilities for conveying out the palette: as illustrated in Figure 4-3, when the system reaches the place p_{18} it is possible for the system to fire the transitions t_{12} or t_{17}. The option between the two transitions requires a decision support system that decides which transition should be fired, i.e. if palette follows the output one or the output two.

The Petri nets model for the lifter has more conflicts, see Figure 4-4, mainly due to the existence of higher number of ports.

In this first conflict, figure 4-4, the palette finds itself in the upper section of the lifter, and now it is needed for decision support system. The two possible paths, one that drives the palette to the output port that is active when transition t_{14}, is the chosen one, and other that drives the palette to the lower section of the lifter and this option is chosen when transition t_{21} is fired.

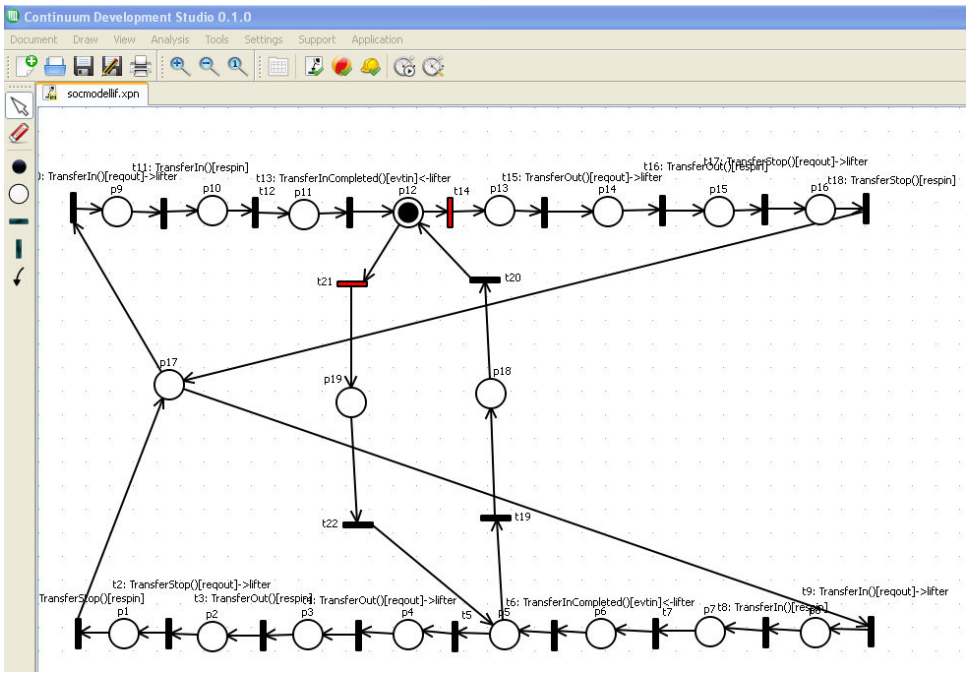


Figure 4-4- First conflict in the Petri nets model for the lifter

In the second conflict, the palette finds itself in the lower section of the lifter, and now it is needed for decision support system. The two possible paths, one that drives the palette to the output port that is active when transition t_5, is the chosen one, and other that drives the palette to the upper section of the lifter and this option is chosen when transition t_{19} is fired.

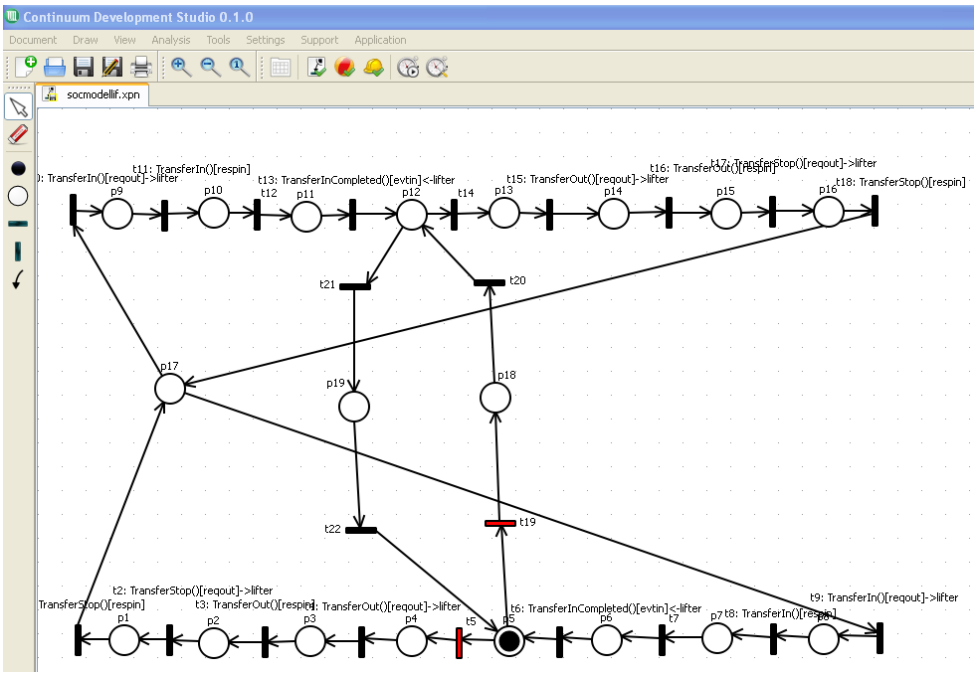


Figure 4-5- Second conflict in the Petri net model for the lifter

The assessment procedure described during this chapter was repeated for all of the created Petri nets models. In the following chapter it will be explained how the individual Petri nets models were composed to achieve the final and global model.

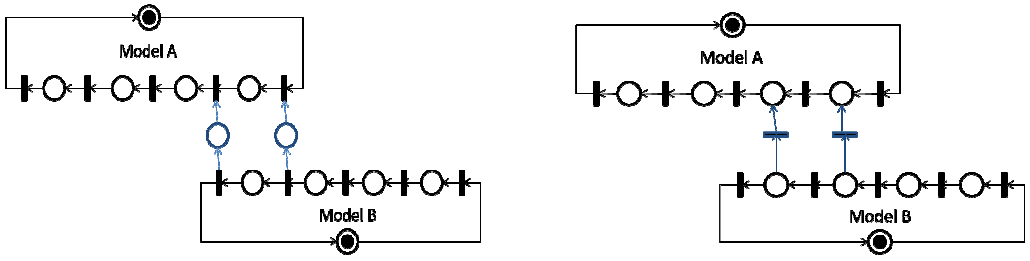
5. COMPOSING PETRI NETS MODELS

The creation and analysis of the individual Petri nets models is crucial when developing behavioral models for individual components. The achievement of a global system model requires the study of the best way of join these individual Petri nets models together; this task is denominated for composition of models. This type of composition follows the same basic principles that are found in [18] and [19], and applies it to Petri net-based models.

The composition of the models is very important, because one thing is have small models work properly separated and another is to be able to join them in a singular model (i.e. synthesizing) and be able to put all of them communicating and synchronized to achieve the proposed goal. Besides that, the global model is much more complex and confuse for analyze. In terms of modeling, the possibility of composing small models is a huge step to avoid failures and errors. The chance of forget a transition or a place or something even more complex is always present when the goal is construct a final model, besides that when the modeler is composing it is possible to run a test and search for the non welcome errors or failures. In terms of the Petri nets, even for more experts in this technique, is difficult to understand what is intended to represent by the models presented there. So imagine the difficulty to find and correct an error in these conditions.

The interaction and synchronization between system components is performed by Petri nets modes. A set of methods have been developed and long time applied for the interactions of real-time systems. Such methods facilitate the synchronization and other kind of interactions like the communication and mutual exclusion specifications. Examples of structures and methods for handling interactions are [20]: Mutex (Mutual Exclusion Objects), Queue Theory, Mailbox and Rendezvouz. The coordination and composition of Petri net models can use some or all the above addressed methods. Due to the graphical and mathematical form of the Petri nets formalism, several

distinct ways can be used to implement the composition methods, namely considering transitions, places, transitions and places, etc., as illustrated in Figure 5-1.



(a) (b)

Figure 5-1-Example of compositions a- with places b- with transitions

Each one of the referred methods has a particular form of representation, e.g. a place plays the role of a MUTEX or a Mailbox, a sequence of places and transitions plays the role of a FIFO queue, the firing of a transition synchronizes two or more processes modeled by two or more input places to this transition, etc [20].

In this work, it was used places to perform the composition task, playing the role of Mailboxes.

5.1. Synchronization between models and communication of the models with the engine

After understanding the importance of composition of the models, it is important to explain what is intended in terms of communication and synchronization between the models and how will this be proceed in the designed models to be able to choose the type of composition that better applies to our case study. For this purpose, the aggregation of three unidirectional conveyors, from the case study presented before, will be analyzed, Figure 5-2. The conveyers are numbered by one to three, and the objective is conveying a palette from conveyer #1 to conveyor #2; the conveyer #1 receives the palette and then dispatches it to conveyer #2 that afterwards sends to conveyer #3 number three. The transportation of the palette between the conveyers requires synchronization and communication. The synchronization and communication between models will be explained using the capability of simulation provided by the CDS tool, through the token game functionality.

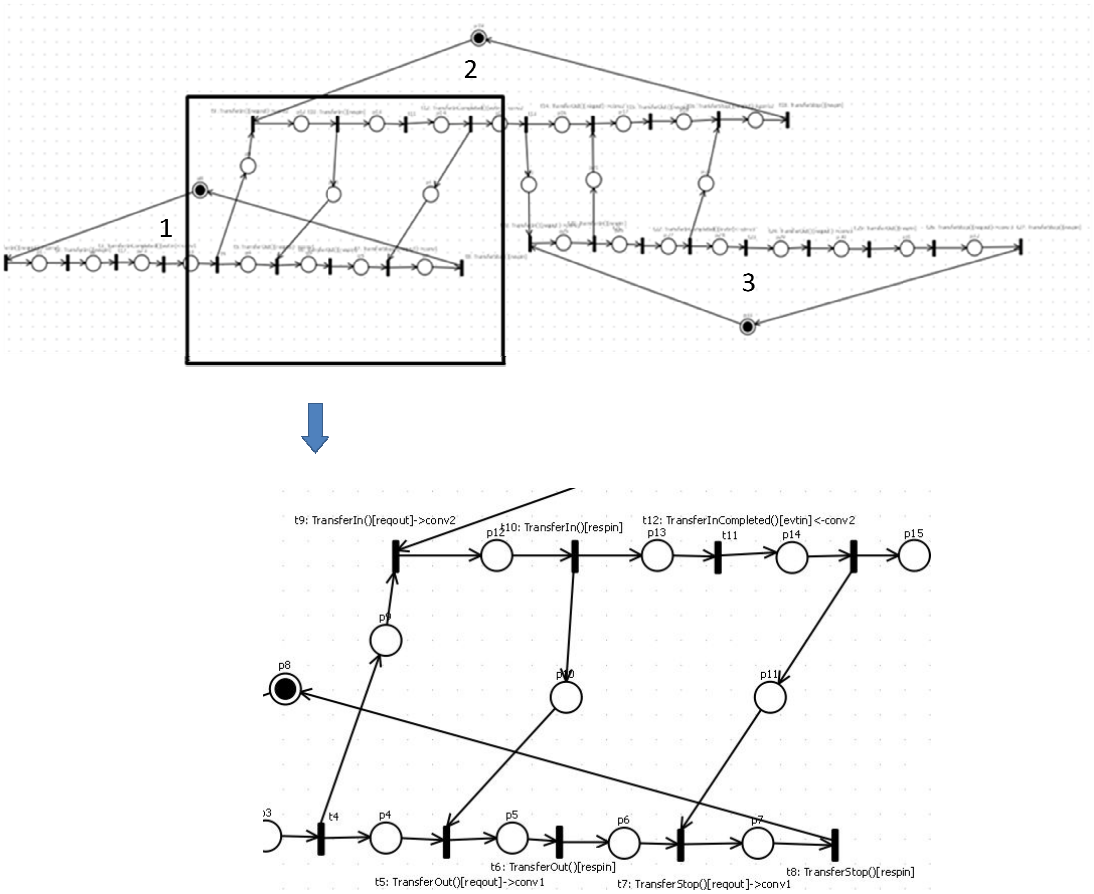


Figure 5-2-Composition of 3 unidirectional conveyers and zoom of the connections

Initially, the tokens are in the places representing the availability positions meaning that the three conveyers are available to receive the palettes. In the Figure 5-3 it is assumed that the conveyer #1 has already received a palette and it is now in a waiting stage, ready to start synchronization between the two conveyers and to start the operation of sending the palette (transitions t_4, t_5, t_6, t_7 and t_8) to the conveyer #2.

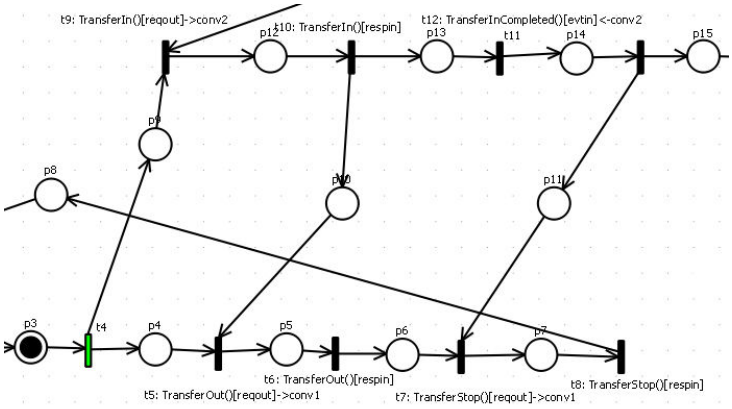


Figure 5-3- Token Game: Beginning of synchronization

At this stage, the engines of the conveyers are stopped because it is necessary to verify if the conveyer #2 is available to receive the palette. The transition t_4 hasn't logic installed, i.e. the purpose of this transition is synchronization of the models, when this transition is fired there are any kind of actions (Web services communication) occurring in the system.

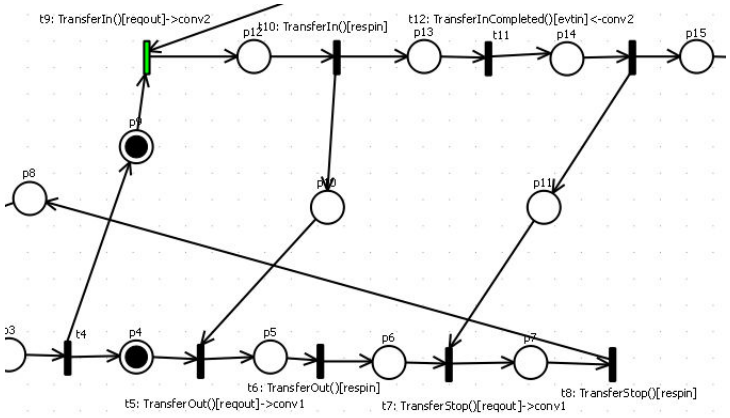


Figure 5-4- Token Game: conveyor #1 waiting to start transfer to conveyor #2

The place p_9 is considered a waiting stage and the transition t_9 can be only fired if the conveyer #2 is available. When the required conditions to fire the transition are granted, the transition t_9 is fired and this triggers the start of the engine of the conveyor #2, with the system evolving to a new state represented in the Figure 5-5. In the model of conveyer #1 it is possible to see a token in the place p_4 that means that it is waiting for the confirmation that the engine of conveyer #2 is working properly. This procedure is the beginning of the transfer *in* operation.

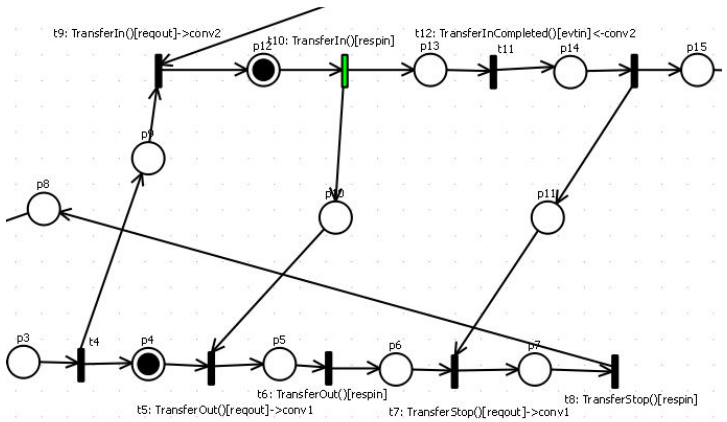


Figure 5-5- Token Game: conveyor #1 is waiting that the engine of conveyor #2 is already started

At this moment, the existence of a token in the place p_{12} fires the transition t_{10}, which represents the confirmation, to the conveyor #1, that the engine of conveyer #2 is working properly. The system evolves to a new state, illustrated in Figure 5-6, that consists in the beginning of the transfer *Out* operation, i.e. the conveyer #1 transfers the palette to the conveyer #2.

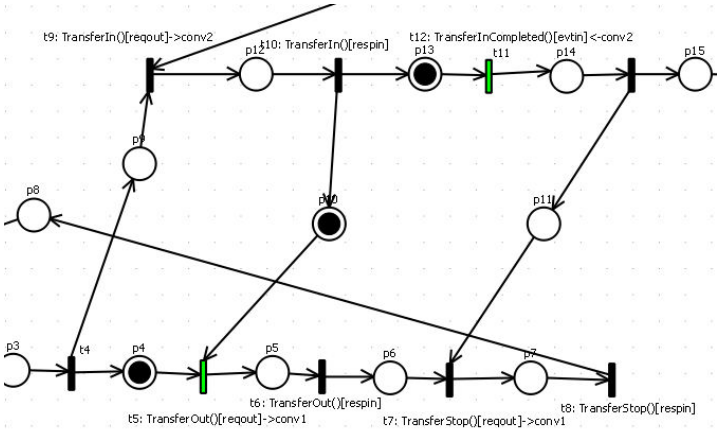


Figure 5-6- Token Game: conveyor #1 starts transferring out the pallet

Now that is assured that the engine of the conveyer #2 is working, it is necessary to put engine of conveyer number one working and this only happens when transition t_5 is fired, represented in Figure 5-7. The transition t_{11} has no logic installed, but it was placed there for future installation of functions that can read sensors embedded in the conveyer, and in this way monitor the evolution of the transfer operation.

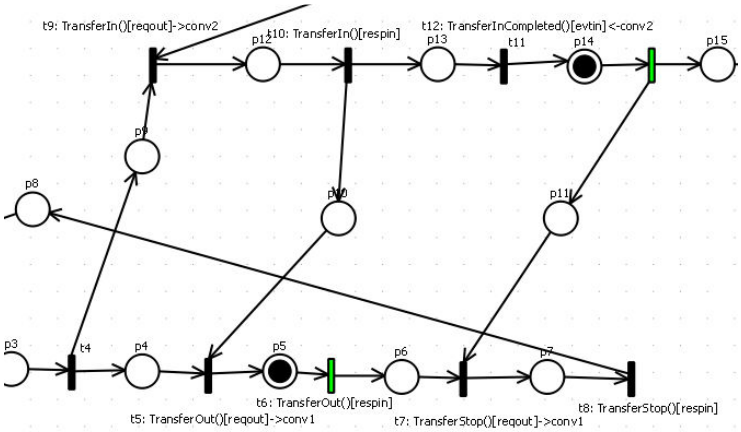


Figure 5-7- Token Game: the engine of conveyor #1 starts working

When the transition t_5 is fired, the engine of conveyer #1 is started. At this moment, the palette starts to be sent to the conveyer #2. Only when the transition t_6 is fired it assured that the conveyer is moving. Note that, only when the sensor presented on the conveyer #2 gives the indication that the palette is completely transfer, the transition t_{12} fires; in this case, and since it was used simulation without sensors, the transition automatically fires. This new state of the system is represented in Figure 5-8.

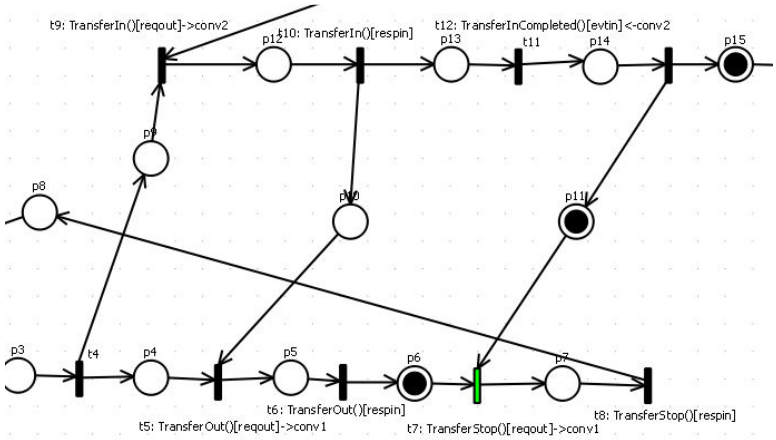


Figure 5-8- Token Game: transfer In completed

Now that the palette is transferred to the conveyor #2 and the transition t_{12} is fired, the engine of the conveyer #2 stops. However, it is also necessary to stop the conveyer #1, and this happens when the transition t_7 is fired. In Figure 5-9 it is possible to see that the conditions to stop the conveyer #1 are granted.

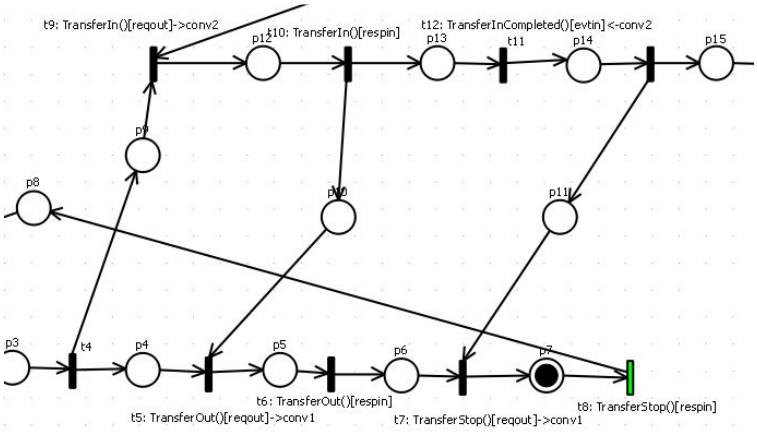


Figure 5-9- Token Game: conveyor #1 stops its engine

At this stage, the system is waiting for an answer of the engine saying that it is stopped. When it is assured that the engine of the conveyor #1 is stopped, the transition t_8 fires and the conveyer #1 returns to a waiting stage, available to starts another operation of transfer *in* of a palette, illustrated in Figure 5-10.

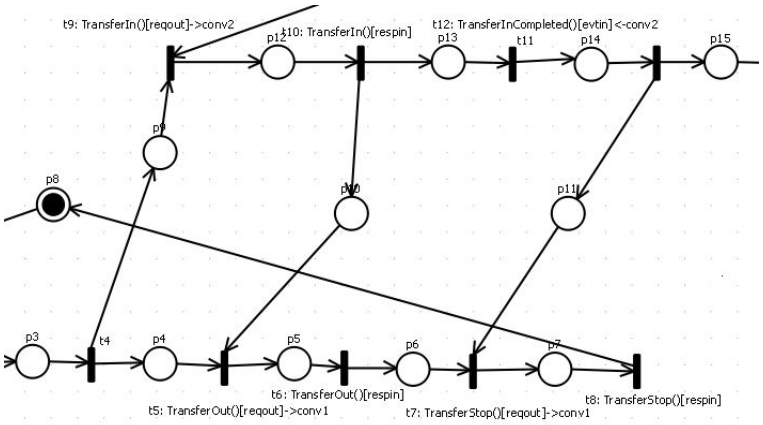


Figure 5-10- Token Game: conveyor #1 is able to start a new transfer in operation

In this example, it was explained how the palettes move between the conveyers. The described operations for transfer in and out are applied to all system, i.e. all the transfer operations of the system follows the same procedure. The communication between the models and the engines are made by Web services. The importance of synchronization is to assure that the palette isn't send to a conveyer that is stopped, because if that happens it has a possibility of damage the receiving conveyer. So, it is need to make sure that the conveyers' communication is working properly and the synchronization of the models is also assured. In this section, it was also possible to see the importance of simulation before apply the constructed models to real situation.

5.2. Composition of the models using the Petri Nets Composer

Now that it is understood the achieved synchronization and communication between the models, it considered that the design engineer is ready to compose the final and global model.

For this purpose it was used the Petri Net Composer (PNC), which is part of the CDS platform. The PNC tool allows the user to create simplified models and then link

them together into a global model, i.e. building the models of each part of the overall model and integrating them with PNC into a final model.

The value of this tool is unquestionable, because of the simplicity and easiness of its use. When a design engineer has to create a model for a complex and enormous system it finds a difficulty constructing it; even when it creates a complete model system it is normal and happens frequently that the modeler forgets some of the parts to be modeled and some details. The PNC tool creates a “bridge” between the models, allowing the modeler to create small and simplified models avoiding the complexity of a global system model and focusing its complete attention in details. Figure 5-11 illustrates an example where the PNC tool is responsible by the creation of the blue connections between two unidirectional conveyers.

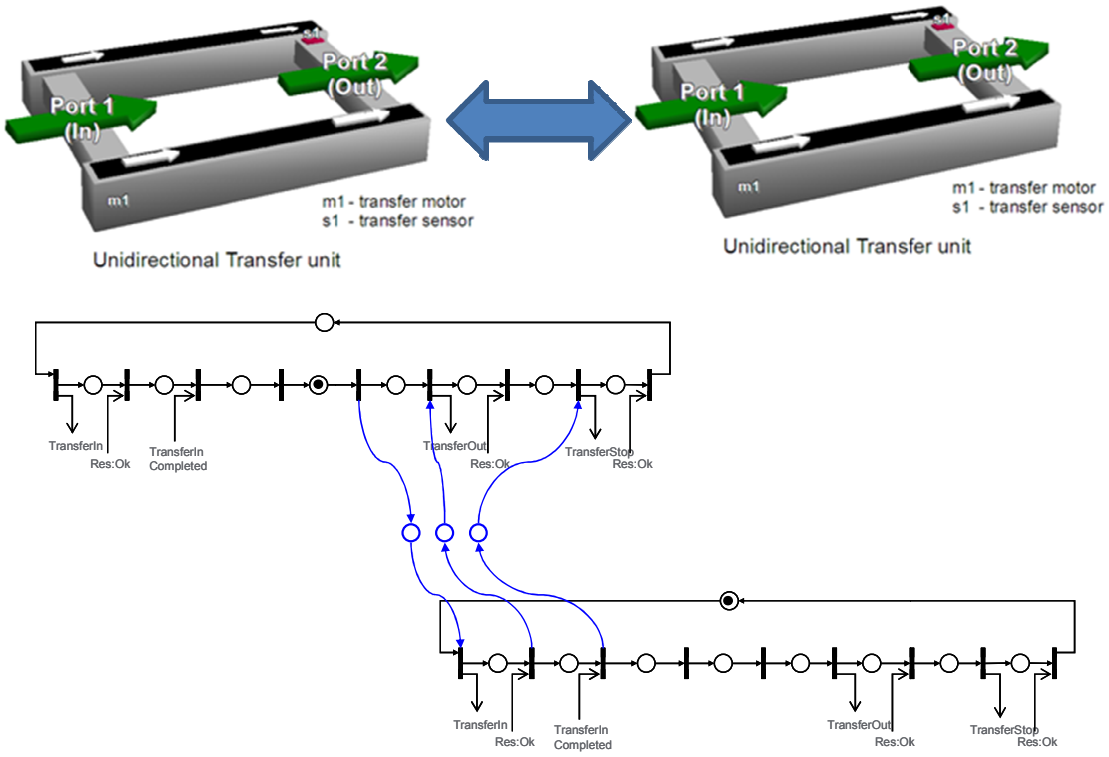


Figure 5-11- Example of a composition between two unidirectional conveyers using the PNC tool

The following procedure explains in a step-by-step manner how to use this compositional feature to compose single Petri nets models.

1. Give a name to the model;

a. Select the page layout;

b. Add a property on the Property editor, with property name resource, data type string and value “name of the model”;

2. After building the model, the user must define the ports of connection. For the example illustrated in Figure 5-12, of a unidirectional conveyor model with the transfer in and transfer out area, the user must define what transitions belong to each port. In Port1, we have the transitions that belong to the transfer out operation and in port2 we have the transitions that belong to the transfer in operations of the conveyor.

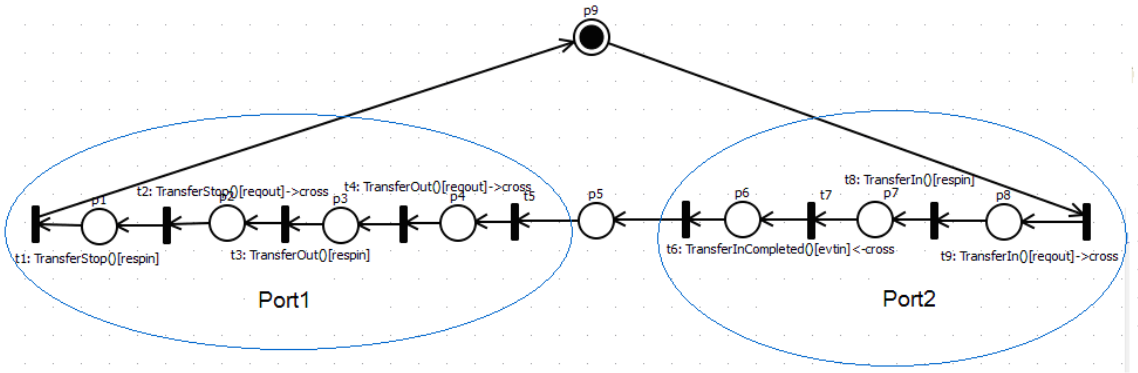


Figure 5-12 - Definition of ports for the connection of models

1. With the ports defined, the user should know which are the transitions of the port that will be connected with the others models. In those transitions the designer has to add a property with the name of the port that it belongs;

   a. Select the transition;

   b. Add a property on the Property editor, with property name port, data type string and value “name of the port which the chosen transition belongs”;

2. After all the transitions are defined and to which port they belong, the user should create a new property. On this property, will be defined if the transition will receive or send a token and the position of the transition in the port. When a transition receives a token the property name has to be a *port_in_seq* and when it is sending a token has to be a *port_out_seq*. A *port_in_seq* of one

model is always connected to a *port_out_seq* of the other model. The connection between the ports is made by the value of this property, i.e. the value one of one port connects with the value one of another model, so the user has to be careful with these details.

a. Select the transition;

b. Add a property on the Property editor, with property name *port_out_seq*, data type string and value “number of the transition on the port”;

3. Now that all the ports and transitions are well defined, the user has to create one xml file. This file will contain the information of the resource and port, and say which resource connects with other and by which port. To create this file it can be used an editor, such as Notepad, Notepad++ and Vim.

```
<?xml version="1.0" encoding="UTF-8"?>
<connections>
  <connection resource1="conv1" port1="port1" resource2="conv2"
              port2="port1"/>
</connections>
```

The meaning of this piece of text is to define that the resource with the name conv1 is connected by the port1 with the resource conv2.

4. At this moment, in the CDS tool, it will be created a file with the PNC.

a. Open the PNC tool;

b. Select the xml file created on step 5;

c. The tool will ask what xpn files the user wants to join in the final model; the user can select all the files at once or one at the time. Afterwards, the user can give a name to the final model.

Once done this composition procedure, the created file is opened in the CDS platform, being generated all the described connections. The final result for the described example is the connection between two conveyer models. This procedure was repeated until the final and global model was achieved.

6. DECISION SUPPORT SYSTEMS USING THE KNOWLEDGE EXTRACTED FROM PETRI NETS

Previously, and after a continuous work developed around the case study, where all the system components were modeled and validated, and after understood the predicted communication and synchronization, the models were composed on an complete and global model that consists in having all the transport system connected. At this stage, it is necessary to start the extraction of knowledge from the Petri nets models to solve some of the problems that were presented before and some problems that may occur in production systems at real-time.

In manufacturing control processes, e.g. those based on service-oriented principles, decision-support systems are crucial to solve conflicts (e.g. which resource should perform the operation or which path should be taken by the pallet) and to recover from unexpected disturbances (e.g. a robot collision during its movement). These conflicts usually require external support by providing a concrete answer to solve the conflict, taking into consideration a set of criteria, e.g. productivity and efficiency.

The proposed approach to decision support systems considers the knowledge extracted from the structure of the Petri nets models, used to describe and execute the service-oriented process behaviors. Petri nets models contain richness knowledge about the process behavior, namely the description of service and device logics, and the available system's work cycles, besides some flexibility in terms of indirections and exceptions that are fundamental for decision systems. The decision to be chosen considers a multi-criteria function, being one of the decision topics the minimization of energy. A formal analysis will be performed, based on weighting T-invariants, according to theoretical hypothesis.

6.1. Decision-Making Based on Knowledge Extracted from High-Level Petri Nets

In flexible manufacturing environments it is normal the existence of conflict situations, since there are several alternatives to execute similar operations. Examples of conflicts are which workstation should perform a drill or which path should be taken to reach a specific workstation. In Petri nets models, conflicts are represented by places that have multiple output arcs going to different transitions. These situations should be handled by decision support systems that will provide services to support the evolution of the system. The complexity of the decision support system is strongly dependent on the flexibility that the system reveals [21].

When a decision node is detected, decision-making services should provide real-time support to the logic control engine. The proposed approach to decision support systems in service-oriented systems uses the powerfulness of the mathematical foundation of Petri nets. In fact, at any state of the system (represented by the current marking of the High-level Petri nets model), past, current and future discrete states of resources (individually or composed services) are known and ready to be used for monitoring and other supervisory control functions.

The basic idea is to consider the knowledge extracted from the High-level Petri nets model's state and the possibilities for evolution to new states, and combine them according to a weighted set of criteria to decide the best alternative solution. For this purpose, the following information is considered as basis for the decision process:

- The workplan and the operation to be executed.
- The process behavior and the structure of the model, i.e. the incidence matrix in the Petri nets terminology.
- The current state of the system, i.e. the net marking.
- The several sequences of operations, i.e. the T-invariants in the Petri nets; they can be determined for example applying the algorithm described in [22].

- The information of each resource related to several parameters, e.g. efficiency and energy consumption.

The first step in the proposed approach is the determination of the valid alternative solutions [21], see Figure 6-1:

- Crossing the current marking with the structure of the Petri net (provided by the incidence matrix), the several options to evolve are determined.

- Matching the achieved information with the set of T-invariants, by checking in each one the existence of both the conflict's transition and the transition that represents the service to be executed, it is determinate the set of valid alternative to execute the required service.

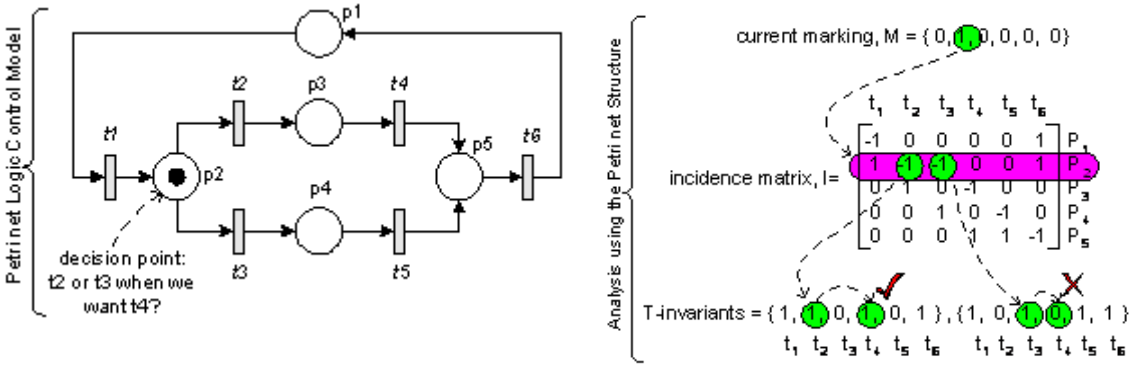


Figure 6-1- Determining the valid alternative solutions by using the knowledge extracted from the Petri nets structure [21].

At this stage, the problem is solved if only one alternative is valid to execute the service. As normally more than one alternative is possible, it is necessary to evaluate them through a decision function that may take into consideration several criteria. It is important to note that reducing the number of alternative sequences (i.e. the T-invariants), means less computer memory and faster processing analysis of the best solution.

The proposed approach for the decision function uses linear algebra operations to combine T-invariants with a vector of decision criteria, where different types of

decision parameters can be considered, including productive and energy efficiency related parameters.

In this method, the alternative solutions are represented by the set of T-invariants, $T_j = \{T_j \mid j \in N\}$, extracted from the Petri nets model. Each T-invariant of the Petri nets model is a vector,

$$T_j = [t_{j1} \quad t_{j2} \quad t_{j3} \quad \dots \quad t_{jn}]$$

where *n* denotes the number of transitions that defines the T-invarant.

A transition may represent a logic condition or a service (i.e. a time-consuming operation). The transitions representing services are associated to a vector,

$$t_i = [V_1 \quad V_2 \quad V_3 \quad \dots \quad V_c]$$

where *V* is the information (performance) of the service related to a specific criterion, e.g. the processing time and the energy consumption, and *c* is the number of criteria used in the decision set. It is needed to create a matrix of measured criterion to every transition in the system. These values can be constantly updated depending on the interest of the user and the achieved goal. Note that in case of transitions representing logic conditions, these values are null.

In the evaluation procedure it is crucial to define the set of decision criteria that will be applied to select the best option. Being a multi-criteria function it is necessary to define the weighting of each criterion for the final decision function, represented by the vector W:

$$W = [W_1 \quad W_2 \quad W_3 \quad \dots \quad W_c]$$

Now, that all the transitions have a criterion matrix associated and the user defined the weights that intended to associate to each criterion, it will be created a vector C where all criteria will be combined with his weight called E.U, and is given by:

$$C = [e.u_1, e.u_2, e.u_3 \dots e.u_n] \text{ and } e.u = W^T * t_j;$$

This matrix C will provide the values that will be associated to the transitions represented in Figure 6-2.

The decision multi-criteria function that evaluates a specific sequence of transitions (i.e. an alternative solution) is then given by:

$$f(j)=C*T^{T}$$

The value *f(j)* represents the evaluation score of the sequence path represented by the *j* T-invariant, considering the set of criteria defined initially and the weighting of each criterion. An important issue to be considered when dealing with a multi-criteria function, is related to normalize the dimensions of the different criteria; only using normalization it is possible to compare different things.

Applying the multi-criteria function to all T-invariants that represent acceptable alternatives, the vector f represents the evaluation score achieved by each alternative solution.

$$f = [f(1) \quad f(2) \quad f(3) \quad \dots \quad f(j)]$$

The set of alternative solutions can be ranked, being the best solution that which have a minimal value.

6.2. Decision Criteria for Manufacturing Systems Energy Awareness

In the evaluation of alternative solutions using the described approach, several criteria can be used. Traditionally, decision-making systems consider some productive related parameters, such as:

Parameter	Description
Cost	Cost to execute the operation by a device.
Processing time	Time necessary to execute the operation.
Location	Distance to the local where the device that will perform the next operation is placed.
Resource utilization	Percentage of utilization of a device.
Confidence degree	Level of trust that the device had granted during its historic operation.
Quality of service	Quality with which the operation is performed.
Maintenance	Costs with repairs and lubrication.

Since nowadays, the issues related to energy efficiency are in the order of the day, an important criterion to be used by such decision support systems is the minimization of energy consumption. In fact, manufacturing is traditionally an energy-intensive industry, using motors, steam, and compressed air systems to transform raw materials into durable goods and consumer products. Adopting more energy-efficient technologies and applying best practices in the control of manufacturing processes, the energy consumed in the factory plant can be significantly reduced and consequently improve the energy efficiency, reduce CO_2 emissions and enhance the fuel flexibility.

Motivated by the importance of energy-aware topics in manufacturing systems, the set of criterion to be used in the decision-making function aims to optimize the manufacturing processes, but also to reduce the energy consumption in the manufacturing devices. In terms of services, it means the selection of the best available service from a set of options that represents the requester's demands (in this case, in terms of process and energy efficiency).

So, aiming to consider the energy efficiency in the decision-making process, several energy-related parameters should be considered. In this work, several examples of energy parameters are identified and briefly characterized, as illustrated in the following table.

Parameter	**Description**
Energy consumed during the execution of an operation	Reflects the energy consumption by a device to perform a specific operation.
Energy consumed during transportation	Reflects the energy consumed during the transportation of materials by the shop floor devices.
Energy consumed for the set-up	Reflects the amount of energy that it is necessary to setup a device.

The determination of the energy consumption for each device requires the monitoring of each one, for example using proper software packages available on the market. These values are registered over the time, and after statistical treatment they can be used by the decision support system as a decision criterion (the historic data should be statistically treated to get more reliable values). The application of learning

methods can be useful to learn and predict new patterns based in the device historical data.

6.3. Application to the Case Study

The proposed decision-making based on the knowledge extracted from the Petri nets models works in service-oriented systems, considering amongst others, the minimization of the energy consumption, was tested in the case study used along this document, i.e. the FlexLink Demonstrator described in chapter 3.

The control of the system is defined in the Petri nets model, illustrated in Figure 6-2, which shows the global behavior in the different operation modes, exposed as services.

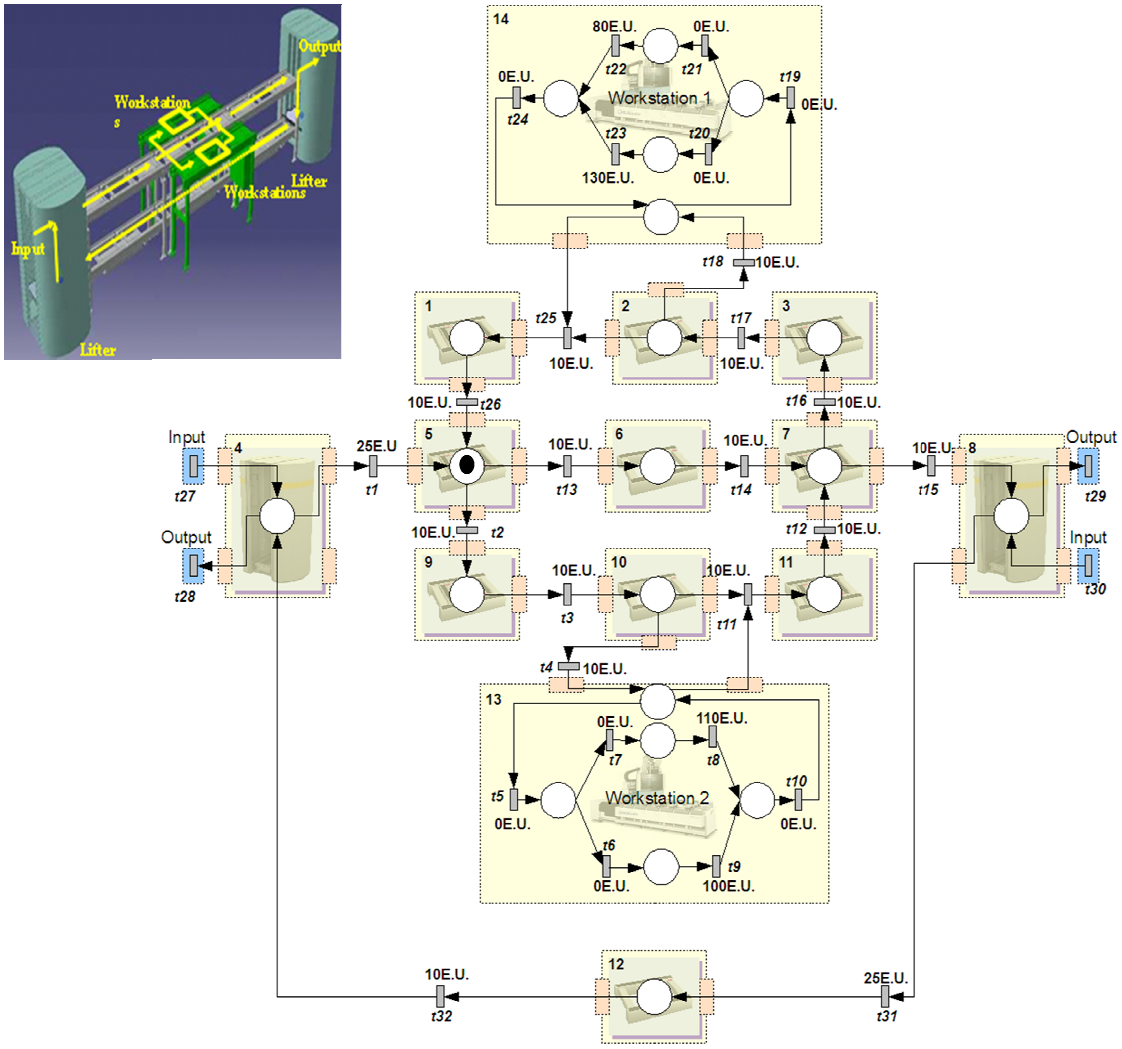


Figure 6-2-Petri net model for the case study.

In the above example, there are two workstations (represented by the transitions t_9 and t_{23}) with similar operations but with different energy consumption indexes. The knowledge extracted from the structure of the Petri nets model is the incidence matrix and the set of T-invariants. This Petri nets model has several T-invariants, but combining them with the workplan, only 3 are valid to execute the expected operation (i.e. in the workstation #1 or #2). They have the following physical meaning:

- $T_1 = \{t_2, t_3, t_4, t_5, t_6, t_9, t_{10}, ...\}$, representing the work cycle that conveys the pallet to the right and contains the workstation #2 (i.e. the transition t_9).
- $T_2 = \{t_{13}, t_{14}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{23}, t_{24}, ...\}$, representing the work cycle that conveys the pallet forward and contains the workstation #1 (i.e. the transition t_{23}).
- $T_3 = \{t_2, t_3, t_{11}, t_{12}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{23}, t_{24}, ...\}$, representing the work cycle that conveys the pallet to the right and contains the workstation #1 (i.e. the transition t_{23}).

Having more than one valid alternative to perform the operation, the decision support system will analyze these 3 T-invariants according to a set of decision criteria.

The decision procedure evaluates the set of T-invariants (i.e. the service invariants) according to pre-defined decision criteria, e.g. productivity, resource utilization or energy.

In a first scenario, only the minimization of energy consumption criterion is considered. The values of energy consumed by the resources to execute the operations are represented in Figure 6-2. Namely, transfer operations performed by conveyor devices spent 10 e.u. per operation (see e.g. transitions t_2 and t_3), and the processing operations performed by workstations spent more energy (100 and 110 e.u. for workstation #2 and 80 and 130 e.u. for workstation #1).

Applying the decision function proposed in the previous section, and considering the energy consumption criterion, the achieved results are,

$$f_{1st\text{-}scenario} = [130 \quad 180 \quad 220]$$

From the previous analysis it is clear that the better solution to execute the desired operation, and taking into consideration only the energy criterion, is the one specified by the T-invariant T_1, i.e. conveying the pallet to the right and using the workstation #2.

A second scenario considers the energy consumption and the processing time parameters, with equal criteria weight. Here, the decision will be taken considering the need to achieve fast processing and also regarding the energy consumption. The processing times assumed in this scenario are 2 e.u. for transportation operations, 70 t.u. for the workstation #1 and 130 e.u. for the workstation #2.

Applying the proposed evaluation function, the achieved results are,

$$f_{2nd\text{-}scenario} = [266 \quad 260 \quad 284]$$

Now, the selected solution is given by the T-invariant T_2 that considers the workstation #1. It is possible to observe that this is only possible because this workstation has a significantly shorter processing time when compared with the workstation #2.

The experimental results shows the flexibility of the decision support system, namely due to the knowledge extracted from the process behavior model (i.e. Petri nets models) and to the parameterization of the decision criteria that allows adjusting dynamically the decision-making. Note that the parameterization of the decision criteria is dependent of the system objectives but also strongly dependent of the learning mechanisms embodied in the decision support system.

These very simple scenarios, applied to the case study, can be easily improved and extended to big systems.

7. FUTURE WORK AND CONCLUSIONS

During this work, several tasks were developed, with the final objective of extraction of knowledge from Petri nets models that were used to design distributed applications in service-oriented automation systems. Initially, it was studied the FlexLink Demonstrator, which constitutes the experimental case study to apply the Petri nets models; particularly, it was studied all the movements and functions that it was intended for a perfect interaction between the intervenient.

After this initial study, it was developed the Petri nets models for the individual components of the overall system, preparing them for the installation of functions and to be composed in a global system's model. Before the models be composed it was necessary to do the assessment of them, checking if all the models are absent of errors, deadlocks, fulfilling the system specifications. The required knowledge, at this time, was understood as the communication and synchronization between the models, to be able to choose the type of composition that we will use. After having completed the modeling of the system, the information that we were capable to extract from Petri nets models was used to create a decision-making system, which considers energy efficiency parameters.

At the end of this work, it was identified several issues that could be further researched as well as new and promising windows related with the potentiality of combining the Petri nets formalism with the Service Oriented Architectures paradigms.

The designed Petri nets models, as described during this document, fulfill the specifications of the case study scenario. However, they need to be slightly improved, mainly introducing more logic, which will be an easy task since they are already prepared to accommodate them. The Continuum Development Studio had revealed a good tool to edit and analyze Petri nets models, being necessary to be improved especially to support efficiently the analysis of huge Petri nets models. The Petri net Composer tool, part of the Continuum Development Studio and developed and tested during this work, presented some problems in the visualization of the composed Petri

nets models and can be simplified when the user needs to define the ports and the XML file could be generated automatically.

In the decision-making system using the extracted knowledge from Petri nets models, some problems need to be solved. Namely, currently this system can be considered as a myopic system, being necessary to have a wider view to be completely efficient, the criteria parameters need to be normalized to allow combining several types of different data.

Other adjacent aspects that need to be explored is the definition of Petri nets-based control models representing process ontology, the meaning of the information obtained from the analysis of those models over the synthetic and semantic way, and the definition of the additional properties of the Petri nets

Another future perspective for this work is the application of the results and methods to other fields, such as medicals management in hospital environments and airports control.

In conclusion, this work can be seen as a success, because most of the described steps were useful to the SOCRADES project and were documented on Deliverable D5.2.2, untitled “Model-based Orchestration Engine integrated in SoA architecture" and annexed document "User Guide & Release Notes". And it was proved the potentiality of the Petri nets when combined with Service-oriented Architectures, namely extracting knowledge to support monitoring and decision-making.

8. REFERENCES

[1] L. Ribeiro, J. Barata and W. Colombo, “MAS and SoA: A Case Study Exploring Principles and Technologies to Support Self-Properties in Assembly Systems”, Second IEEE International Conference on, Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2008.

[2] Li Gang, Li Yongqiang, S. Linyan, Ji Ping, “Multi-agent Coordination Schemas in Decentralized Production Systems”, Learning Systems and Multi-agents, vol. 4456, pp 347-356, 2007

[3] M. Wooldridge, “An Introduction to Multi-Agent Systems”, John Wiley & Sons, 2002.

[4] N. R. Jennings and M. Wooldridge, “Applications of Intelligent Agents”, Agent Technology, Berlin 1998, pp 3-28.

[5] R. Shoop, R. Neubert and A.W. Colombo “A MultiAgent-based Distributed Control Platform for Industrial Flexible Productions Systems”, IEEE 27th conference in Industrial Electronics Society, IECON 2001.

[6] R. Harrison, A. Colombo, A. West, and S. Lee, "Reconfigurable modular automation systems for automotive power-train manufacture," International Journal of Flexible Manufacturing Systems, vol. 18, 2006, p. 175.

[7] P. Leitão, F. Restivo and G. Putnik “A Multi-agent based Cell Controller”, Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation, pp. 463-470 Antibes, 2001.

[8] F. Jammes and H. Smit, “Service-Oriented Paradigms in Industrial Automation”, 23rd IASRED International Multi-Conference on Parallel and Distributed Computing and Networks, Innsbruck Austria, 2005.

[9] T. Murata, "Petri nets: Properties, analysis and applications", *IEEE*, vol. 77, pp. 541-580, April 1989.

[10] R. Zurawski and MengChu Zhou, “Petri nets and industrial applications: A tutorial”, IEEE Transactions on Industrial Electronics**,** vol. 41, pp. 567-583, 1994.

[11] A. Desrochers and R. Al-Jaar, “Applications of Petri Nets in Manufacturing Systems-Modeling, Control and Performance Analysis”, IEEE Press, 1995.

[12] M. Silva and R. Valette, “Petri nets and flexible manufacturing”, In Advances in Petri nets, LNCS 424, Springer-Verlag London, pp. 374-417, 1990.

[13] P. Leitão, “An Agile and Adaptive Holonic Architecture for Manufacturing Control”, PhD Thesis, University of Porto, 2004.

[14] J. Ferber, “Multi-Agent Systems”, An Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999.

[15] M. Silva, “Las Redes de Petri en la Automática y la Informática”, Editorial AC, 1985.

[16] K. Feldmann and A.W. Colombo, “Monitoring of Flexible Production Systems Using High-Level Petri Nets”, *Control Engineering Practice*, n. 7, pp. 1449-1466, 1999.

[17] J.M. Mendes, A. Bepperling, J. Pinto, P. Leitão, F. Restivo and A.W. Colombo, “Software Methodologies for the Engineering of Service-oriented Industrial Automation: The Continuum Project”, Proceedings of the 33rd Annual IEEE International Conference on Computer Software and Applications (COMPSAC’09), Seattle, Washington, 20-24 July, 2009, pp. 452-459.

[18] J. Lastra, "Reference Mechatronic Architecture for Actor-Based Assembly Systems," Doctoral Dissertation, Tampere University of Technology, 2004.

[19] R. Harrison, A. Colombo, A. West, and S. Lee, "Reconfigurable modular automation systems for automotive power-train manufacture," International Journal of Flexible Manufacturing Systems, vol. 18, 2006, p. 175.

[20] P. Leitão, A. W. Colombo and F. Restivo, “Formal Specification of ADACOR Holonic Control System: Coordination Models”, Proceedings of the 44th IEEE International Conference on Decision and Control (CDC’05), Seville, Spain, 12-15 December, 2005, pp. 2137-2142.

[21] J. Martinez and M. Silva, “A Simple and Fast Algorithm to Obtain All Invariants of a Generalized Petri net”, In C. Girault and W. Reisig (eds.) Fachberichte Informatik, vol.52, Springer Verlag , pp. 301-310, 1982.

[22] P. Leitão, J. Alves, J.M. Mendes, A. W. Colombo, “Energy Aware Knowledge Extraction from Petri Nets Supporting the Decision-Making in Service-oriented Systems”, to be submitted to the IEEE International Symposium on Industrial Electronics (ISIE’10), Bari, Italy, 4-7 July 2010.

Appendix A

This appendix intends to compile the designed individual Petri nets models, and the illustration of using the PNC tool by describing an example step-by-step. At the end it will be presented the complete Petri nets model for the experimental case study.

A.1- Individual Models

The following figures illustrate the designed Petri nets models for the components belonging to the case study.

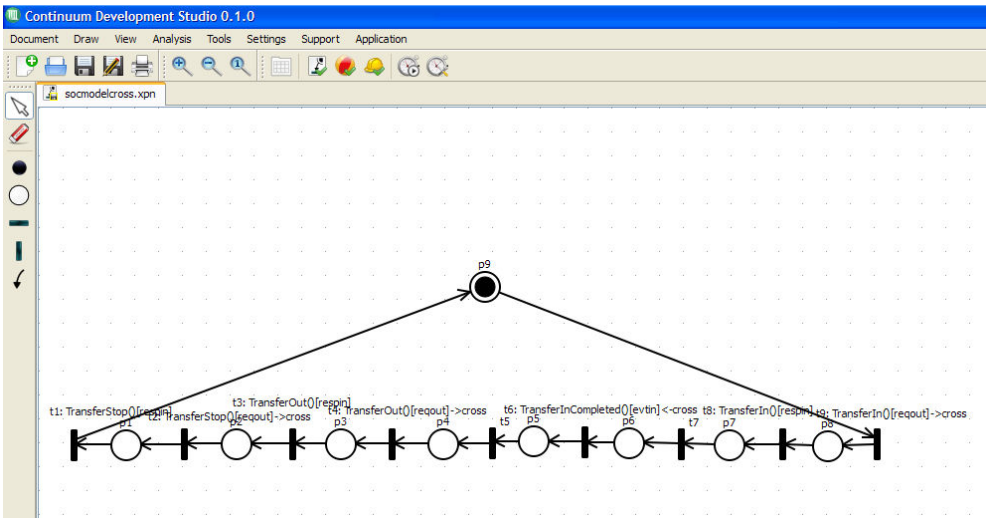


Figure 8-1- Model of the unidirectional conveyer

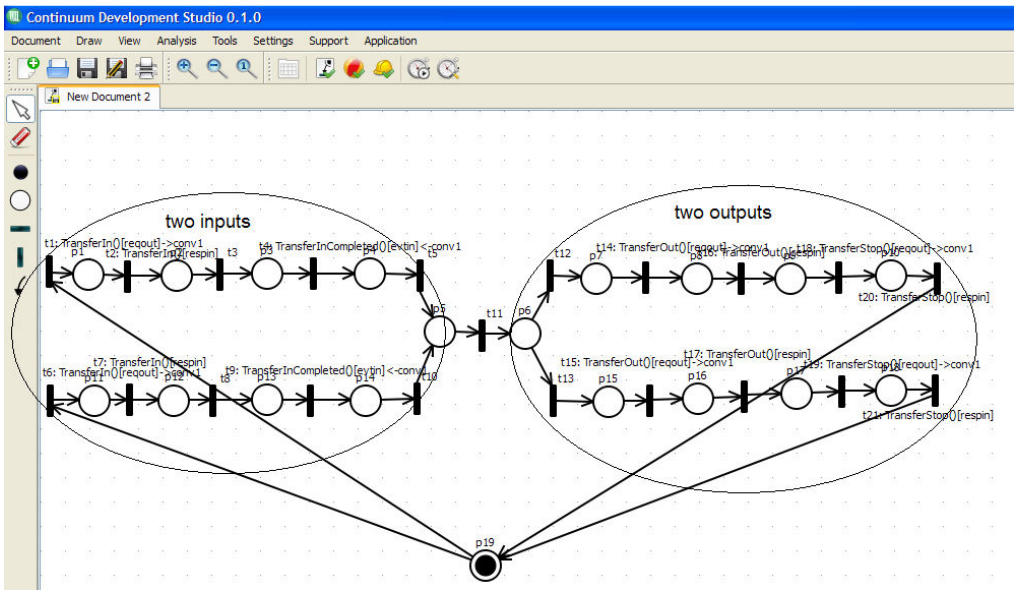


Figure 8-2-Model of cross unity

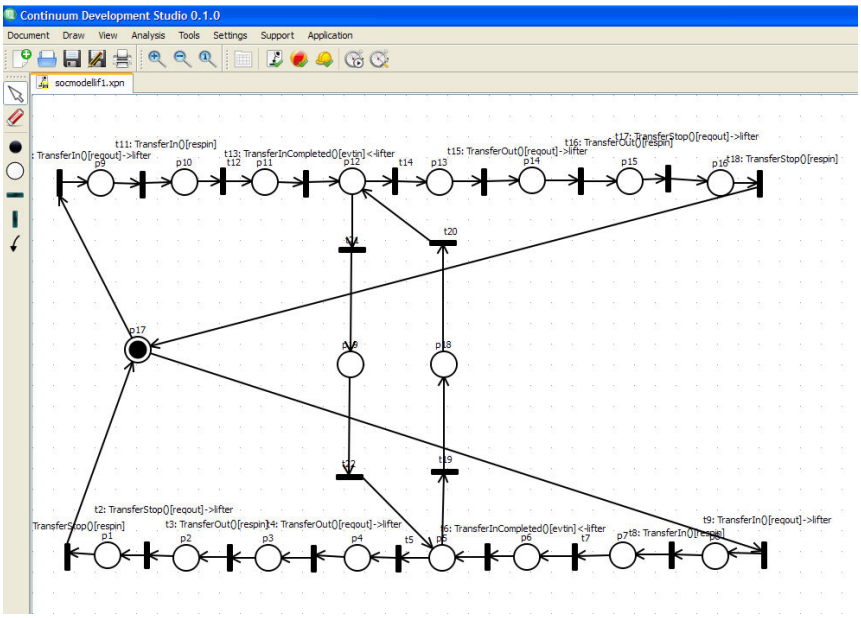


Figure 8-3- Model of the lifter

A.2- Step-By-Step with PNC

In this section, a step-by-step example of a composition between two unidirectional conveyers, using the PNC tool, will be presented.

1) Give a name to the model;

1.1) **Select** the page layout;

1.2) **Add** a property on the Property editor, with **property name** resource, **data** type string and **value** “name of the model”;

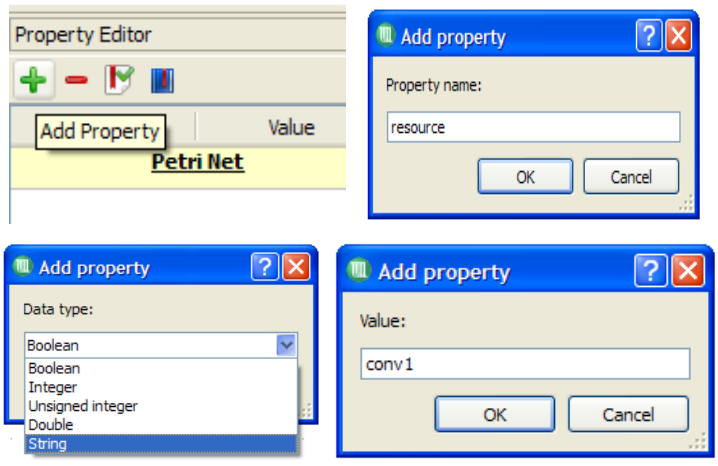


Figure 8-4- Add property name resource

2) After building the model, the user must define the ports of the connection. In the following example, considering an unidirectional conveyer model with the transfer in and transfer out areas, the user must define what transitions belong to each port. Port1 of Figure 8-5 includes the transitions that belong to the transfer out operation and port2 includes the transitions that belong to transfer in operations of the conveyor.

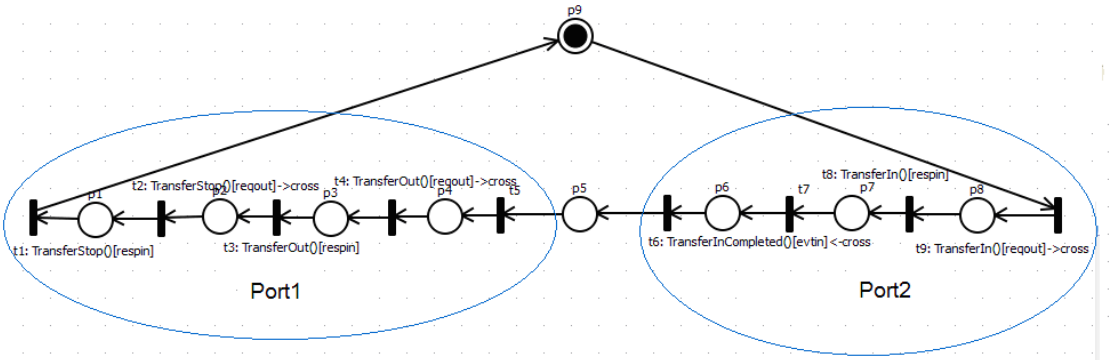


Figure 8-5- Definition of ports

3) With the ports defined, the user should know which are the transitions of the port that will be connected with others models. In those transitions he has to add a property with the name of the port that she belong;

   3.1) **Select** the transition;

   3.2) **Add** a property on the Property editor, with **property name** port, **data type** string and **value** “name of the port which that the transition belongs”;

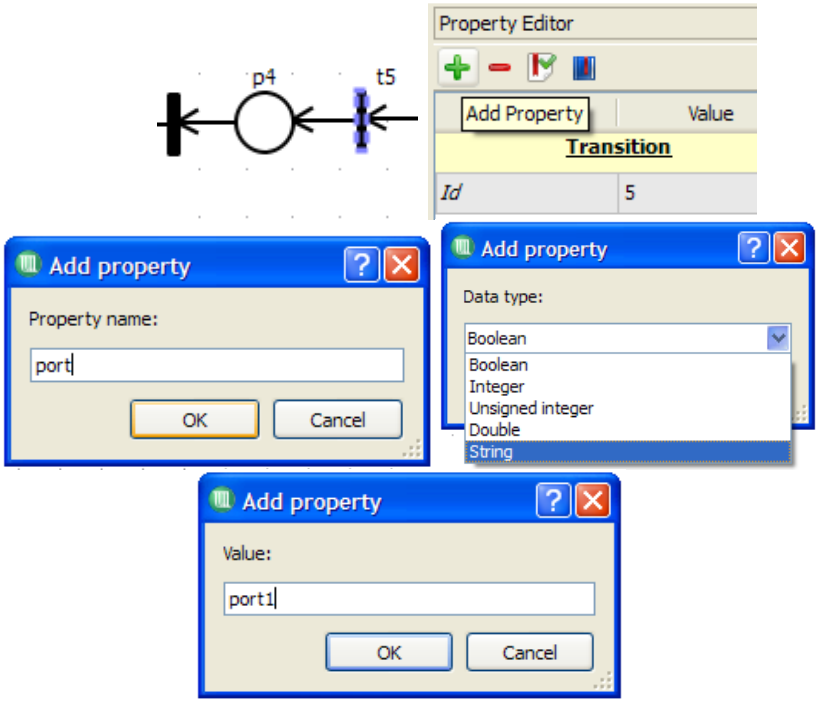


Figure 8-6- Add property name of the port

4) After all the transitions are defined, to which port belong, the user should create a new property. On this property, will be defined if the transition will receive or send a token and the position of the transition in the port. When a transition receives a token the property name has to be a port_in_seq and when it is sending a token has to be a port_out_seq. A port_in_seq of one model is always connected to a port_out_seq of the other model. The connection between the ports is made by the value of this property, i.e. number one of one port connects with number one of the another model, so the user has to be careful with this details.

   4.1) **Select** the transition;

   4.2) **Add** a property on the Property editor, with **property name** port_out_seq, **data type** string and **value** “number of the transition on the port”;

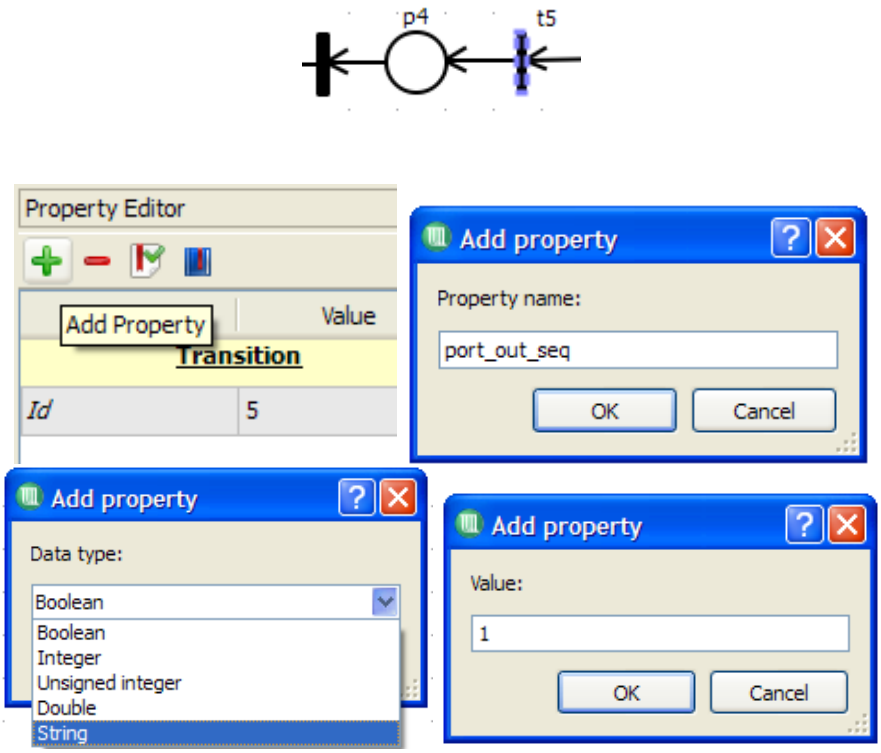


Figure 8-7- Add property sequence and number of the port

5) Now that all the ports and transitions are well defined, the user has to create one xml file. This file will contain the information of the resource and port, and say which resource connects with other and by which port. To create this file it can be used Notepad, EditPlus, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<connections>
    <connection resource1="conv1" port1="port1"
          resource2="conv2" port2="port1"/>
</connections>
```

What this piece of code do, is to define that the resource1 with the name conv1 is connected by port1 with the resource conv2 by port1.

6) Now in the CDS, it will be created a file with the PNC.

6.1) **Open** the PNC tool;

6.2) **Select** the xml file created on step 5;

6.3) Then the tool will ask what xnp files you want to join in the final model, user can select all the files in one time or one at the time. The tool always asks if you want to join another file, when all the files are selected, answer **no**. And afterwards, will be open a save option where the user give a name to the final model.

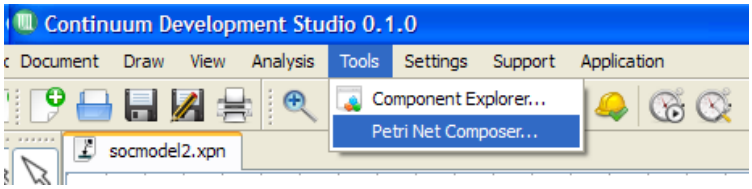


Figure 8-8- Open the PNC tool

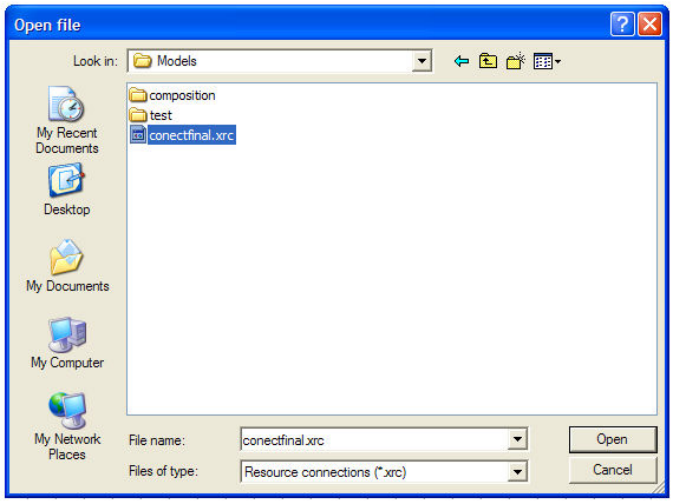


Figure 8-9- Select xml file

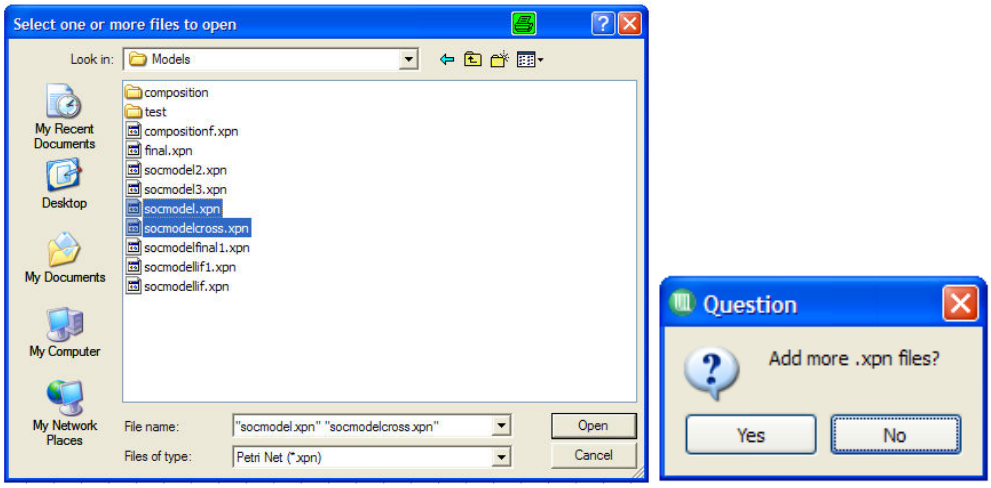


Figure 8-10- Add models to the final model

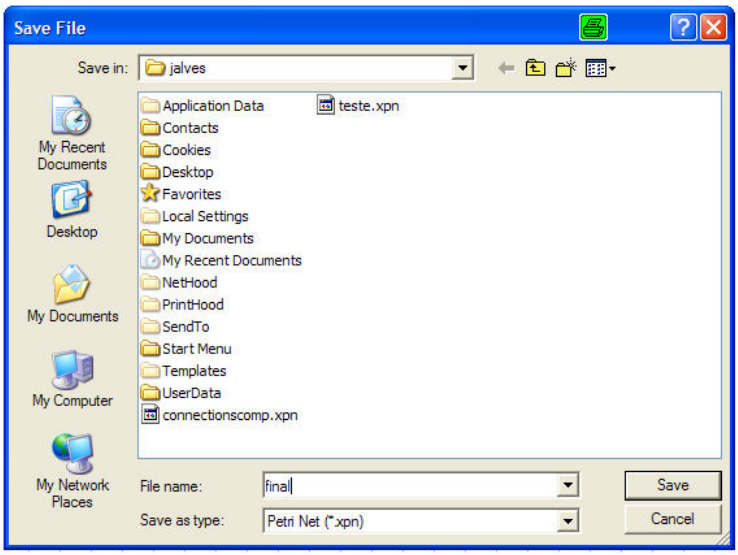


Figure 8-11- Saving the final model already generated

7) Open the file created in the final step with CDS and all the connections described by the user will be generated.

A.3- Complete model of the Case Study

After composing the individual models, a unique model is achieved and represented in the Figure bellow.

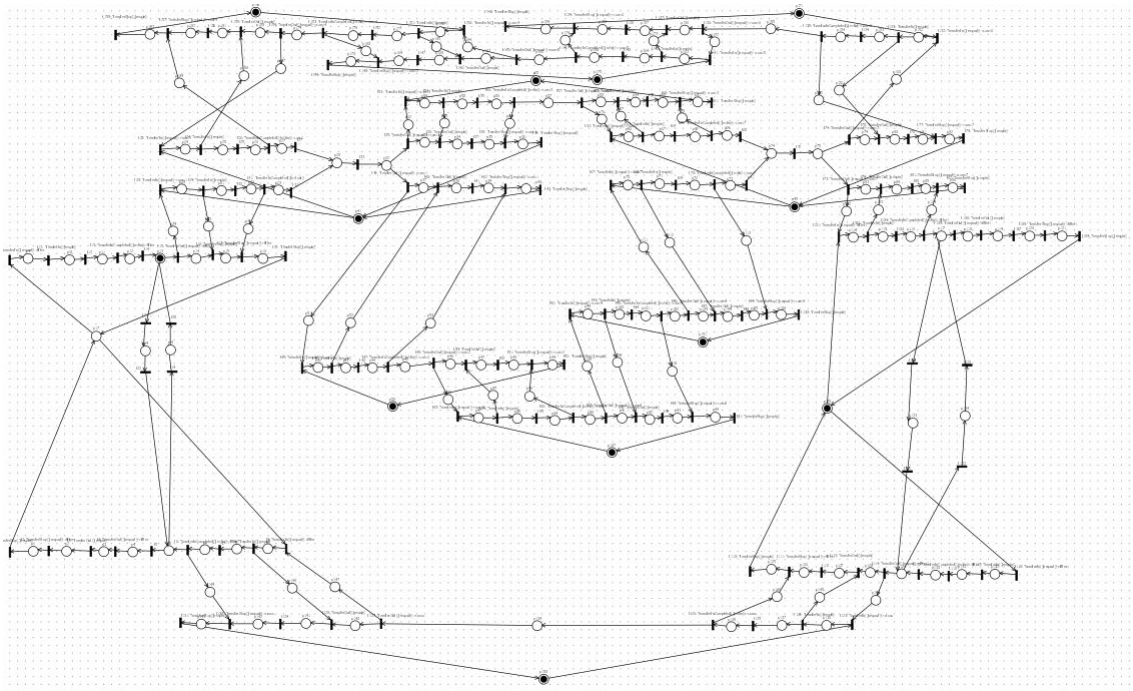


Figure 8-12- Complete model of the Case Study