HTM5-Trade Model for Relationship Based Trade Modelling in Multi Agent Systems

Vineet Nagrath
Laboratoire Le2i,
UMR CNRS 5158, Le Creusot, FRANCE

Aamir Saeed Malik
UTP, Perak, MALAYSIA
Department EE

Olivier Morel,
Laboratoire Le2i,
UMR CNRS 5158, Le Creusot, FRANCE

M Naufal B M Saad
Department EE,
UTP, Perak, MALAYSIA

Fabrice Meriaudeau,
Laboratoire Le2i,
UMR CNRS 5158
Le Creusot, FRANCE

Abstract—**Cloud computing and multi agent systems are two different but correlated flavors of distributed computing. Cloud computing is a business oriented model with efficient infrastructural usage as the prime focus while the multi agent system research is oriented towards the development of intelligent applications on distributed infrastructure. The commonality between the two appears when agents in a multi-agent system trade services with other agents. In recent years, model driven engineering is changing the way software is developed for complex distributed systems. Multi-view models are collection of models for a system under study, where every model is called a 'view' of the system and captures a different aspect of the design. Relationships are important aspects of multi-agent system design as agents are evolved from the concept of objects in object oriented modelling. The current work proposes a model for relationship based modelling of service oriented trade in a multi agent system. The proposed model may also be used to model intelligent cloud computing services based on multi-agent systems. The model is one of the 5 'views' of a 5-View Hyperactive Transaction Meta-Model HTM5 and thus called HTM5-Trade Model.**

Keywords—Cloud Computing; Model Driven Architecture; Multi Agent Systems; Multi View Modelling; Software Engineering

I. INTRODUCTION

The concept of 'objects' in object oriented programming came from the real world idea of objects. Modeling in object oriented methodology includes not only specifying class attributes and operations but also the relationships that exists between different classes. There is no single definition of an 'agent' that can fully capture the different functionalities and domains it is associated to. In general, an agent is an autonomous entity in a system of computing entities which interact with each other for completion of their personal and corporative goals [1], [2] and [3].

Distributed Artificial intelligence or DAI [4] is the distributed version of artificial intelligence and has two main sub-divisions. Sub-division which focuses on the distribution of the problem solving process is called Distributed Problem Solving or DPS. The second sub-division studies the interaction and behavioural complexities and is known as Multi Agent Systems or MAS [5]. Multi Agent systems are systems of computational entities in which every entity have specific objectives and roles in an environment which may have other computational entities with possibly different objectives and roles [6]. A system of agents working together giving equal weightage to personal and system goals reach a state of intelligence by the combined effect of various simple collaborations and competitions amongst its members [7].

USA's National Institute of Standards and Technology gives the following definition of Cloud Computing: "*Cloud computing is a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.*" Given the exponential growth of internet enabled devices and services, cloud computing takes distributed computing to a whole new level. Today cloud computing is used mainly for efficient usage of distributed resources by enabling scalable use of high end resources from multiple remote locations [8].

Cloud computing as a business model is becoming increasingly useful for businesses as well as for individual users. Multi agent systems could be one important constructs to implement intelligent services on the cloud computing business model. Cloud computing infrastructures are an ideal platform for agents and the advancement in large scale multi agent systems could lead to emergence of a whole new generation of intelligent services on the cloud [9].

II. MODEL DRIVEN ENGINEERING FOR AGENTS

A collection of statements about a system being studied is in definition, a model of the system [10]. The statements should collectively describe the workings of the system or they could specify the behaviour of a system in different scenarios. A model made to specify other models is called a meta-model [10]. Meta model are statements about a model that collectively describe the model. Model Driven Engineering or MDE is an engineering approach where the designs are developed without giving much importance to the implementation details. Implementation details are thus pushed as later in the project development cycle as possible giving a more abstract and easy to modify design. As the designs made in MDE are simple to understand, the non-technical stakeholders can play a bigger part in the design process and thus the end product is closer to what the client actually needs. In recent years MDE is adopted by many industries including the software industry. Robotics and multi agent systems that use cloud computing are systems with high implementational complexities and hardware dependence. MDE thus provides an attractive opportunity to improve the software development life cycle for robotic and multi agent systems.

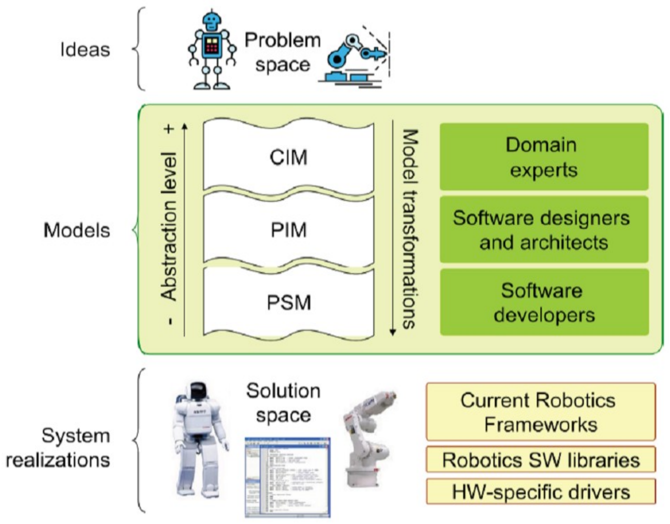

Fig. 1. MDA adapted to Industrial Robotic Product Development

Object Management Group (OMG), a consortium for computer industry in June 2003 made available its Model Driven Architecture (MDA) Guide [11]. MDA has gained popularity in industrial and research sphere, and is seen as the next big development in the way software systems are developed. OMG's MDA is a three layered model namely Platform Specific (PSM), Platform Independent (PIM) and Computation independent model (CIM). "Fig. 1" presents an adaptation of MDA in the industrial robotic product development [12].

For systems with greater complexities, more than one model is used to specify the system. In such a design methodology, the different models highlight different features of the system under study and are called different 'views' of the system. Providing various models to various viewers of the system improves readability of the design and this methodology is known as multi-view modeling. An example of this methodology is the 3-View Component Meta-Model (V3CMM) [12] which is a multi-view meta-model used for development of software components for robotic systems.

The three views namely structural, coordination and algorithmic views separately contain the structural, event-handling and algorithmic logic and provide abstract and well classified statements on the system in study "Fig. 2".

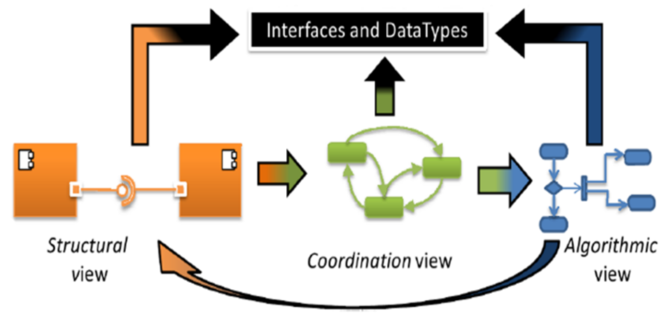

Fig. 2. A 3-View Component Meta-Model for robotic software development

III. Concepts in Multi Agent Systems

Multi Agent System transactions are similar to transactions between human agents. Not surprisingly, the BDI agents [13] are agents designed on the ideas of Beliefs, Desires and intensions which are human concepts. Institutions [14], Ethics [15] and Trust [16] are some other human concepts which commonly appear in the study of sophisticated multi agent system research.

As the multi agent systems become more humanoid, a number of human concepts will find place in multi agent systems. Beneath these complicated concepts, there are some fundamental system features that enable these complicated phenomenons to exist. Following are some of the key parameters knowledge of which could sufficiently describe its fundamental working logic.

- Structure and Location: System's structure and the physical locations of different agents including details of the hardware where they are hosted.
- Communication: Details of the network(s) that connect different agents.
- Relations: 'Named' relationships between two or more agents with details of the roles related agents play in these relationships.
- Services: Any kind of functionality that an agent provides to other agents, which can be invoked and used as a remote resource or information.
- Demands: The resources available as services in other agent, which an agent needs to use in order to complete personal or system tasks assigned to it.
- Economy: Any mechanism that enables the trade of services in a multi agent system (a currency like mechanism enabling exchange of resources as services).
- History: Capability of an agent to remember personal or system variables over an extended period of time.
- Learning: A mechanism which can induce a change in the behaviour of an agent based on the history of events.

- Behavioural Scenarios: A documentation of expected behaviour of multi agent system in some perceivable event scenarios.

Many of the above parameters exist with little variation in the cloud computing business model. Structure, Location and communication details are mostly hidden to the users in cloud computing but Services oriented economy is the fundamental component of cloud computing. Trade logic in cloud computing provides Platform, Software, Infrastructure as a service (PaaS, SaaS and IaaS) [17] which is not much different than the services that software Agents offer to one another in a Multi Agent System. As against the straightforward pay per use approach and list based service discovery in current cloud computing services, a multi agent based cloud computing service will bring a greater level of intelligence in the system. Likewise, availability of Cloud computing infrastructure to multi agent systems would generate a new generation of large-scale multi agent systems.

relationships are 'named' and they give information about the roles different related agents play in the relationship.

- Trade Model/View: This model contains the trade logic on the multi-agent system. The availability of services, the associated demands and the economic variables are specified in this model. As relations play a major role in multi-agent system trade, this is essentially a relationship based trade specification model.
- Hyperactivity Model/View: Hyperactivity is the ability of an agent to transmit its knowledge to its associated agents [18]. Hyperactivity mechanism is not just the transfer of information between agents but a whole mechanism by which agents learn from their event history, and then use the activity/hyperactivity mechanism to modify their own behavior (activity) or the behaviour of an associated agent (hyperactivity).

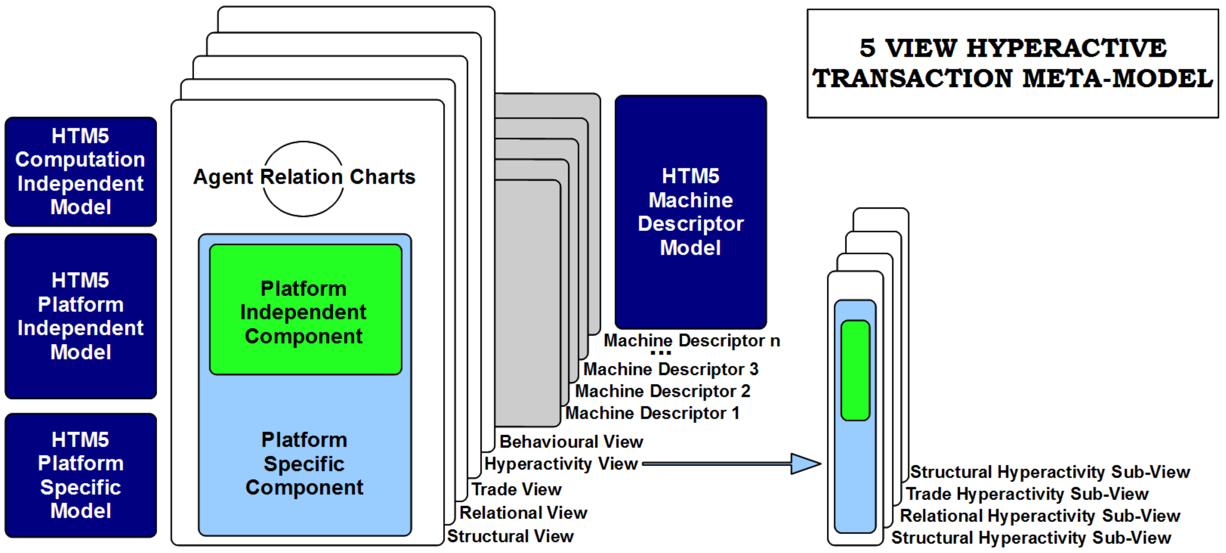
5 VIEW HYPERACTIVE TRANSACTION META-MODEL
HTM5 Computation Independent Model
HTM5 Platform Independent Model
HTM5 Platform Specific Model
Agent Relation Charts
Platform Independent Component
Platform Specific Component
HTM5 Machine Descriptor Model
Machine Descriptor n
...
Machine Descriptor 3
Machine Descriptor 2
Machine Descriptor 1
Behavioural View
Hyperactivity View
Trade View
Relational View
Structural View
Structural Hyperactivity Sub-View
Trade Hyperactivity Sub-View
Relational Hyperactivity Sub-View
Structural Hyperactivity Sub-View

Fig. 3. The 3 Layers and various Views of the 5-View Hyperactive Transaction Meta-Model

IV. An Overview Of The 5-View Hyperactive Transaction Meta-Model (HTM5)

5-View Hyperactive Transaction Meta-Model (HTM5) is an OMG's MDA based 3-layer meta-model for designing Multi-Agent Systems. The three layers of HTM5 are in agreement with the three layers of MDA and are named accordingly "Fig. 3". Being a multi-view model, HTM5 has 5 models (views) for representing different design aspects of the multi-agent system under study. The five views of HTM5 are as follows:

- Structural Model/View: Contains information about Multi-Agent System's overall structure, physical locations and details of the hardware on which each agent is hosted. This view also shows various kinds of networks that join different hardware, and thus the agents that are hosted on that hardware.
- Relational Model/View: This model informs about the relationships that exists between different agents. The

HTM5 components are passive, active or hyperactive.

- Behavioural Model/View: A model to capture multi-agent system's behaviour in various event driven usage scenarios.

HTM5's 5 views capture the 9 multi-agent system concept parameters discussed in section III. In addition to the 5 main views of HTM5 model, there is a machine descriptor view/model (MDM) for specifying the hardware on which various agents are hosted "Fig. 3". The hyperactivity view further contains sub views representing the 4 different kinds of hyperactivity present in the system under study. In all 5 main views of HTM5, the Computation Independent Layer contains an Agent Relation Chart (ARC) [19] that specifies the view specific design.

There are 5 different ARCs in for different views on HTM5 and they represent an abstract view design specification for structural, relational, trade, hyperactivity, and behavioural elements of the multi-agent system under study.

For the lower layers, Platform Independent and Platform Specific Components specify the functionalities associated to a particular view (or sub-view). In general, any component in HTM5 based multi-agent system is an agent. Along with regular agents, there are agent-components which are given special name (and symbol in ARCs) because of their specific functionality. "Merge-agents" or "Merges" are agent-components responsible for multiplexing/demultiplexing type operations in the multi-agent system. "Relationship agents" or "Relations" are agent-components which help maintain relationships amongst other agents. The Machine Descriptor Model (HTM5-MDM) introduces the concept of internal and external names for hardware specific parameters. With the change of host hardware, only minor changes in the Machine Descriptor Model are enough to reuse the HTM5 based design. Inclusion of HTM5-MDM greatly increases the number of hardware platforms where a component can be reused without modification.

HTM5 is a multi-view model and thus each of its views can be studies independently. In the current work, we are presenting the agent relationship based HTM5-Trade Model.

V. HTM5-TRADE MODEL

HTM5 Trade model is one of the 5 views of HTM5. Although these views represent different aspects of the multi-agent system and can be studied independently, there is still a number of correlating parameters which connect them. For example, the Trade 'relations' and location dependent 'costs' cannot be specified without first knowing the structural and relational details (HTM5 Structural and Behavioural views) of multi-agent system, and the Trade 'behaviour' cannot be fully specified without referring to the behavioural Model (HTM5 Behavioural view). When studying an individual view of HTM5, it is to be assumed that other views are also available for reference.

Each of the HTM5 views has elements in all three layers of the model "Fig. 3". We now present the Trade view elements for each of the three layers.

A. Computation Independent Layer

In HTM5, Computation Independent trade design is specified using Trade-View Agent Relation Charts (T-ARC). Following are some of the HTM5 and ARC [19] elements "Fig. 4" which will be required for Behavioural Trade modelling at Computation Independent Layer:

- HTM5 Component: HTM5 is a component based architecture where every agent is modeled as one independent component. Agent components with specific jobs are further specified as "merge agent components" and "relational agent components". These components could be passive, active or hyperactive "Fig. 4" based on whether they have learning capabilities and the capability to transfer what they learned to other components [18].

- Agent Component: An HTM5 agent component is a representative of the base software in the multi-agent system. The agent and the base software are both running on a hardware host, and the agent is connected to other agents in the system via one or more network clouds. HTM5 agent component has a control unit which is governed by a fixed number of control parameters. Presence of an update mechanism for control parameters makes an agent "active" and a hyperactivity mechanism qualifies them as a "hyperactive" agent [18].

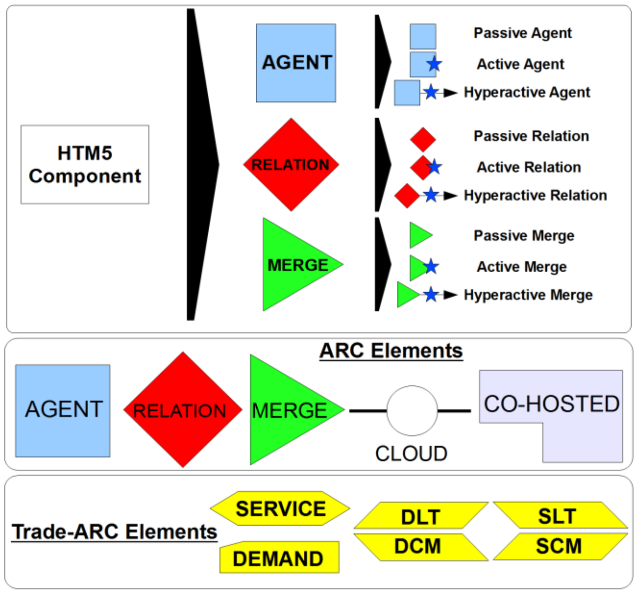

Fig. 4. HTM5 Trade and Behavioural View Elements

- Merge Component: HTM5 merge agent components are agent components which are modeled for the specific operation of multiplexing/demultiplexing information from one root channel to a number of similar components. Like any other agent component, "Merge" could be passive, active or hyperactive in design.

- Relational Component: HTM5 manages agent relationships using special agent components called "Relation". These agent components are responsible for maintaining relationship amongst other agents and they store and manage the parameters that define a relationship. Like "Merges" and "Agents", HTM5 "Relations" could also be passive, active or hyperactive.

- Co-Hosted Components: In case more than one HTM5 components are hosted on the same hardware, they are grouped together in ARC diagrams as co-hosted components. They are used to include the physical location of components in the design model.

- Cloud: A "cloud" in HTM5 is any kind of network that enables communication between components. To identify the location of a specific network, and to know which of the components are connected using a particular network, the "clouds" are uniquely numbered in the ARCs [19].

- Service: Cloud computing is a service oriented business model which enables resources to be traded as a service (XaaS, where X could be Infrastructure, Platform, Software, Strategy, Collaboration, Business Process, Database, Network or Communication [20]). In HTM5 Trade model, a service could be any of the above resources that an HTM5 component makes available to be used by other agent components.

- Demand: Any resource required by any HTM5 component is a demand by that agent component. An HTM5 component may invoke one of the other HTM5 components which are offering that resource as a service.

- Lookup Tables: A "Service" or a "Demand" made by a HTM5 component has to be listed at a location for other HTM5 components to see. Ideally "Relation" components may store Lookup tables for Demands and Services available in a relationship (Demand Lookup Table: DLT and Service Lookup Table: SLT "Fig. 4"). These tables may also be used for enlisting other items that are essential for sustaining a relationship and they may be stored at any HTM5 components other than the "Relation" components.

- Cost Metrics: Once a "Service" or "Demand" is identified by a HTM5 component, the cost metrics enables the component to make trade related decisions by giving costs associated with a particular "Service" or "Demand". These "costs" could be any variables that specifies the location, distance, quality, reliability of the service (or demand), and this information enables a HTM5 component to make a wise decision on whether to take an offered "Service" (or to offer a service to a particular "Demand"). The Demand and Service Cost Matrices (DCM and SCM in "Fig. 4") could ideally be stored at "Relation" components (or any other HTM5 component as per the designer's conviction) and they may also contain any other information that is required for a trade relationship.

The elements mentioned above are used to create Trade-ARCs which are an abstract representation of the service oriented trade logic of the multi-agent system under study. A step by step usage example of the model is presented in “Section VI” of the paper.

B. *HTM5 Platform Independent and Platform Specific Component Design*

As in OMG’s MDA guide, the second layer of HTM5 is a Platform Independent Model “Fig. 3” for components which were introduced in the Computation Independent layer. The elements of HTM5 components are object oriented in nature and they are specified using elements of Unified Markup Language (UML) [21]. “Fig. 5” shows the elements of a HTM5 component. A HTM5 component is subdivided into two component models, one for Platform Independent elements and the other one for Platform Specific elements. For each of the views (and Hyperactivity sub-views) there is a corresponding class in Platform Independent Component (e.g. PIC_T is a class for trade view “Fig. 5”). Classes of Platform independent Component are abstract classes as the platform specific data and functional elements are abstract [22]. In Platform Specific Component, the classes for individual view override the abstract classes of the platform Independent Component (e.g. PSC_T extends/inherits PIC_T class and overrides the abstract elements of PSC_T class “Fig. 5”). Objects of the Platform Specific Classes thus contain Platform Independent components from the abstract classes and the overridden Platform Specific Components from the Platform Specific Classes.

Other elements of the HTM5 component are UML based Use-Case and sequence diagrams for modeling behaviour within individual classes (Intra-Class) and in between different classes (Inter-Class). Global data and methods and algorithmic definition for class methods are also a part of the HTM5 component. For Behavioural Trade modeling, the Trade view specific classes are modeled. When each of the 5 views model their individual classes, the inter class diagrams are modeled giving a complete model for the HTM5 component. The idea behind having separate views, and separate classes for views is to maintain modularity in both design and implementation level and improves reusability of components. “Section VI”

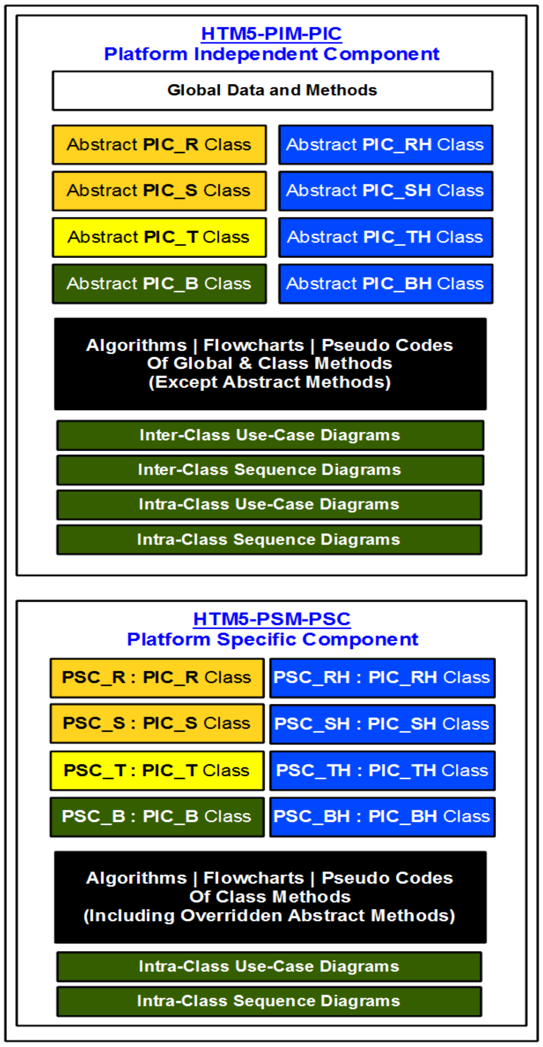
HTM5-PIM-PIC
Platform Independent Component
Global Data and Methods
Abstract PIC_R Class
Abstract PIC_RH Class
Abstract PIC_S Class
Abstract PIC_SH Class
Abstract PIC_T Class
Abstract PIC_TH Class
Abstract PIC_B Class
Abstract PIC_BH Class
Algorithms | Flowcharts | Pseudo Codes Of Global & Class Methods (Except Abstract Methods)
Inter-Class Use-Case Diagrams
Inter-Class Sequence Diagrams
Intra-Class Use-Case Diagrams
Intra-Class Sequence Diagrams
HTM5-PSM-PSC
Platform Specific Component
PSC_R : PIC_R Class
PSC_RH : PIC_RH Class
PSC_S : PIC_S Class
PSC_SH : PIC_SH Class
PSC_T : PIC_T Class
PSC_TH : PIC_TH Class
PSC_B : PIC_B Class
PSC_BH : PIC_BH Class
Algorithms | Flowcharts | Pseudo Codes Of Class Methods (Including Overridden Abstract Methods)
Intra-Class Use-Case Diagrams
Intra-Class Sequence Diagrams

Fig. 5. Platform Independent and Platform Specific Component Models in HTM5

gives a detailed usage example for the Trade View component modeling in HTM5.

VI. A HTM5-Trade Modelling Example

A. Sandbox: Computation Independent Model

The first step in modelling of any multi-agent system is to identify the system components. We start by taking a hypothetical multi-agent system named "Sandbox" which has 5 identifiable hardware units as follows:

- Robots: 2 ground robots capable of making a 3Dimensional map of their surroundings.
- Computers: One laptop onsite capable of communicating with the two robots via an infrared communication link and a remote desktop computer connected to the internet.
- Web Server: One web server connected to the two computers via internet.

We now design a system with 7 agents running on the above mentioned hardware units. In a real project, the design team will brainstorm on the structure and relational elements of the system to create an Agent Relation Chart [19] for the system like the one shown in "Fig. 6". The ARC specifies the following design elements of the "Sandbox" multi-agent system (MAS):

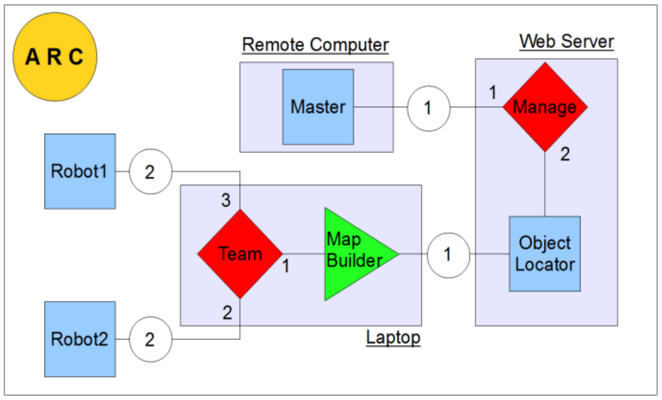

Fig. 6. ARC for Sandbox multi agent system

- The Sandbox MAS has one "Merge", two "Relation" and 5 agent components. In total there are 7 HTM5 components.
- "Team" and "Map Builder" components are co-hosted on "Laptop" while "Manager" and "Object Locator" are co-hosted on the "Web Server".
- The two "Robots" and "Map Builder" are in a "Team" relationship with "Map Builder" as the team leader (Assuming "Team" relationship defines port 1 as the port for team leader).
- "Master" manages the "Object Locator" component as they are related by the "Manager" relation with port 1 assigned to "Master".
- The two network clouds in the MAS are named as "1" and "2" where "1" stands for internet and "2" stands for an infrared communication link.

The next step in the modelling process is to identify different services, demands and relational trade elements. Once identified, these elements of the relational trade logic can be modeled onto the Trade-Agent Relation Chart [19] like the one shown in "Fig. 7".

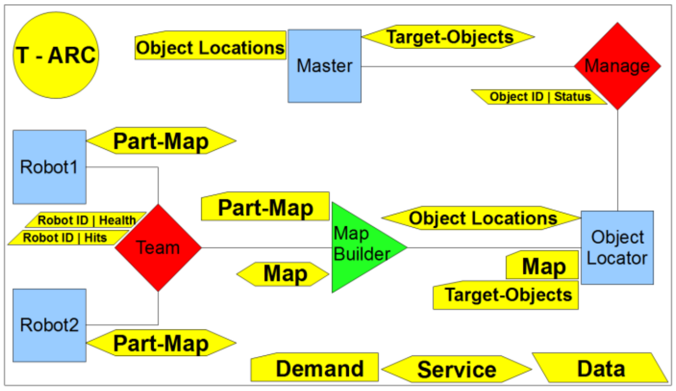

Fig. 7. Trade-ARC for Sandbox multi agent system

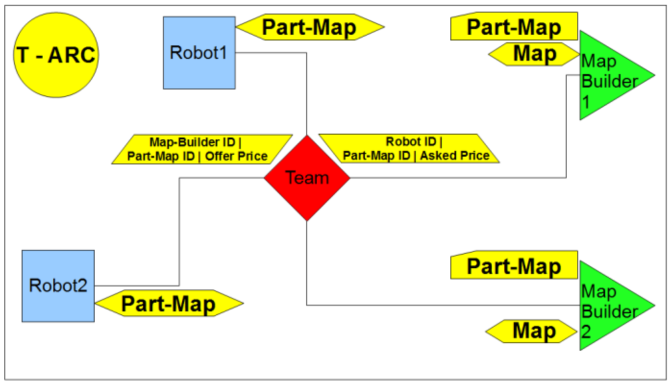

Fig. 8. Trade-ARC for scenario with Lookup Tables

The Trade-ARC in "Fig. 7" contains the following relational trade information about the "Sandbox" MAS:

- The two "Robots" provide the 3D maps that they build as a service. These partial maps are a demand at "Map Builder" component.
- The "Map Builder" component provides the complete 3D map as a service to the "Object Locator" component.
- The "Object Locator" component demands the information about the objects which are to be found in the map, these objects are provided by the "Master" component as a service.
- "Master" component demands the locations of the found objects, which is a service provided by "Object Locator".
- There are 2 look-up tables in the "Team" relation. The lookup table helps maintain the team structure by

providing the team leader ("Map Builder") information that it can use to take team-managerial decisions.

- The "Manage" relation contains a cost metric to track the status of individual search objects.

It is to be noticed that the lookup tables and the cost metrics in the "Sandbox" multi-agent system are not used for economic logic implementation. Here they are relationship variables which are essential for maintaining that particular relationship. In another scenario, imagine that the "Robots" are not working for free in the "Team" and they are independent workers offering parts of the map to the "Map Builder" at an asking price (which may be different for different robots). Likewise there may be more than one "Map Builders" and they all offer different prices for different parts of the 3D maps (depending on their business logic). In such scenario, there will be a service lookup table in the "Team" relationship that enables "robots" to publish the parts of map that they have with their asking prices. There will also be a demand lookup table to enable different "Map Builders" to showcase the parts of map they need along with their offered prices. "Fig. 8" shows the section of T-ARC in the above mentioned scenario.

B. *Sandbox: Platform Independent Model*

Once we reach the platform independent component design phase of the modelling process, a design team may go in different directions to implement the computation independent model. We here present one of many ways in which HTM5 methodology can be implemented. The process explained in this section is for the HTM5 component "Robot1". The same procedure is to be followed for each of the 7 components of the "Sandbox" multi-agent system. In "Fig. 9" the main class for the "Robot1" component is named as "Sandbox_PIC_Robot1". Within the class, there are objects being defined belonging to the 5 PSC classes (See "Fig. 5 for HTM5 PIC and PSC component structure"). The 5 PSC classes have to be defined separately, and are reusable for another component as classes are reusable templates. We now explain how these 5 classes are modeled by taking an example for the Trade-View class "Sandbox_PSC_Robot1_T".

We can see in "Fig. 9" that the "Sandbox_PSC_Robot1_T" class extends the "Sandbox_PIC_Robot1_T" class. The "Sandbox_PIC_Robot1_T" class is an abstract class because the "Get_System_Time ( )" is an abstract function. This function is just declared but not defined in the "Sandbox_PIC_Robot1_T" class. Abstract classes are thus incomplete classes and cannot be used to make objects. In essence, all platform specific functionalities (related to Trade) are to be kept in abstract functions while other functionalities and data items are declared as well as defined in the PIC trade class itself.

C. *Sandbox: Platform Specific Model*

Once the abstract PIC classes are modeled, the third layer of the HTM5 model is used to define the Platform specific components of the HTM5 component. For different platforms, a different PSC model is made. This enables the top two layers (Computation Independent and platform independent layers) to remain unchanged and reused for different platforms.

The platform specific classes extend the platform independent classes and then override the abstract functions. In "Fig. 9", the abstract function "Get_System_Time ( )" is overridden, and thus the incomplete elements of the platform independent classes are completed in the platform specific classes. As the PSC classes are not abstract classes, they can be used to make objects. The "Robot1" component class defines the object for the "Sandbox_PSC_Robot1_T" class. The reusability aspect of using classes for individual views within the components can be exemplified by a scenario where the PSC class made for "Robot1" is reused in "Robot2" just by defining an object of "Sandbox_PSC_Robot1_T" class in "Robot2".

D. *Sandbox: Hardware Independence*

The layers of HTM5 based on OMG's MDA enables the use of different platform specific models without modifying the computation independent and platform independent layers. The change in hardware on which a platform is running however may require a change in the PIC and PSC classes of HTM5. To avoid this situation, the HTM5 Machine Descriptor Model is included in the HTM5 methodology ("See Fig. 3 for complete set of HTM5 views").

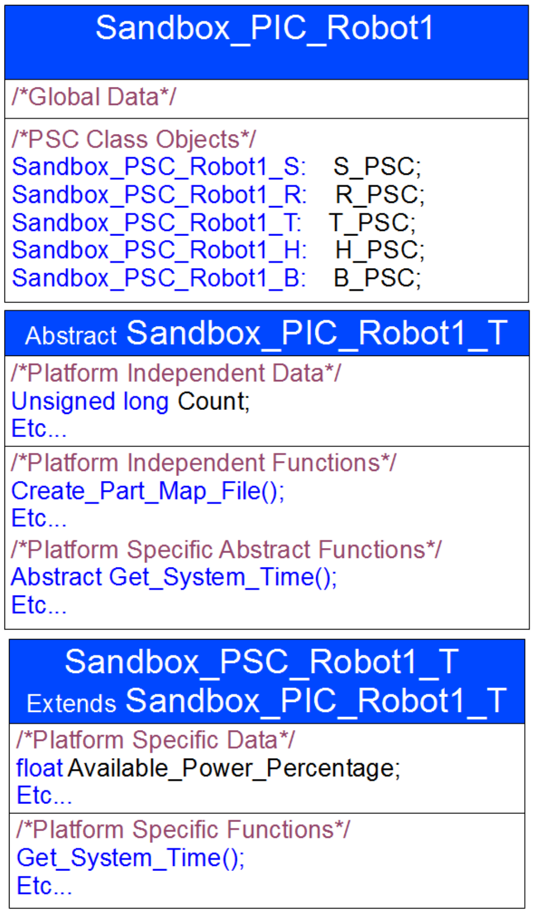
Sandbox_PIC_Robot1
/*Global Data*/
/*PSC Class Objects*/
Sandbox_PSC_Robot1_S: S_PSC;
Sandbox_PSC_Robot1_R: R_PSC;
Sandbox_PSC_Robot1_T: T_PSC;
Sandbox_PSC_Robot1_H: H_PSC;
Sandbox_PSC_Robot1_B: B_PSC;
Abstract Sandbox_PIC_Robot1_T
/*Platform Independent Data*/
Unsigned long Count;
Etc...
/*Platform Independent Functions*/
Create_Part_Map_File();
Etc...
/*Platform Specific Abstract Functions*/
Abstract Get_System_Time();
Etc...
Sandbox_PSC_Robot1_T
Extends Sandbox_PIC_Robot1_T
/*Platform Specific Data*/
float Available_Power_Percentage;
Etc...
/*Platform Specific Functions*/
Get_System_Time();
Etc...

Fig. 9. PIC and PSC Trade View classes for Robot1 Component

HTM5-MDM is parallel to the PIM and PSM layers of the HTM5 model ("Fig. 10"). Both PIC and PSC classes of HTM5 uses the internal names (defined in MDM for the hardware) to access hardware functionality and variables. These internal names remain unchanged with the change of hardware and thus the PIC and PSC classes can be used as it is when the hardware is changed. An Internal to external name mapping and adjustment section in MDM enables the internal names to retain their meaning when hardware is changed ("Fig. 10").

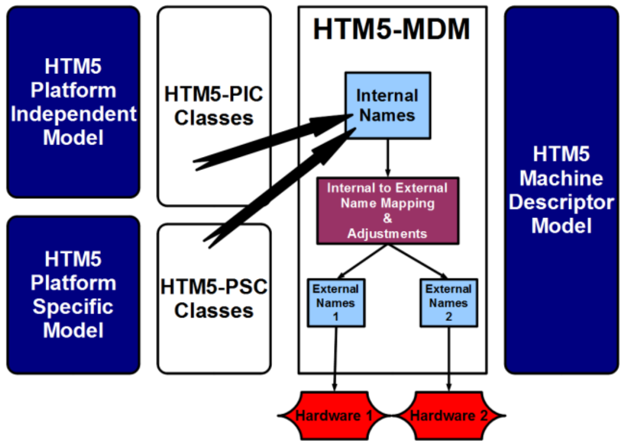
HTM5 Platform Independent Model
HTM5 Platform Specific Model
HTM5-PIC Classes
HTM5-PSC Classes
HTM5-MDM
Internal Names
Internal to External Name Mapping & Adjustments
External Names 1
External Names 2
HTM5 Machine Descriptor Model
Hardware 1
Hardware 2

Fig. 10. HTM5 Machine Descriptor Model

E. *Steps for HTM5 Relational Trade Modeling*

Based on the example explained in the current "Section VI", following are the steps for using HTM5 for Relational Trade modelling:

- System Identification and creation of Agent Relation Chart (ARC).
- Trade modelling based on relations and creation of Trade-ARC.
- Creation of Platform Independent Abstract classes for the Trade view (and other views).
- Extending the Platform Independent classes to create Platform Specific classes.
- Creation of HTM5 Component classes by making objects of PSC classes.
- Creation of Algorithmic documentation and Use-Case Diagrams for each of the HTM5 components.
- Defining HTM5 Machine Descriptor Model if required.

VII. Conclusion

Model driven architecture is an excellent software engineering methodology for designing software for complex distributed systems. Multi agent systems based on cloud computing business model will give rise to a new generation of cloud enables intelligent services. To model such systems, the multi view modelling methodologies like HTM5 will provide a complete toolset based on industrially accepted practices like component based software engineering and UML. For agent relationship based trade modelling, the HTM5-Trade model provides a multi layered, component based methodology.

The current work presented the key elements of the trade model along with a simple but comprehensive example. The example presented scenarios where the HTM5 Trade model can be used for designing cloud computing based trade logic as well as the relationship based exchange of services.

VIII. Acknowledgment

The current research is being funded by the Laboratory Le2i (CNRS 5158, Le-Creusot, FRANCE), Bourgogne regional council (Regional French administration) and the Universiti Technologi Petronas (Perak, Malaysia).

References

[1] M. Wooldridge. "An Introduction to Multi-agent Systems", Published in February 2002 by John Wiley & Sons (Chichester, England), ISBN: 0 47149691X, 2002.

[2] M. Luck, P. McBurney and C. Preist. "Agent Technology: Enabling Next Generation Computing", In A Roadmap for Agent-Based Computing, ISBN: 0854327886, ver. 1.0. Southampton: AgentLink 2003.

[3] M. Luck, P. McBurne, O. Shehory and S. Willmott. "Agent Technology: Computing as Interaction", In A Roadmap for Agent Based Computing, Compiled, written and edited by M. Luck, P. McBurney, O. Shehory, S. Willmott and the AgentLink Community, pp. 11-12, 2005.

[4] P. Stone and M. Veloso. "Multi-agent Systems: A Survey from a Machine Learning Perspective", In Autonomous Robots, vol. 8, no. 3, pp. 345-383, July 2000.

[5] L. Panait and S. Luke. "Cooperative Multi-Agent Learning: The State of the Art", In Autonomous Agents and Multi-Agent Systems, Ed. Springer-Verlag, vol. 11, no. 3, pp. 387-434, 2005.

[6] N. R. Jennings and S. Bussmann. "Agent-Based Control Sys-tems. Why Are They Suited to Engineering Complex Sys-tems?", In IEEE Control Systems Magazine, vol. 23, no. 3, pp. 61-73, Jun. 2003.

[7] G. Weiss. "Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence", Edited by Gerard Weiss. ISBN: 0-262-23203-0, 1999.

[8] M. Armbrust, et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, April 2010.

[9] Domenico Talia "Cloud Computing and Software Agents: Towards Cloud Intelligent Services", in proc. Of CEUR-WS, Vol.741, pp-2-6, 2011.

[10] E. Seidewitz, "What models mean", IEEE Softw., vol. 20, no. 5, pp.26–32, 2003.

[11] OMG, Model Driven Architecture Guide, version v1.0.1, June, 2003. http://www.omg.org/docs/omg/03-06-01.pdf

[12] Diego ALONSO, Cristina VICENTE-CHICOTE, Francisco ORTIZ, Juan PASTOR, Bárbara ÁLVAREZ "V3CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software Development," Journal of Software Engineering for Robotics (JOSER), January 2010, 3-17.

[13] D. Kinny, M. Georgeff, A. Rao: "A Methodology and Modelling Technique for Systems of BDI-Agents" in: W. van der Velde, J. Perram (eds.): Agents Breaking Away. Proc. MAAMAW'96, LNAI 1038, Springer, Berlin, etc., 1996.

[14] M. Luck, P. McBurney, Chrs Preist, "Agent Technology: Ena-bling Next generation Computing. A Roadmap for Agent Based Computing", AgentLink II, January 2003.

[15] Singh, M. An Ontology for Commitments in Multiagent Sys-tems: Toward a Uni-fication of Normative Concepts. Artificial Intelligence and Law v. 7 (1) (1999) 97-113.

[16] Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jen-nings. Trust in multi-agent systems. Knowl. Eng. Rev., 19(1):1–25, 2004.

[17] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," UC Berkeley Reliable Adaptive Distributed Systems Laboratory, February 10, 2009.

[18] Vineet Nagrath, Fabrice Meriaudeau, Aamir Saeed Malik, Olivier Morel, “Introducing The Concept of Hyperactivity in Multi Agent Systems,” IEEE-CSNT2013, April 2013.

[19] Vineet Nagrath, Fabrice Meriaudeau, Aamir Saeed Malik, Olivier Morel, “Agent Relation Charts (ARCs) for Modeling Cloud based transactions,” IEEE-CSNT2012, May 2012.

[20] "ITU-T Newslog - Cloud computing and standardization: Technical reports published". International Telecommunication Union (ITU). Retrieved 16 December 2012. Link: http://www.itu.int/ITU-T/newslog/Cloud+Computing+And+Standardization+Technical+Reports+Published.aspx

[21] OMG, Unified Modeling Language (UML) Superstructure specification v2.1.1, formal/2007-02-05, Feb. 2007. Available: http://www.omg.org/cgi-bin/apps/doc?formal/07-02-03.pdf

[22] http://en.wikipedia.org/wiki/Abstract_type