Chapter 2

Specifying and Verifying Holonic Multi-Agent Systems Using Stochastic Petri Net and Object-Z: Application to Industrial Maintenance Organizations

Belhassen Mazigh and Abdeljalil Abbas-Turki

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50229

1. Introduction

In Industrial Maintenance Company (IMC) a vast number of entities interact and the global behaviour of this system is made of several emergent phenomena resulting from these interactions. The characteristics of this system have increased both in size and complexity and are expected to be distributed, open and highly dynamic. Multi-Agent Systems (MAS) are well adapted to handle this type of systems. Indeed, the agent abstraction facilitates the conception and analysis of distributed microscopic models [9]. Using any holonic perspective, the designer can model a system with entities of different granularities. It is then possible to recursively model subcomponents of a complex system until the requested tasks are manageable by atomic easy-to-implement entities. In multi-agent systems, the vision of holons is someway closer to the one that MAS researchers have of Recursive or Composed agents. A holon constitutes a way to gather local and global, individual and collective points of view. A holon is a self-similar structure composed of holons as sub-structures. A hierarchical structure composed of holons is called a holarchy. A holon can be seen, depending on the level of observation, either as an autonomous atomic entity or as an organisation of holons (this is often called the Janus effect [12]). Holonic systems have already been used to model a wide range of systems, manufacturing systems [15, 16, 33], health organizations [32], transportation [2], etc. The different organisations which make up an IMC must collaborate in order to find and put in place various strategies to maintain different production sites. In order to honour its contracts, the IMC should handle the whole of its resources (human and material), ensure the follow up in real time the equipment in different production sites and plan actions to be executed. A part of the maintenance could be remotely achieved (tele-maintenance and/or tele-assistance [17], e-maintenance [13], etc.). Several constraints should be integrated in the process of strategy search and decision making before mobilizing operation teams. Concretely, the search for an efficient maintenance strategy should be

found while taking into account the following constraints: (a) The urgency level of the maintenance task requested by a given production site, (b) A distance between mobile teams and production sites, (c) The estimated intervention duration, (d) The respect of the legal daily working time of maintenance crew, (e) Verification of the availability of the stored spare parts, (f) The reconstitution of new teams in view of the mentioned constraints.

To satisfy some of these constraints, we propose a formal holonic approach for modelling and analysis all the entities that constitutes an IMC. We use Holonic Multi-Agent Systems (HMAS) in which holons are agents that may be composed of agents for developing complex systems. To this end, we use an Agent-oriented Software Process for Engineering Complex Systems called ASPECS [8]. The process is considered by their authors as an evolution of the PASSI [3] process for modelling HMAS and it also collects experiences about holon design coming from the RIO (Role, Interaction and Organization) approach [11]. It is sufficient to say that the definition of the MAS meta-model adopted by the new process has been the first step and from this element all the others (activities, guidelines and workflow) have been built according to this guideline [4, 30, 31]. This meta-model defines the underlying concepts. A step-by-step guide from requirements to code allows the modelling of a system at different levels of details. Going from each level to the next consists in a refinement of the meta-model concepts. The objective of this work consists in consolidating the ASPECS methodology by using a formal specification and analysis of the various organizations and the interactions between them. This phase will facilitate the code production of organizations, roles and holons. In addition, it will be possible to test each organization, their roles and each holons independently. This type of analysis, will allow checking certain qualitative properties such as invariants and deadlock, as well as a quantitative analysis to measure the indicators of performance (cost of maintenances, average duration of the interventions, average time to reach a site of production, etc). In this chapter, our extended approach will be used to model and analyze an Industrial Maintenance Company (IMC). After a brief presentation of the framework, the maintenance activities in a distributed context are presented. ASPECS process and modelling approach will be introduced in section 3. Analysis and conception phase of this process and their associated activities are then described in order to identify the holonic IMC organisation. In section 4, first we present our specification formalism and second we assign an operational semantics to it. Additionally, we illustrate how to use the operational semantics as a basis for verification purposes. The specification formalism we intend to present combines two formal languages: Stochastic Petri Nets and Object-Z. Finally, Section 5 summarises the results of the chapter and describes some future work directions.

2. Industrial maintenance company distributed context

For economic and/or efficacy reasons, many companies are being implanted in geographically wide spread areas. Hence, many IMC are compelled to represent their organisations so as to meet efficiently the demands of their clients. Therefore this activity is among those witnessing a rise on the word market in spite of the economic moroseness (particularly large scale and multiple competence maintenance companies). In this distributed context, the maintenance activities are divided on two following structures: (a) Central Maintenance Teams (CMT) which realizes the process of reparation - corrective maintenance, (b) Mobile Maintenance Teams (MMT) which carries out inspections, replacement and several other actions on the various production sites.

To ensure the maintenance of several production sites, many teams specialized in various competence fields should be mobilized. Those in charge of handling these resources should

overcome complex logistical problems thereby the need to develop aiding methods and tools for decision making to efficiently manage this type of organizations. Among the services proposed by an IMC, we can mention: (a) On site intervention, (b) Technical assistance via telephone or distance intervention, (c) The study and improvement of equipment, (d) Organisation and engineering, (e) Formation in industrial maintenance, (f) The search for and the development of new maintenance methods. The focus in this research is to find an efficient maintenance policies taking into account multiples alias. We propose a formal approach, based on the paradigm HMAS, to specify and analyze an efficient and adaptive IMC. As such, a specification approach based on HMAS to be a promising approach to deal with the unpredictable request of maintenance due to their decentralization, autonomy, cooperation features and their hierarchical ability to react to unexpected situation. For this specification, we use ASPECS methodology to identify holonic organization of a steady system and we combine two formal languages: Object-Z and Petri Nets for modeling and analysis specification of an IMC.

3. A holonic specification approach of an IMC

3.1. A quick overview of ASPECS process

The ASPECS process structure is based on the Software Process Engineering Metamodel (SPEM) specification proposed by OMG [25]. This specification is based on the idea that a software development process is collaboration between abstract active entities, called Roles that perform operations, called Activities, on concrete, tangible entities, called Work Products. Such as it was proposed by [4], ASPECS is a step-by-step requirement to code software engineering process based on a metamodel, which defines the main concepts for the proposed HMAS analysis, design and development. The target scope for the proposed approach can be found in complex systems and especially hierarchical complex systems. The main vocation of ASPECS is towards the development of societies of holonic (as well as not-holonic) multi-agent systems. ASPECS has been built by adopting the Model Driven Architecture (MDA) [25]. In [5] they label the three meta-models "domains" thus maintaining the link with the PASSI meta-model. The three definite fields are: (a) The Problem Domain. It provides the organisational description of the problem independently of a specific solution. The concepts introduced in this domain are mainly used during the analysis phase and at the beginning of the design phase, (b) The Agency Domain. It introduces agent-related concepts and provides a description of the holonic, multi-agent solution resulting from a refinement of the Problem Domain elements, (c) The Solution Domain is related to the implementation of the solution on a specific platform. This domain is thus dependent on a particular implementation and deployment platform.

Our contribution will relate to the consolidation of the Problem Domain and the Agency Domain. We propose a formal specification approach for analysis the various organizations and the interactions between them facilitating therefore the *Solution Domain*.

3.2. Requirements analysis

The analysis phase needs to provide a complete description of the problem based on the abstractions defined in the metamodel problem domain CRIO (Capacity, Role, Interaction and Organization). All the activities that make up this first phase and their main products can be identified. Indeed, this phase shows the different steps that can be used for the requirements

since the description of the field requirements to capacities identification. It also shows how documents and UML diagrams must be constructed for each step. In the following, we present objective, description and diagrams for each step used in the requirements analysis phase.

3.2.1. Requirements domain description

The aim of this phase is to develop a first description of the application context and its functionalities. This activity aims to identify, classify and organize in hierarchy all functional and non functional requirements of different project actors. It must also provide a first estimated scope of the application as well as its size and complexity. In this work, the analysis approach adopted is based on the use case UML diagrams. To facilitate the analysis and reduce the complexity of the system studied, we decomposed the system studied into three modules: Mobile Maintenance Teams (MMT), Maintenance Policies (MP) and Maintenance Mediation (MM) which plays the role of mediator between the first two as shown in Figure 1. Let us now explain the role of each part of the system. The objective of "Mobile Maintenance

Figure 1. Set parts associated to the IMC

Teams" (MMT) is: (a) Receive maintenance requests from "Mediation Maintenance", (b) Planning for maintenance actions, (c) Execute maintenance tasks, (d) Generate maintenance report. The role of "Mediation Maintenance" (MM) is: (a) Receive maintenance requests of different production sites, (b) Diagnosing problems, (c) Classify problems, (d) Responding to customers, (e) Send request maintenance to MMT or MP, (f) Receive execution plans of maintenance from MT or MP if necessary. The objective of "Maintenance Policies" (MP) is: (a) Receive maintenance requests from "Mediation Maintenance", (b) Scheduling maintenance tasks, (c) Execute maintenance tasks, (d) Generate maintenance report, (e) Send report to maintenance "Mediation service", (f) Develop new maintenance methods, (g) Provide training maintenance.

As an example, we'll just show the use case diagram of different scenarios associated to the MMT (Figure 2.). Similarly, we can use identical approach to describe other roles and establish their use case diagrams.

3.2.2. Problem ontology description

First, problem ontology provides a definition of the application context and specific domain vocabulary. It aims to deeper understanding the problem, completing requirements analysis and use cases, with the description of the concepts that make up the problem domain and their relationships. Ontology plays a crucial role in ASPECS process development. Indeed, its structure will be a determining factor for identifying organizations. The ontology is described here in terms of concepts, actions and predicates. It is represented using a specific profile for UML class diagrams. The ontology of the IMC is described in Figure 3.

This ontology represents the knowledge related to the Mobile Maintenance Teams, Mediation Maintenance, Maintenance Policies, the different concepts that compose and the relationships between these components.

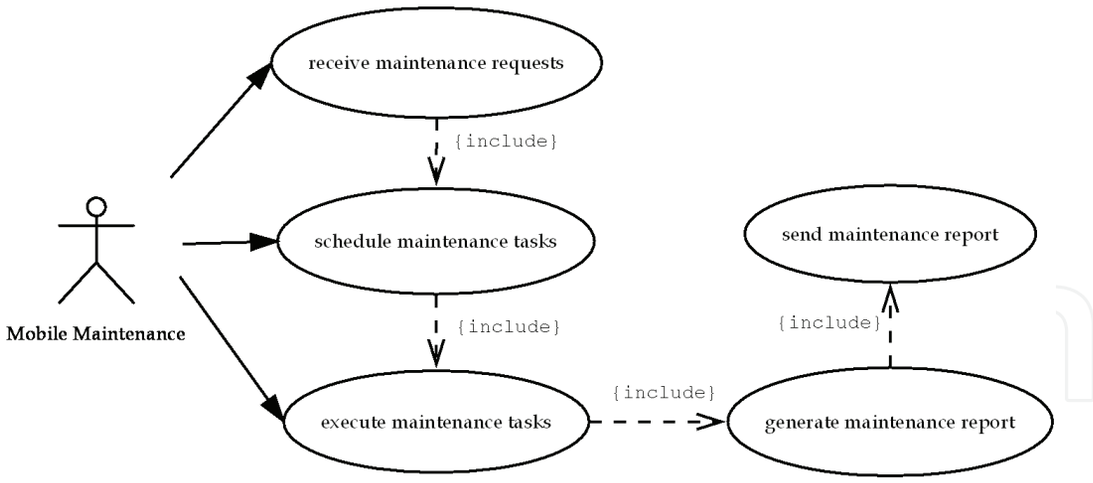

Figure 2. Use case diagram of Mobile Maintenance Teams

3.2.3. Identifying organizations

This action must establish a decomposition of the organizational system and define the objectives of each organization. Every needs identified in the first activity, has an associated organization incarnating the global behavior in charge to satisfy or to realize. Identified organizations are added directly to the use case diagram, in the form of packages including stereotyped use cases diagram that are responsible to satisfy. The Mobile Maintenance Teams organization can be decomposed into three sub-organizations: MMT Forwarding, Tasks Planning and Tasks Execution as shown in Figure 4.

The same process can be used to describe Mediation and Mobile Maintenance Teams organization. According to this decomposition, we can obtain organizational hierarchy of the IMC (Figure 5).

In this hierarchy, MMT Forwarder serves as an intermediary between Mediation Maintenance and Mobile Maintenance Teams. Similarly, CMT Forwarding is directly linked to both organizations Mediation Maintenance and Maintenance Policies.

3.2.4. Identification of roles and interactions

The context and objectives of each organization are now identified. The identification of roles and interactions aims to decompose the global behavior incarnated by an organization into a set of interacting roles. This activity must also describe the responsibilities of each role in satisfying the needs associated with their respective organizations. Each role is associated with a set of concepts in the ontology and generally a subset of those associated with his organization. Roles and interactions that constitute each organization are added to their class diagrams as described in Figure 6. A role is represented by a stereotyped class, and an interaction between two roles is represented by an association between classes of roles. Note also that in this figure, the link "Contributes to" means that an organization contributes in part to the behavior of a role at a higher level of abstraction.

3.2.5. Description of interaction scenarios

The objective of this activity is to specify the interactions between roles to induce higher level behavior. This activity describes the interactions between the roles defined within a given

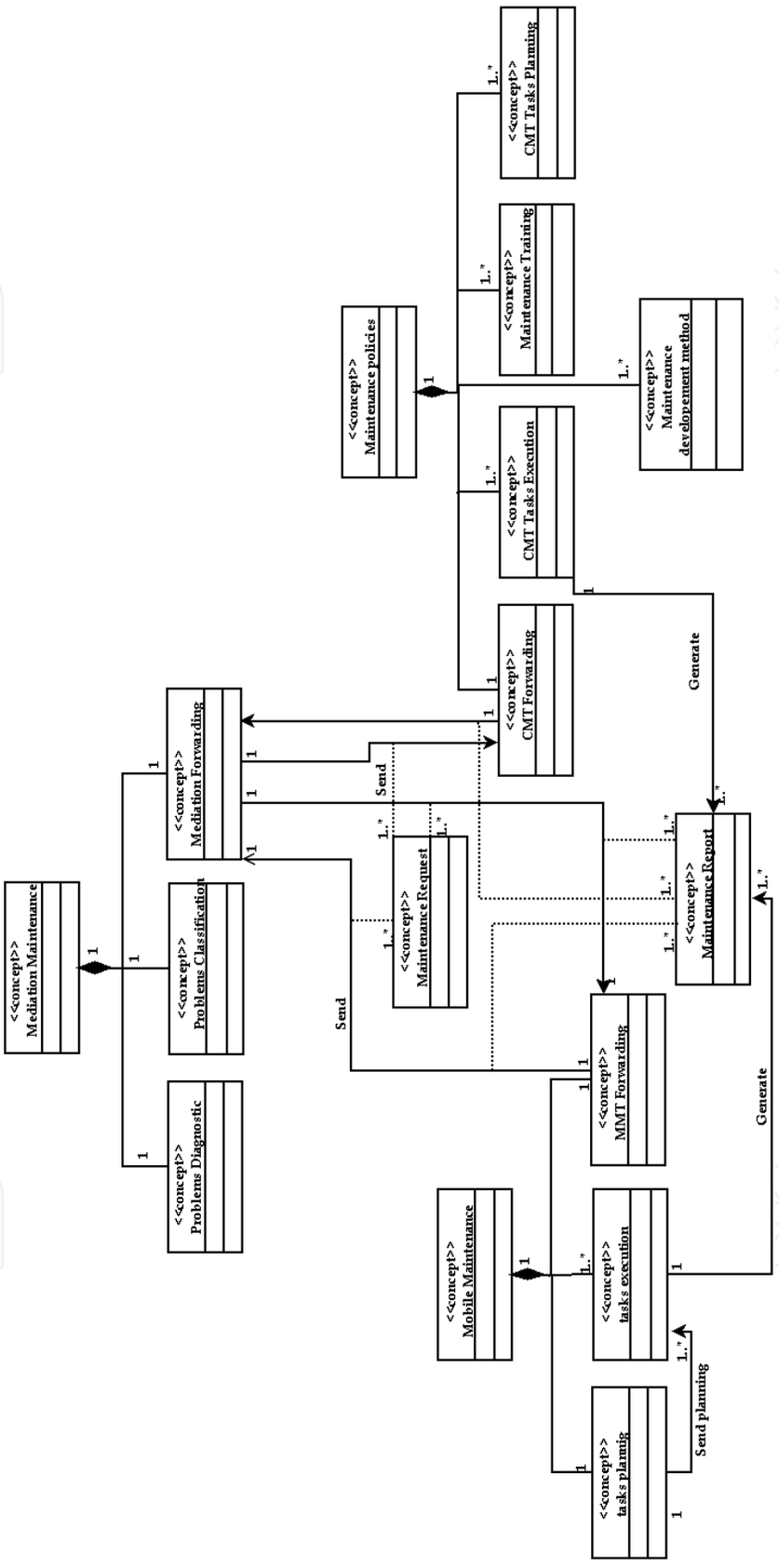

Figure 3. Ontology of the IMC system

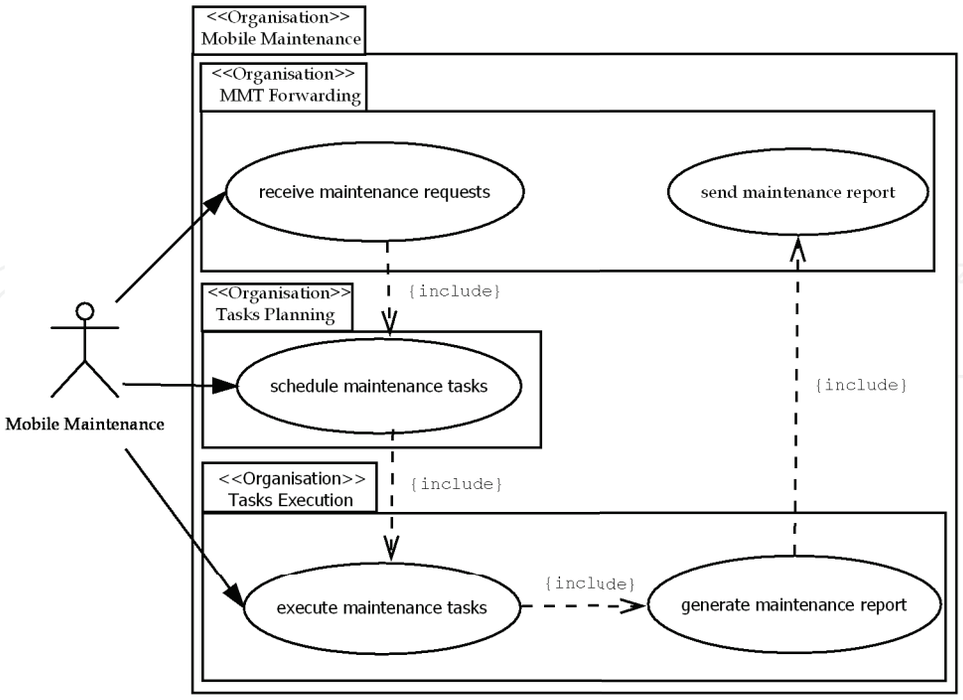

Figure 4. Mobile Maintenance Teams Organization

organization and specifies the means of coordination between them to meet the objectives of their organization. Consequently, each organization is associated with at least one scenario and it may involve defined roles in different organizations. Indeed, an organization usually requires information from other organizations at different level of abstraction. The scenarios detail the sequence of arrival or transfer of such information. Describing interaction scenarios is supported by a set of UML sequence diagrams. An example of interaction scenario associated with the organization IMC system is shown in Figure 7. This diagram above describes possible scenarios within the organization "Mobile Maintenance Teams". For each received request, "Planning Tasks" will schedule maintenance tasks and "Tasks Execution" executes maintenance tasks and finally generates a maintenance report. Scenarios of "Mediation Maintenance" and "Maintenance Policies" organization are not complicated and are not represented here.

3.2.6. Behavioral roles plans of organizations

The description of the behavior plans of roles specify the behavior of each role adapted with the objectives assigned to it and interactions in which it is implicated. Each plan describes the combination of behavior and sequencing of interactions, external events and tasks that make up the behavior of each role. Figure 8 shows the behavior plans of the roles that constitute the IMC organizations.

3.2.7. Identification of capacities

This activity aims to increase the generic behavior of roles by separate clearly the definition of these behaviors of their external dependencies organizations. It is particularly to refine the behavior of roles, to abstract the architecture of the entities that will play them, and ensuring

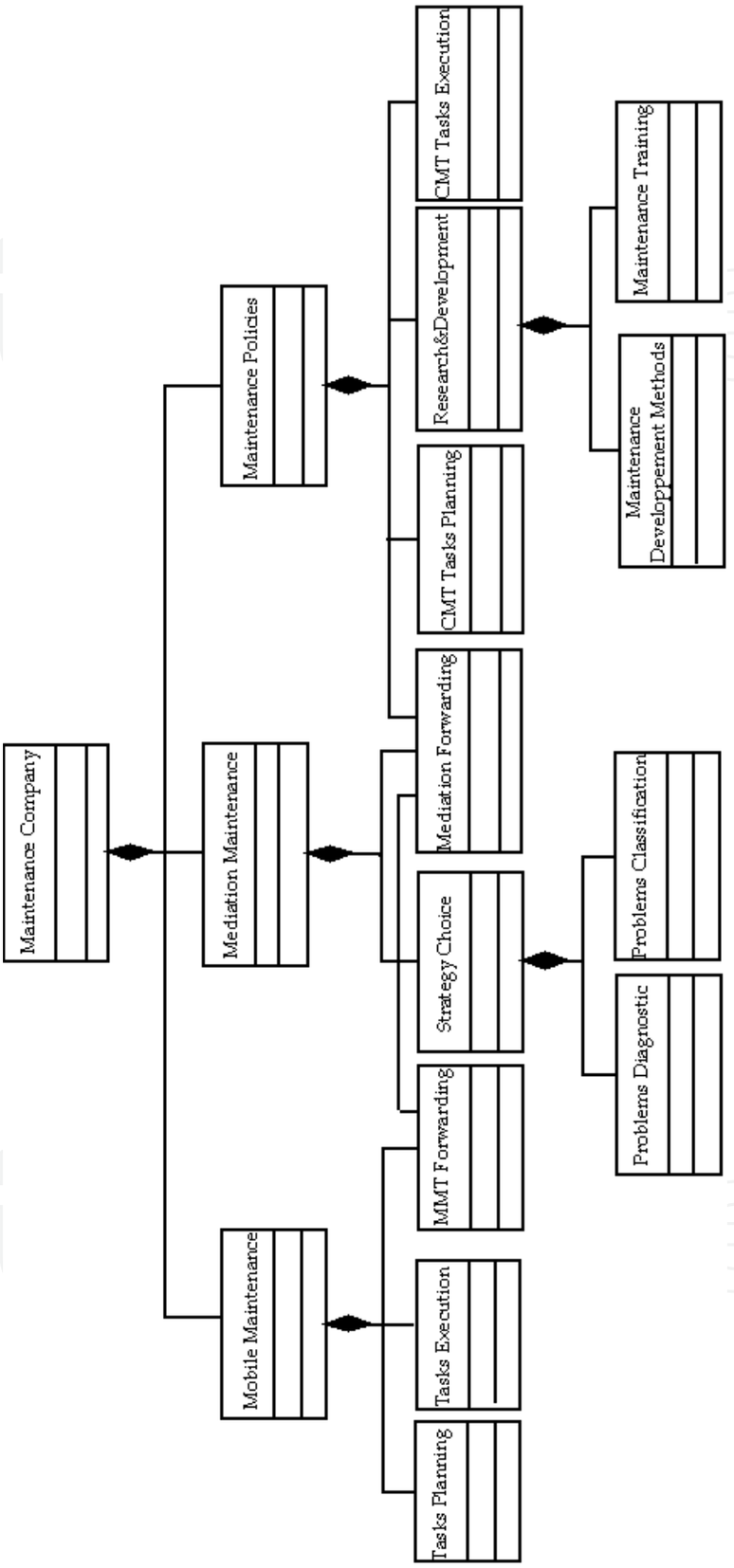

Figure 5. Organizational hierarchy of the IMC

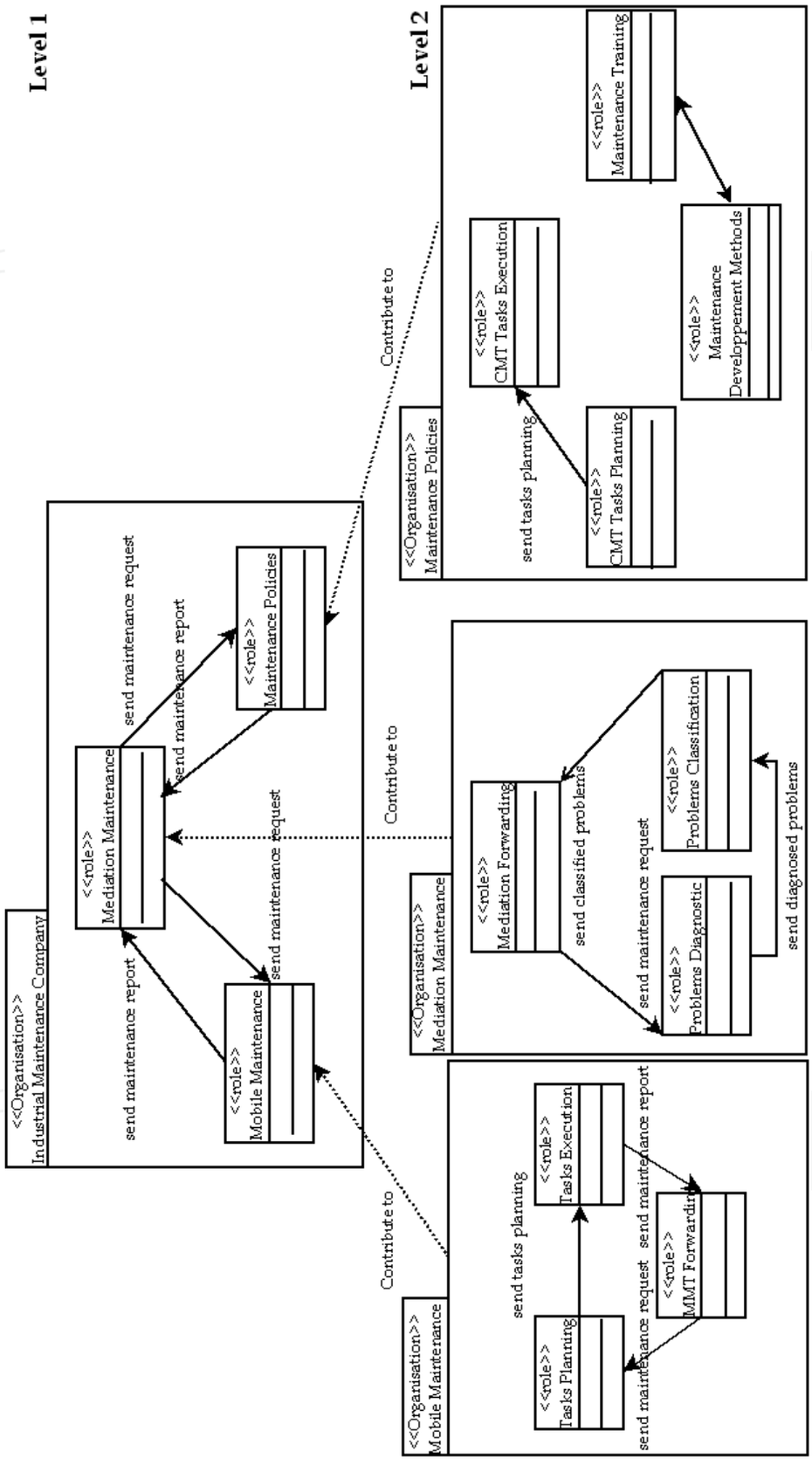

Figure 6. Description of a few roles and interactions of the IMC organization

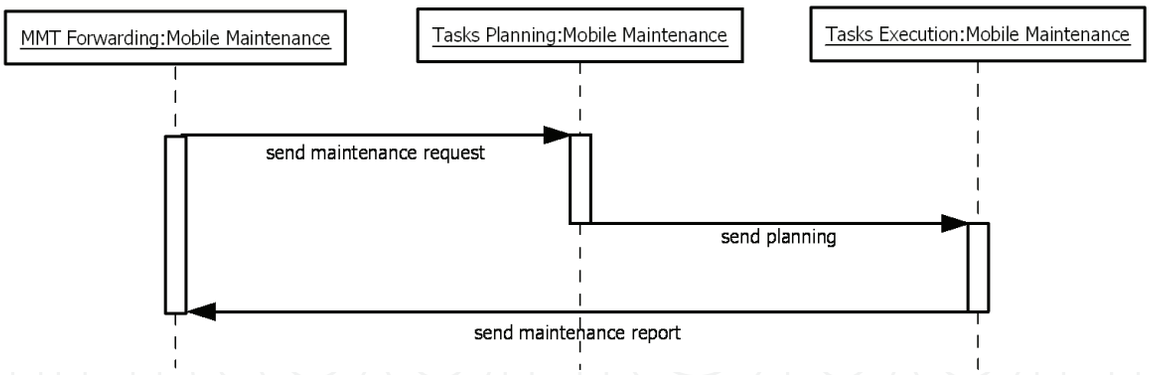

Figure 7. Description of interaction scenarios for MMT

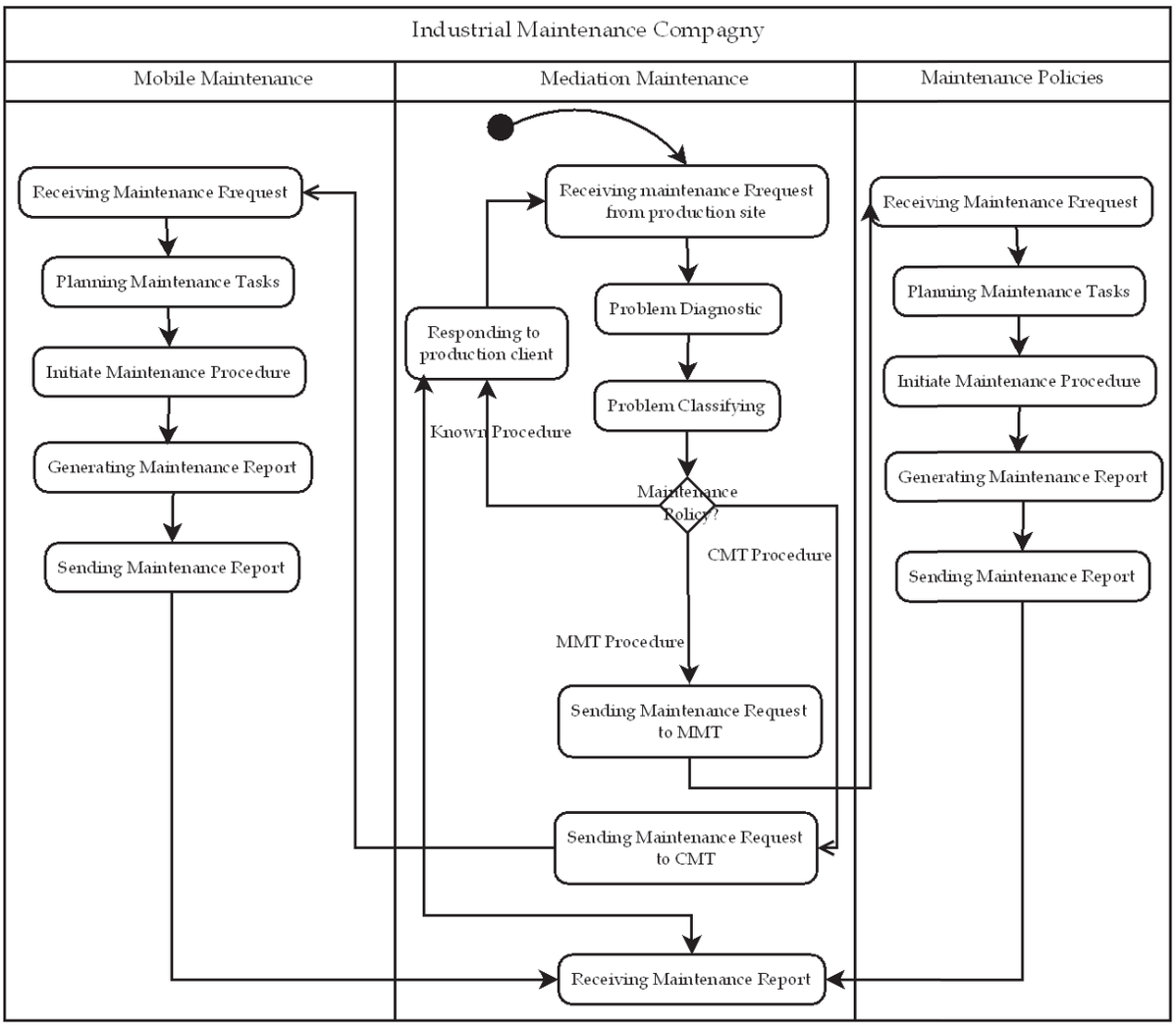

Figure 8. Behavioral roles plans description of the IMC organization

their independence from any external element not associated to them. The identification of capacity needs to determine the competences set required for each role. Capacities are added as stereotyped classes in UML class diagrams of the organizations concerned. In our case, we will identify the capacities required by the roles of the IMC organization (Figure 9).

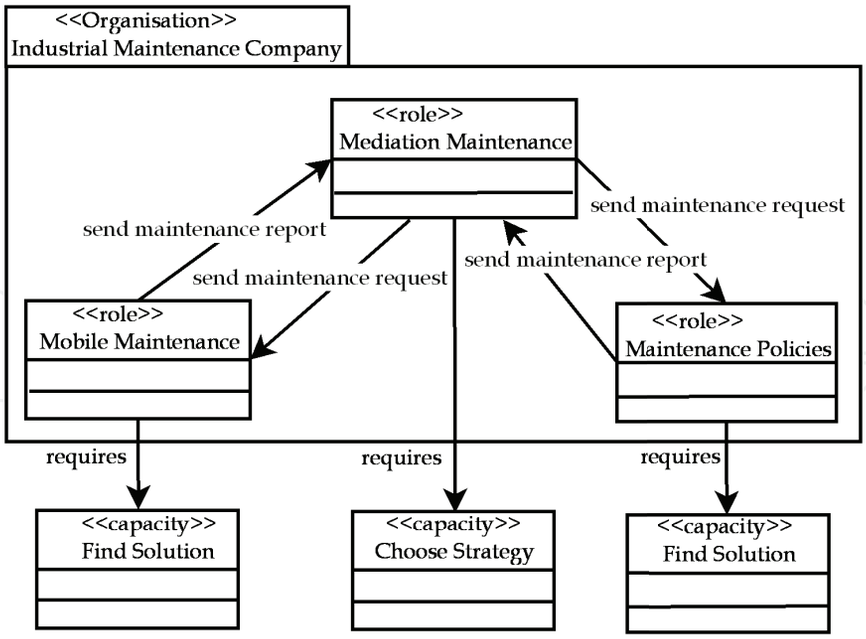

Figure 9. Identification of capacities required by the roles of the IMC organization

3.3. Holarchies design

At this stage of the process, all system organizations, their roles and associated communications are now fully described and specified. Holarchies design is the last activity of the design phase. It performs a global synthesis in which the results of all previous work are summarized and combined in one product. It is devoted to the agentification of the organizational hierarchy and the definition of the entities in charge of running it. Its objective is to define holons system and deduce the holarchy structure. To build the holarchy of the system, the organizations that compose the system are instantiated as groups. A set of holons is then created at each level, each playing one or more roles in one or more groups in the same level. Composition relations between super-holons and sub-holons are then specified in accordance with the contributions from the organizations defined in the organizational hierarchy. The organizational hierarchy is directly associated with the hierarchy of holons (or holarchy). The dynamics governing rules of holons, and the types of governance of each composed holon, are also described. All these elements are then synthesized to describe the structure of the initial holarchy system. To represent static structure of holarchy, the notation used is inspired from the "cheese board" diagrams proposed by [7]. However, it was adapted to better represent the holonic approach. The holonic structure of the IMC is presented in Figure 10.

At level 0 of the holarchy, we find three super-colons H1, H2 and H3, which play respectively Mobile Maintenance, Mediation Maintenance and Maintenance Policies in the group g0: Industrial Maintenance Company. It is reminded that this name means that the group g0 is an instance of the organization Industrial Maintenance Company. The super-holon H1 contains an instance of the Mobile Maintenance organization (g2 group), the super-holon H2 contains an instance of the Maintenance Mediation organization (g4 group) and the super-holon H3 contains an instance of the Maintenance policies (g8 group). The holon H7 of Mediation organization is also a super-holon strategy Choice who plays in the group g4 (Mediation Maintenance). This super-holon contains an instance of the organization strategy Choice (g6

group). Holon H11, of the Maintenance Policies organization, is also a super-holon which acts in the Research & Development group g8.

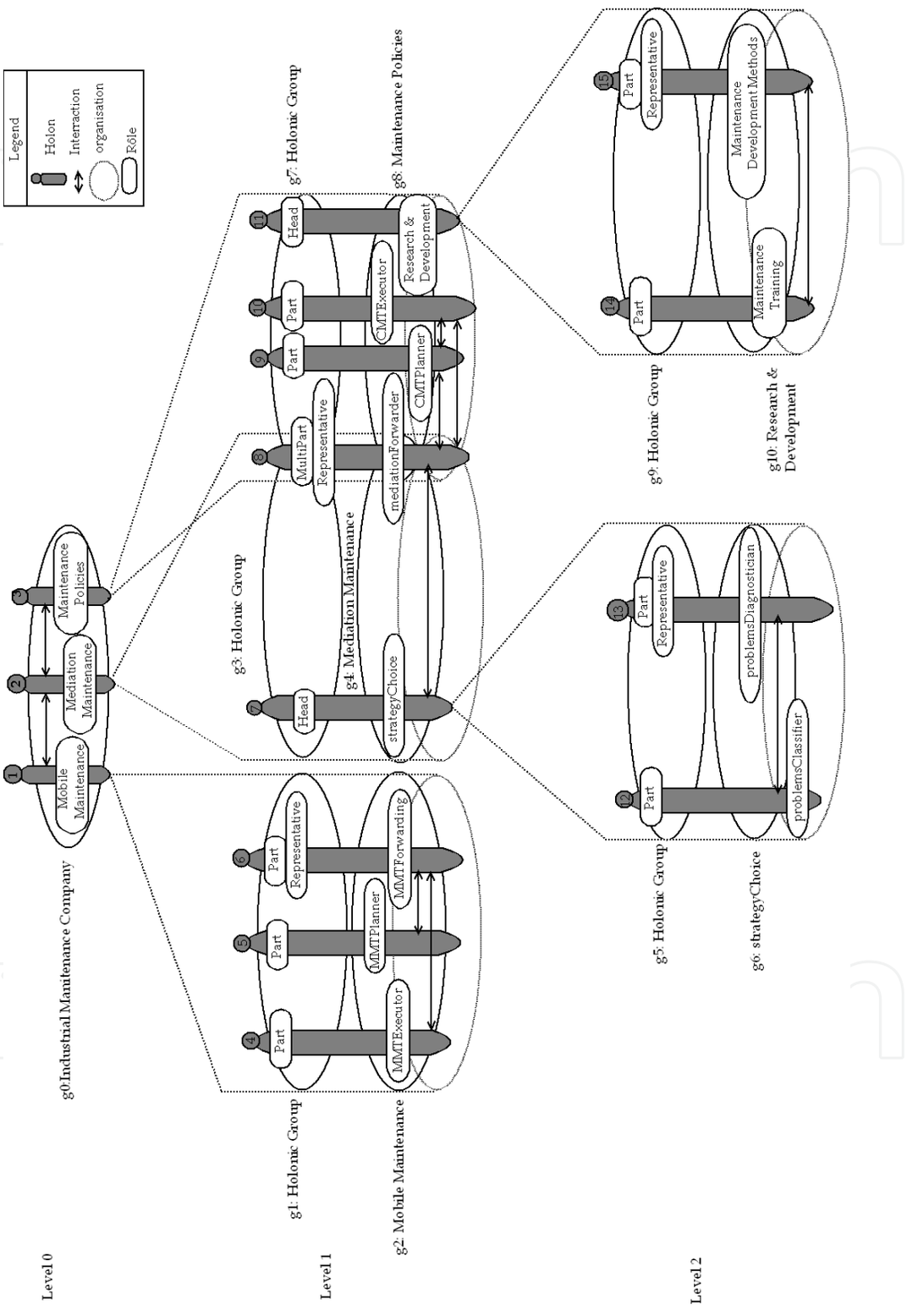

Figure 10. Holonic Structure of the IMC

This super-holon contains an instance of the Research & Development organization (g10 group). Holon H8 Playing Mediation role is named Multi-part role since sharing between Maintenance Policies and Maintenance Mediation organizations. In different organizations, interactions between holons are represented by arrows.

4. Heterogeneous formal specification and verification of holonic organisation based on SPN and object-Z language

In this section, we present an integration method of Stochastic Petri Nest (SPN) and Object-Z (OZ) by define coherent formalism called SPNOZ. This formalism is an extension of our formalism which was first defined and based on Z language [14] and GSPN called ZGSPN [19]. Petri nets (PN) are an excellent graphic formal model for describing the control structures and dynamic behaviour of concurrent and distributed systems, but Petri nets lack modelling power and mechanisms for data abstraction and refinement [22, 28].

OZ [6] is a formal notation for specifying the functionality of sequential systems. It is based on typed set theory and first order logic and thus offers rich type definition facility and supports formal reasoning. However, OZ does not support the effective definition of concurrent and distributed systems and OZ specifications often do not have an explicit operational semantics. The benefits of integrating SPN with OZ include: (a) a unified formal model for specifying different aspects of a system (structure, control flow, data types and functionality), (b) a unified formal model for specifying different types of systems (sequential, concurrent and distributed systems), (c) a rich set of complementary specification development and analysis techniques. Our approach consists in giving a syntactic and semantic integration of both languages. Syntactic integration is done by introducing a behaviour schema into OZ schema. The semantic integration is made by translating both languages towards the same semantic domain as shown in (Figure 11). An operational semantics is aimed to the description of how the system evolves along the time.

The semantic entity associated to a given specification can be seen as an abstract machine capable of producing a set of computations. Because of this, we believe that an operational semantics is a suitable representation for verification and simulation purposes. The approach consists in using a pre-existent model checker rather than developing a specific one. Both transition system models of a SPNOZ class can be used for verification purposes by model checking. In this work, the resulting specification is model-checked by using the Symbolic Analysis Laboratory (SAL) [23]. One of the reasons for choosing SAL is that it also includes verification tools and procedures that support from deductive techniques and theorem proving.

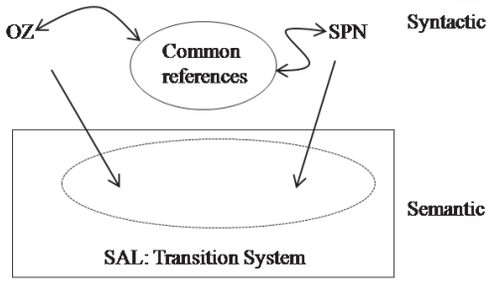

Figure 11. Composition formalisms Approach

4.1. Our syntactic integration method

To be able to build a multi-formalisms specification system, it is necessary to establish a relation of composition between partial specifications. Indeed, any whole of partial specifications must at one moment or another indicates or calls upon part of the system specified by another whole of partial specifications. As the process of composition that we use is based in type integration, the formalism of the Petri Nets is integrated into Object-Z formalism to specify schema with dynamic aspects. This composition can be expressed in several ways: sharing variables with the constraints expressed on the same entity or with translation into a single formalism for all formalisms used. SPN formalism is integrated into the formalism Object-Z to specify classes with behavioral aspects. This integration from the syntactically point of view, is based on a shared syntactic domain which consists of two parts: (a) A set of types and classes Object-Z specifying the main aspects of the SPN, (b) A function that converts a SPN in syntactic elements of the shared domain. This syntactic domain does not share instead of translation of Petri nets to Object-Z but is used to reference within the class Object-Z the elements of the Petri nets included. For instance, the approach presented here assigns to Object-Z the description of data structures and functions, and to the Stochastic Petri Nets the description of behavioral aspects. This section presents a simplified description of the operational semantics of SPNOZ [20] specification models. To express the aspects of SPN in Object-Z, we must have rules to translate a SPN into syntactic elements of the domain. For this, we use a function like relationship between PN and Object-Z scheme.

4.1.1. From PN to SPN

In [19], we proposed some syntactic integration of different classes of Petri Nets with abbreviations and extensions. We remind here that corresponding to SPN after recalling the semantic of ordinary PN. Inherently, the basic components of a Petri net, the concepts of marking and firing. Petri nets (PNs) introduced by C. A. Petri [27] were originally intended as a means for the representation of the interaction, logical sequence and synchronization among activities which are only of a logical nature [26]. A PN is a directed bipartite graph which comprises a set of places P, a set of transitions T, and a set of directed arcs defined by input and output incidence application (Pre and Post). Each place contains an integer (positive or zero) number of tokens or marks. The number of tokens contained in a place P_i will be called either $m(P_i)$ or m_i. The net marking, m, is defined by the vector of these markings, i.e., $m = \{m_1, m_2, \ldots, m_{|P|}\}$. The marking defines the state of the PN, or more precisely the state of the system described by the PN. The evolution of the state thus corresponds to an evolution of the marking, an evolution which is caused by firing of transitions, as we shall see. Generalized PN is a PN in which weights (noted by w, strictly positive integers) are associated with the arcs. When an arc $P_i \rightarrow T_j$ has a weight w, this means that transition T_j will only be enabled (or firable) if place P_i contains at least w tokens. When this transition is fired, w tokens will be removed from place P_i. When an arc $T_i \rightarrow P_j$ has a weight w, this means that when T_j is fired; w tokens will be added to place P_i.

The graph of markings (or reachability graph) is made up of summits which correspond to reachable markings and of arcs corresponding to firing of transitions resulting in the passing from one marking to another. The specification of a PN is completed by the initial marking m0. Since standard Petri nets did not convey any information about the duration of each activity or about the way in which the transition, which will fire next in a given marking, is actually

selected among the enabled transitions, a lot of research effort has been made to exploit the modeling power of PNs. Most efforts were concerned with embedding PN models into timed environments. Stochastic Petri Nets (SPN) was introduced independently by [21, 24]. Both efforts shared the common idea of associating an exponentially distributed firing time with each transition in a PN, but differed in delays.

This random variable expresses the delay from the enabling to the firing of the transition. A formal definition of a SPN is given by the following 8-tuple [18]:

$SPN = (P, T, Pre, Post, C, Ih, R, m_0)$; where
$P = \{P_1, P_2, ..., P_n\}$ is a finite, not empty, set of places;
$T = \{T_1, T_2, ..., T_m\}$ is a finite, not empty, set of transitions;
$P \cap T = \oslash$ is the sets P and T are disjointed;
$Pre : P \times T \longrightarrow N$ is the input incidence application;
$Post : P \times T \longrightarrow N$ is the output incidence application;
$C : P \longrightarrow N^+ \cup \{\infty\}$ is the capacity of places;
$Ih \subset \{P \times T\}$ is the set of k-inhibitor arcs;
$R : T \rightarrow \{\text{firing rate expressions}\}$ associated with timed transitions;
$m_0 = \{m_{01}, m_{02}, ., m_{0n}\}$ is the initial marking , with $m_{0i} = m_0(P_i)$.

Basically, a Stochastic PN may be considered as a timed PN in which the timings have stochastic values. The firing of transition T_j will occur when a time d_j has elapsed after its enabling and this time is a random value. In this basic model, usually called stochastic PN, the random variable d_j follows an exponential law of rate $\lambda_j \in \lambda(\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|T|}\})$. Noted that firing rates may depend on places markings, and in this case, firing rate expression is used. This means that: $Pr[d_j \leq t + dt | d_j > t] = \lambda_j.dt$. The probability density and the distribution function of this law are, respectively, $h(t) = \lambda_j e^{-\lambda_j t}$ and $H(t) = Pr[d_j \leq t] = 1 - e^{-\lambda_j t}$.

The average value of this law is $1/\lambda_j$ and its variance $1/\lambda_j^2$. It is clear, from the previous equations, that this law is completely defined by the parameter λ_j. A fundamental feature of an exponential law is the memoryless property, i.e.: $Pr[d_j \leq t_0 + t | d_j > t_0] = Pr[(d_j \leq t)$.

This property may be interpreted in the following way: let d_j be a random variable exponentially distributed, representing for example the service time of a customer. The service of this customer begins at time $t = 0$. If at time t_0 the service is not yet completed, the distribution law of the residual service time is exponential with the same rate as the distribution law of d_j. This property is important since it implies the following one.

Property: If transition T_j, whose firing rate is λ_j, is q-enabled at time t and if $q > 0$:

$$q \leq min_{i:P_i \in {}^0T_j} \left(m(P_i) / Pre(P_i, P_j) \right) < q + 1$$

${}^0T_j$ is the set of input places of T_j, then $Pr[\lambda_j$ will be fired between t and $t + dt] = q.\lambda_j.dt$, independently of the times when the q enabling occurred (simultaneously or not). The product $q.\lambda_j = \lambda_j(m)$ is the firing rate associated with T_j for marking m. It results from the previous property, that the marking $m(t)$ of the stochastic PN is an homogeneous Markovian process, and thus an homogeneous Markov chain can be associated with every SPN. From the graph of reachable markings, the Markov chain isomorphic to SPN is obtained. A state of the Markov

chain is associated with every reachable marking and the transition rates of the Markov chain are obtained from the previous property. Note that there is no actual conflict in a SPN. For example, the probability that firing transition T_i occurs simultaneously with transition T_j is zero since continuous time is considered. One approach may be used to analyze a SPN consists of analyzing a continuous time, discrete state space Markov process (bounded PN). Let $T(m)$, denote the set of transitions enabled by m. If $T_k \in T(m)$, the conditional firing probability of T_k from m is: $Pr[T_k \text{ will be fired } |m] = \lambda_k(m) / \sum_{j:T_j \in T(m)} \lambda_j(m)$; the dwelling time $\lambda(m)$ follows an exponential law, and the mean dwelling time in marking m is $1/\lambda(m)$ with $\lambda(m) = \sum_{j:T_j \in T(m)} \lambda_j(m)$.

4.1.2. SPN expressed in OZ

This section discusses aspects of SPN expressed in OZ. These aspects can be obtained by successive refinements starting from formal definition of ordinary PN. Figure 12 shows the specification of a SPN expressed by the OZ syntax. The class schema SPN includes, from top to bottom, an abbreviation declaration and two free types [1] places and transitions. After that, comes an unnamed schema generally called the state schema, including the declaration of all class attributes. Next schema INIT includes a predicate that characterize the initial state of the class. The last schema defines specific operation of the SPN class. The first two lines in the predicates determine the input and output places of a transition. The third and fourth predicate verifies if the transition is firable (enable): input places contain enough tokens and output places have not reached their maximum capacity. The fifth verifies inhibitor arc between P and T authorizes the firing. Finally, the last two predicates express the marking change after firing timed transition. At the end, we specify the invariants of the SPN model. Now that we have expressed aspects of SPN in OZ, we must have rules for translating any PN of syntactic elements. For this we inspired from [11] using a function like relationship between PN and the types and patterns of the OZ domain. This function transforms any PN into OZ specification written with the schemas defined in the previous section. The function we call $\aleph$ is the basis of the mechanism of syntactic integration of our multi-formalisms. This function is defined inductively $\aleph$ as follows: (a) If ψ is an ordinary Petri Nets then $\aleph(\psi)$ is a PN scheme, (b) If ψ is a Stochastic Petri Nets then $\aleph(\psi)$ is a SPN scheme.

4.2. Syntactic integration based in SPNOZ formalism

4.2.1. The case of the IMC organisation

In order to illustrate syntactic integration of our approach, we specify a part of our Industrial Maintenance Company (IMC-Part). We have limited our work to the specification of the Mobile Maintenance Teams Organization which is a part of the holonic structure of the system studied with two MMT. We assume that the choice of the intervening teams depends on the following information: the availability of the MMT, the distance at which the MMT is from the production site and spare parts stock level of MMT. We suppose that our system can be in three different states: Mobile Team(i) Available (MTA(i)), Mobile Team(i) on Production Site (MTPS(i)) and Mobile Team(i) with Critical Level of Stock (MTCLS(i)). For this reason, we use a free or built type to describe the system state:

STATE_IMC-Part ::= MTA | MTPS | MTCLS.

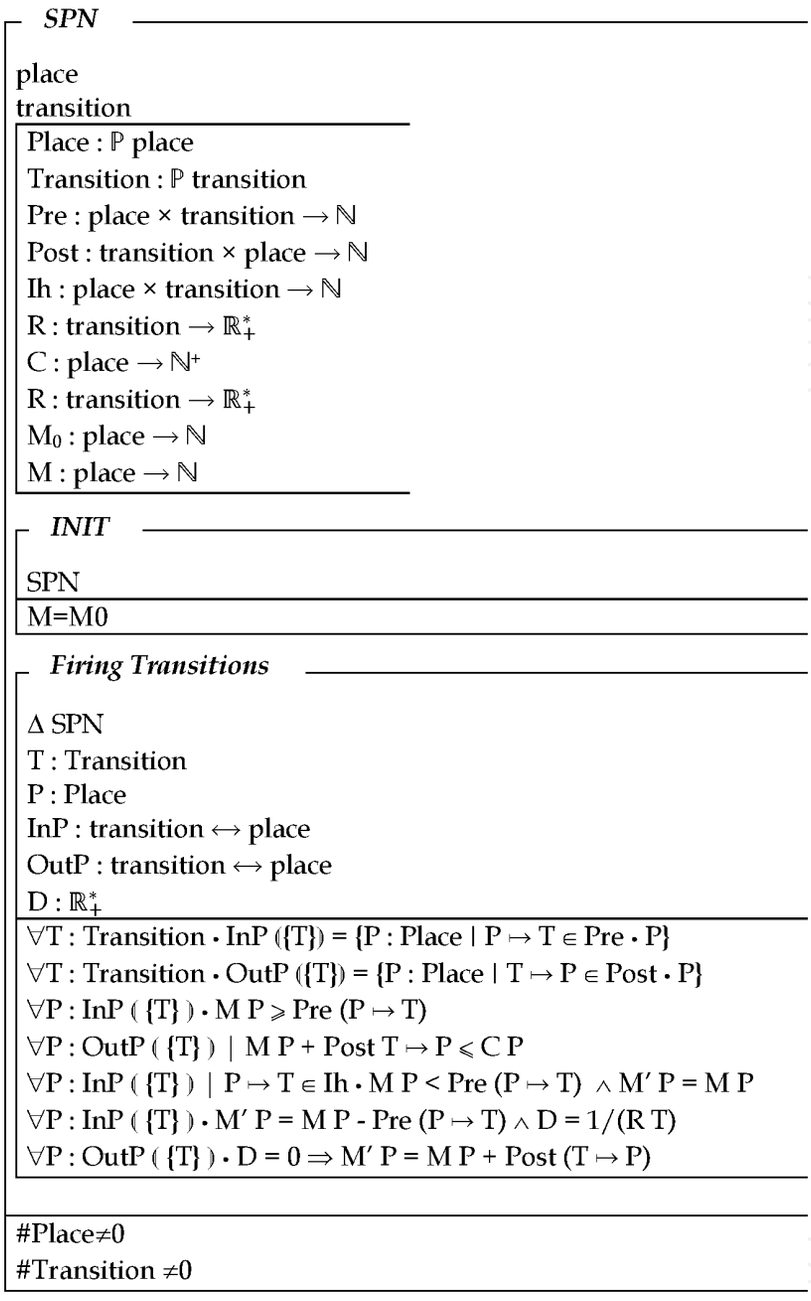

Figure 12. Specification SPN based on OZ class syntax

The system to be specified is described by its state and following average times, estimated by the Maintenance Policies organization, such as: t_{DMMTi} the average time Displacement of Mobile Maintenance Team(i) to reach Production site, associated to transition T(i); $t_{RepMMTi}$ the average time for intervention of Mobile Maintenance Team(i), associated to transition T'(i); t_{SD} the time limit to which Maintenance Team must arrive on a production site; t_{RepCMT} the average time for Repairing the defective parts by Central Maintenance Team, associated to transition T''(i). Other parameters are introduced to supplement the specification such as: Ci the level stock of Mobile Maintenance Team(i); Cmin(i) the minimum level stock of Mobile Maintenance Team(i) (below this value, MMT(i) must re-enters to the IMC); m and n the initial state of stocks. Syntactically, SPNOZ specification IMC-Part is like OZ class, with the addition

of a behaviour schema, which includes a SPN. The IMC-Part class on Figure 13 specifies a part of the IMC system. The class IMC-Part includes an abbreviation declaration and the behaviour schema which containing SPN. The state system is presented with class schema IMC-Part. In the initial state, all the MMT are available and the spare parts stock level is at its maximum (m and n). The initial state is presented with Init_IMC-Part schema.

In Figure 13, transitions in dotted lines (T1, T2, T"1 and T"2) are transition that interact with MMT forwarding and Tasks Planning organizations.

4.2.2. SPNOZ syntax

Formally, an SPNOZ class C is defined by giving a triple (V_C, B_C, O_C). The set V_C includes the variables of the class, as named in the state schema. B_C is the behaviour SPN and O_C is the set of operations of the class, the names of the operation schemas of the class. For the IMC-Part class, variables and operations are:

$$V_{IMC-Part} = \{S, t_{DMMT1}, t_{DMMT2}, t_{RepMMT1}, t_{RepMMT2}, t_{RepCMT}, t_{SD}, C1, C2, Cmin1, Cmin2\}$$

$$O_{IMC-Part} = \{SelectTeam1, SelectTeam2\}$$

Operation Select Team1 and Select Team2 can select a mobile team to involve on a production site. This selection will be made according to predefines criteria (availability of MMT, the average time Displacement of MMT and his spare parts stock level). If a team meets the different criteria, it will be chosen and its associated SPN model will be instantiated with different values for the new crossing rates $(\lambda_i, \lambda_i', \lambda_i'')$. Otherwise MMT will not be selected and its associated model will be blocked (Pre(MTAi $\mapsto$ T_i)=∞) and second team will be solicited. Finally, Select Team1 and Select Team2 expressions translate the fact that if the level of the inventories of MMTi teams with reached critical level, it will not have the possibility of intervening on any site of production (probably it will turn over to IMC).

4.3. SPNOZ semantics

The semantic integration is made by translating both languages towards the same semantic domain. The semantic entity associated to a given class takes the form of a transition system, in two possible versions, timed or untimed. As the approach proposed in [10], rather thaninsert one language into the other produces their semantic integration by adopting a common semantic domain, i.e., transition systems. The semantic description of a SPNOZ class C consists in representing the set of computations that C can take. Computations are sequences of states subject to causal restrictions imposed by the structure and the elements of C. The state of class C, which we call a situation, is essentially a pair $s = (v, m)$. Symbol $v : V_C \rightarrow D$ denotes a estimation of all the variables of C, with D denoting the super domain where all the variables take values, each one according to its type. Symbol m represents a state configuration of the behavior Stochastic Petri Nets. A state configuration is a state that can be active multiple transitions. The initial situation $s_0 = (v_0, m_0)$, is determined as follows. The initial valuation v_0 is a valuation that satisfies the predicates of the INIT scheme. Variables that do not appear in the INIT scheme usually are given default values m_0 is the initial state configuration. The basic evolution stage is the situation change, called firing, which we describe now. Step $i + 1$ takes the system from situation i to situation $i + 1$ and is noted

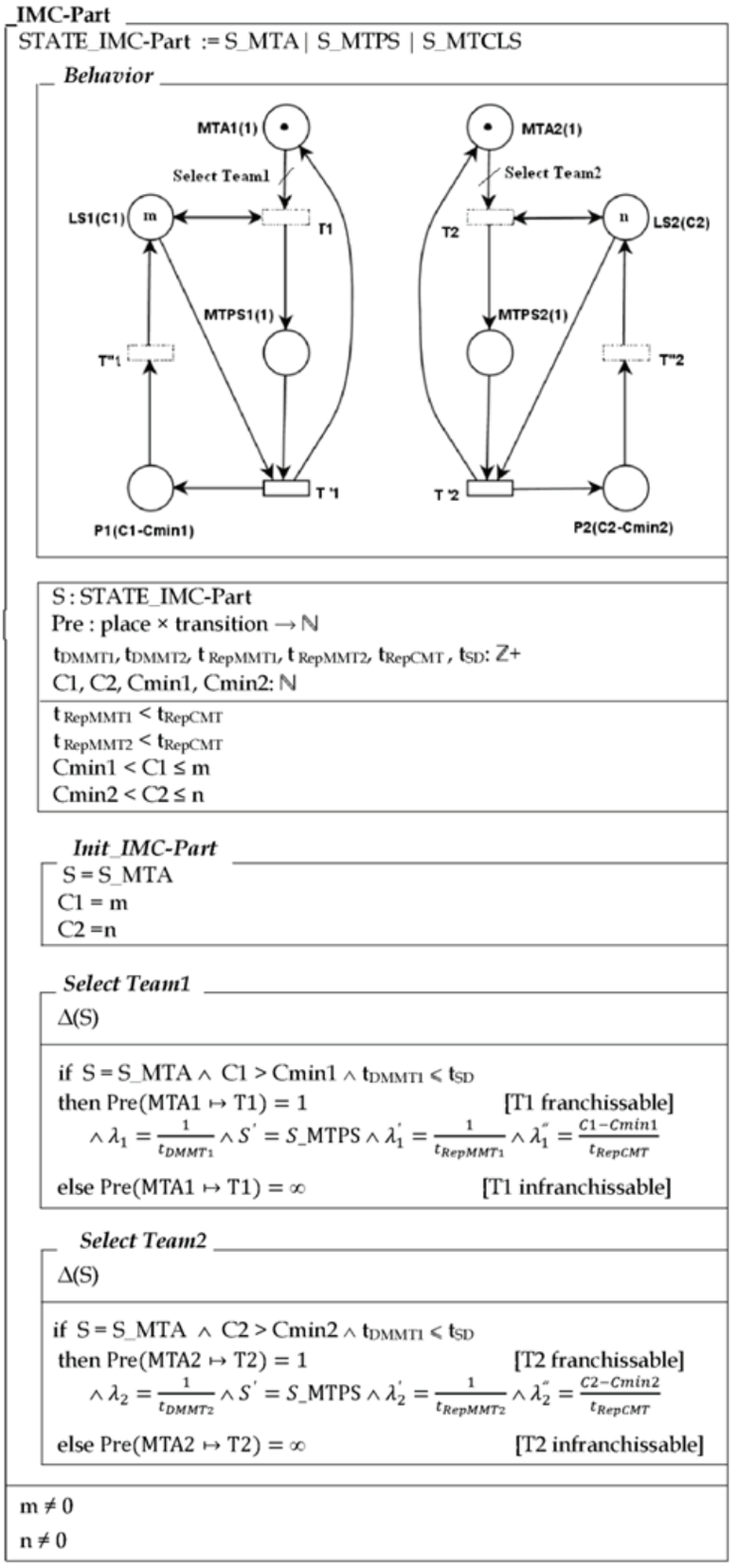

Figure 13. IMC-Part system specification based on SPNOZ syntax

$(v_i, m_i) \overset{T(m_i)}{\rightarrow} (v_{i+1}, m_{i+1})$ where $T(m_i)$ is the set of transitions activated at step i. (marking m_i). The step occurs when at least one of the SPN's transitions is enable. To describe the situation transformation produced by a step, we adopt the formalism of transition systems, particularly the Mana and Pnueli notation style by means of predicates. With class C, we associate the transition system $Tr_{sys} = (V, \phi, T_{sys})$. Symbol $V = V_C \cup \{m\}$ represents the set of variables. Variable m takes value in the graph of markings. The states of the transition system Tr_{sys} are the situations of C, i.e., valuations of the variables in V. If s denote a state of Tr_{sys}, we simplify notation as follows: for all $v \in V, s[v]$ denotes the value of v at s. Symbol ϕ represents the initial state predicate. Any valuation of V that satisfies ϕ is an initial state of the system: s is an initial state if $s[\phi] = true$ (where $s[\phi]$ denotes the valuation of formula ϕ from the value of its variables in s). Symbol $T_{sys} = T_{B_C} \cup T_{O_C}$ represents the set of transitions of PNOZ class where T_{B_C} represent the set of PN shown in behavior scheme of the class and T_{O_C} represent the set of transition generated from the operation liste O_C of the class C. A transition $T_i \in T_{sys}$ defines an elementary change of the state of the transition system. Such a change is described by a transition relation: $\rho_i = V \times V' \rightarrow \{TRUE, FALSE\}$. To the set V of variable symbols we add the set V' of variable symbols decorated with a prime character ('). For any $x \in V$, an occurrence of symbol x in ρ_i represents the valuation of x in the source state of transition T_i and an occurrence of x' the valuation of x in the destination state of T_i. The couple of states (s, s') can be a couple (source; destination) of transition T_i if $\rho_i(s[V], s'[V']) = true$, where $s[V]$ denotes the valuation of unprimed variables in the state s and $s'[V']$ the valuation of primed variables in state s'.

Concerning Timed transition system (TTS) associated to the SPNOZ class C, from the transition system the transition system $Tr_{sys} = (V, \phi, T_{sys})$ previously defined, we define $TTS_{sys} = (V^t, \phi^t, T^t_{sys})$ obtained as follows: (a) Augment variables set V with time variable t;$V^t = V \cup \{t\}$ which takes value in some totally ordered set with a lowest bound. Typically, $\mathbb{N}$ (the natural integers) is used to model discrete time and $\mathbb{R}^+$ to model dense time, (b) Let $\phi^t = \phi \wedge (t = 0)$, (c) For each transition $T_i \in T^t_{sys}$, add a variable that represents the real-time in the system. Next, we define the set of computations that a timed transition system can yield, which is to be considered as the timed semantics of a SPNOZ classes. We note $m_i \overset{T_i}{\rightarrow} m_{i+1}$ to assert that the transition system goes from state m_i to state m_{i+1} by means of transition T_i and take time $t = 1/\lambda_i$, (with λ_i : firing rate of T_i). We define a finite macro-step to be a finite succession $m_0, m_1, \ldots, m_n$ of state that: (a) $m_0 \models \phi^t$, (b) For every state m_i of TTS_{sys}, there is a transition $T_i \in T^t_{sys}$ such that $m_i \models Pre(T_i)$ and $m_i \models Post(T_i)$ and $t = 1/\lambda_i$.

In [19], more details about the full semantic integration is presented.

4.4. Validation and simulation of SPNOZ specification

SALenv contains a symbolic model checker called sal-smc allows users to specify properties in Linear Temporal Logic (LTL), and Computation Tree Logic (CTL). However, in the current version, SALenv does not print counter examples for CTL properties. When users specify an invalid property in LTL, a counter example is produced. LTL formulas state properties about each linear path induced by a module. For instance, the formula G(p⇒F(q)) states that whenever p holds, q will eventually hold. The formula G(F(p)) states that p often holds infinitely. The example illustrated by Figure 14 shows some properties of the system written in the form of theorems with the LTL and CTL formulas. The SAL language includes the clause theorem for declaring that a property is valid with respect to a modeled system by a

```
IMC: CONTEXT =
BEGIN

        Time : Real;
         Vals : TYPE = [0 .. n];
        STATE_IMC-Part : TYPE = {S_MTA, S_MTPS, S_MTCLS};
        IMC-Part : MODULE =
        BEGIN
            INPUT t_DMMT1 ,t_DMMT2, t_RepMMT1, t_RepMMT2, t_RepCMT1, t_RepCMT2, t_SD : Time
            INPUT  C1, C2, Cmin1, Cmin2, n, m : INTEGER
            INPUT Select Team1, Select Team2 : BOOLEAN
            OUTPUT  S : STATE_IMC-Part
            OUTPUT  λ1, λ2 : Time
           OUTPUT  M : ARRAY[MTA1, LS1, P1, MTPS1, MTCLS1, MTA2,
LS2, P2, MTPS2, MTCLS2] OF Vals
             LOCAL Trans_sys IN {T1, T'1, T''1,  T2, T'2, T''2}

INITIALIZATION
       S = S_MTA,
       M0 = {1, m, 0, 0, 1, n, 0, 0},
       C1 = m,
       C2 = n,
       t RepMMT1 < t RepCMT , t RepMMT2 < t RepCMT , Cmin1 < C1 ≤ m, Cmin2 < C2 ≤ n

      TRANSITION  [
        Pre (S : STATE_IMC-Part , T : Trans_sys ) : INTEGER
       t_Select Team1 :
           IF  (S = S_MTA   AND   M[MTA1] = 1 AND   C1 > Cmin1 AND
t_DMMT1⩽ t_SD )
            THEN
              S' = S_MTPS   AND M[MTPS1] = 1 AND λ1 = 1/t_DMMT1 AND
λ1' = 1/t_RepMMT1 AND λ1'' = (C1−Cmin1)/t_RepCMT1
            ELSE
               Pre (S_MTA , T1) = ∞
           ENDIF
       []

    t_Select Team2 :
          IF (S= S_MTA AND  M[MTA2]= 1 AND  C2 > Cmin2  AND t_DMMT1⩽
t_SD )
            THEN
              S' = S_MTPS   AND M[MTPS 2]= 1 AND   λ2 = 1/t_DMMT2   AND
λ2' = 1/t_RepMMT2   AND λ2'' = (C2−Cmin2)/t_RepCMT2
            ELSE
              Pre (S_MTA , T2) = ∞
           ENDIF
   END;

       %------------------------
       % system properties
       %------------------------
      th1 : THEOREM IMC-Part |- G(S = S_MTA AND M[LS1] > Cmin1 AND
λ1 = (1/ t_DMMT1) AND  t_Select Team1 => F( S' = S_MTPS));
       Liveness : THEOREM  IMC-Part |- G(M[LS1] /= 0 AND M[LS2] /=0);
       Boundedness : THEOREM  IMC-Part |- G(M[MTA1] + M[MTPS1]=1 AND
M[P1]+M[LS1]= m);

 END
```

Figure 14. SAL CONTEXT associated to the SPNOZ IMC

CONTEXT. The first theorem th1 can be interpreted as whenever the system in state S_MTA and Select Team1 is true, transition holds, the system will probably in S_MTPS state. The following command line is used:

./sal-smc IMC th1
proved.

SALenv also contains a Bounded Model Checker called sal-bmc. This model checker only supports LTL formulas, and it is basically used for refutation, although it can produce proofs by induction of safety properties. The following command line is used:

./sal-bmc IMC th1
no counterexample between depths [0, 10]

Remark: The default behavior is to look for counterexample up to depth 10. The option -depth=<num> can be used to control the depth of the search. The option -iterative forces the model checker to use iterative deepening, and it is useful to find the shortest counterexample for a given property. Before proving a liveness property, we must check if the transition relation is total, that is, if every state has at least one successor. The model checker may produce unsound result when the transition relation is not total. The totality property can be verified using the sal-deadlock-checker. The following command line is used:

/sal-deadlock-checker IMC IMC-Part
Ok (module does NOT contain deadlock state).

The liveness theorem can be interpreted as always, the quantity of stock of piece is not null in the two teams. Now, we use sal-smc to check the property liveness with the following command line:

./sal-smc -v 3 IMC-Part liveness
proved.

The Boundedness theorem can be interpreted as always, the state space system is bounded. Now, we use sal-bmc to check the property Boundedness with the following command line:

./sal-bmc IMC Boundedness
no counterexample between depths [0, 10]

5. Conclusion

In this chapter we showed that HMAS is well adapted to analyse and design an IMC holarchy. The meta-model utilized can be exploited in the implantation stage with the advantage of having formally validated its structure and its behaviour by using Heterogeneous formal specification based on Stochastic Petri Nets and Object-Z. For the moment, we are now refining our SPNOZ tool to establish a semantic-based in Markov chain isomorphic to SPN. This semantic seems best adapted to be transformed into Transition systems. Our future works will focus on a finer analysis of this system type and on a formal modelling of the various scenarios associated with the analysis stage. The notion of multi-views should be integrated. Indeed, the search for and the choice of strategy depends on the point of view of the person or the team required to take decisions according not only the constraints linked to the system but also to their environments. At the same time, it will be interesting to use HMAS which proposes multi-view holarchy introduced in [29] and consequently integrate it in the different existing meta-models.

Author details

Belhassen Mazigh
Faculty of sciences, Department of Computer Sciences, 5000, Monastir, Tunisia

Abdeljalil Abbas-Turki
Laboratoire SET, Université de Technologie de Belfort Montbéliard, Belfort, France

6. References

[1] Arthan, R.D. (1992) On Free Type Definitions in Z, Published in the Proceedings of the 1991 Z User Meeting, Springer Verlag.
[2] Burckert, H.-J., Fischer, K., Vierke, G. (1998) Transportation scheduling with holonic MAS-the teletruck approach, Proceedings of the Third International Conference on Practical Applications of Intelligent Agents and Multi-agent, pp. 577-590.
[3] Cossentino, M. (2005) From requirements to code with the PASSI methodology, In B. Henderson-Sellers & P. Giorgini (Eds.), Agent-oriented methodologies, Hershey, PA, USA: Idea Group Publishing, Chap. IV, pp. 79-106.
[4] Cossentino, M., Gaglio, S., Garro, A., Seidita, V. (2007) Method fragments for agent design methodologies: From standardization to research, In international Journal on Agent Oriented Software Engineering, 1(1), pp. 91-121.
[5] Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A. (2010) ASPECS: an agent-oriented software process for engineering complex systems How to design agent societies under a holonic perspective, Auton Agent Multi-Agent System.
[6] Duke, R., Rose, G., Smith, G. (1995) Object-Z: A specification Language Advocated for Description of Standards, Technical report Software Verification Research Center, Departement of Computer Science, University of Queensland, AUSTRALIA.
[7] Ferber, J., Gutknecht, O., Michel, F. (2004) From agents to organizations: an organizational view of multi-agent systems, In Agent-Oriented Software Engineering 4th International Workshop, volume 2935 of LNCS, Melbourne, Australia, Springer Verlag, pp. 214-230.
[8] Gaud, N. (2007) Systèmes Multi-Agents Holoniques : de l'analyse à l'implantation, Ph.D. thesis, Université de Technologie de Belfort-Montbéliard, France.
[9] Gruer, J.P., Hilaire, V., Koukam, A. (2001) Multi-agent approach to modeling and simulation of urban transportation systems, IEEE International Conference on Systems, Man, and Cybernetics, IEEE 4, 2499-2504.
[10] Gruer, P., Hilaire, V., Koukam, A., Rovarini, P. (2003) Heterogeneous formal specification based on Object-Z and statecharts: semantics and verification, In journal Systems and Software, Elsevier Science.
[11] Hilaire, V., Koukam, A., Gruer, P., Müller, J.-P. (2000) Formal specification and prototyping of multi-agent systems, In A. Omicini, R. Tolksdorf, & F. Zambonelli (Eds.), ESAW, LNAI (No. 1972), Springer Verlag.
[12] Koestler, A. (1967) The Ghost in the Machine, Hutchinson.
[13] Lebold, M., Thurston, M. (2001) Open standards for Condition-Based Maintenance and Prognostic Systems, In Proceedings Of 5th Annual Maintenance and Reliability Conference, Gatlinburg, USA.
[14] Lightfoot, D. (2000) Formal Specification Using Z, Palgrave MacMillan, United Kingdom, 2nd Revised edition.

[15] Maturana, F. (1997) Metamorph: an adaptive multi-agent architecture for advanced manfacturing systems, Ph.D. thesis, The University of Calgary.
[16] Maturana, F., Shen, W., Norrie, D. (1999) Metamorph: An adaptive agent-based architecture for intelligent manufacturing, International Journal of Production Research 37 (10) (1999) 2159-2174.
[17] Mayer, G., Wan Abdullah, Z., Lim Ai, M. (2003) Tele-Maintenance for Remote Online Diagnostic and Evaluation of Problems at Offshore Facilities, Sarawak, SPE Asia Pacific Oil and Gas Conference and Exhibition, Jakarta, Indonesia.
[18] Mazigh, B. (1994) Modeling and Evaluation of Production Systems with GSPN, Ph.D. thesis, Institut Polytechnique de Mulhouse, France.
[19] Mazigh, B. (2006) ZGSPN: Formal specification using Z and GSPN, Technical Report MAZ-SPEc-01-06, Department of Computer Science, University of Monastir, Tunisia.
[20] Mazigh, B., Garoui, M., Koukam, A. (2011) Heterogeneous Formal specification of a Holonic MAS methodology based on Petri Nets and Object-Z, In Proceedings of the Federated Conference on Computer Science and Information Systems, 5th International Workshop on Multi-Agent Systems and Simulation, IEEE Computer Society Press,Pologne, pp. 661-668.
[21] Molloy, M. K. (1981) On the Integration of Delay and Throughput Measures in Distributed Processing Models, Ph.D. thesis, UCLA, Los Angeles, CA, USA.
[22] Natarajan, S. (1989) Petri nets: Properties, analysis and applications, Computer Science Laboratory SRI International, Proceedings of IEEE, 77(4).
[23] Natarajan, S. (2000) Symbolic Analysis of Transition Systems, Computer Science Laboratory SRI International.
[24] Natkin, S. (1980) Réseaux de Petri Stochastiques, Ph.D. thesis, CNAM, Paris.
[25] Object Management Group. (2003) MDA guide, v1.0.1, OMG/2003-06-01.
[26] Peterson, J. L. (1982) Petri Net Theory and the Modeling Systems, Prentice-Hall, Englewood Cliffs, NJ.
[27] Petri, C. A. (1966) Communication with Automata, Ph.D. thesis, Technical Report RADC-TR-65-377, New York.
[28] Reisig, W. (1985) Petri Nets an Introduction, EATCS Monographs on Theoretical Computer Science 4.
[29] Rodriguez, S., Hilaire, V., Koukam, A. (2007) Towards a holonic multiple aspect analysis and modeling approach for complex systems: Application to the simulation of industrial plants, In journal Simulation Modelling Practice and Theory 15, pp. 521-543.
[30] Seidita, V., Cossentino, M., Hilaire, V., Gaud, N., Galland, S., Koukam, A (2009) The metamodel: A starting point for design processes construction, In international Journal of Software Engineering and Knowledge Engineering (IJSEKE).
[31] SPEM. (2007) Software process engineering metamodel specification, v2.0, final adopted specification, ptc/07-03-03, Object Management Group.
[32] Ulieru, M., Geras, A. (2002) Emergent holarchies for e-health applications: a case in glaucoma diagnosis, IECON 02 28th Annual Conference of the Industrial Electronics Society, IEEE, vol. 4, pp. 2957-2961.
URL: people.mech.kuleuven.ac.be/jwyns/phd/order.html
[33] Wyns, J. (1999) Reference architecture for holonic manufacturing systems-the key to support evolution and reconfiguration, Ph.D. thesis, Katholieke Universiteit Leuven.