Agent driven Peer-to-Peer Cloud Robotics*

V. Nagrath, O. Morel, A. Malik, N. Saad, F. Meriaudeau

Abstract— **Cloud enabled robotics is currently understood as one or many robotic clients making use of the resources and services made available by remote servers placed across the network cloud. These servers provide gateways to access the infrastructure, platform, software, algorithm or process as a service. Tapping into online infrastructure and knowledge saves the cost of carrying all required capabilities onboard the robot. In peer-to-peer cloud computing, every robot can act as a service provider. This paper presents a meta-model for enabling agent driven trade over local and global network clouds. The presented meta-model is one of the five views of HTM5, a five-view hyperactive transaction meta-model for multi agent system design. The meta-model represents the robots by their respective agents in the cloud, along with special agents which host and maintain service and demand registry. The robotic agents may have predefined relationships and service contracts with other robotic agents. Special agents are present in the system to maintain the trade between agents which have a predefined relationship or a trade contract. Unlike client-server based cloud robotics, the peer-to-peer structure enables a more active exchange of services between the robots. The presence of a relationship and contract based mechanism for trade preserves the fundamental business logics of cloud computing.**

I. Introduction

Cloud enabled robotics or cloud robotics is an emerging trend in distributed intelligent systems. In the past decade there has been a remarkable improvement in the speed and reach of the computer networks. Lower costs of network enabled devices and connectivity brought us in the age of anywhere-anytime computing. The cloud computing business models have made network enabled sharing of computing infrastructure, platforms and software systems a reality. Network (cloud) enabled robotic clients can make use of cloud computing business models to share resources and services. A robot may now carry hardware only for a limited set of capabilities onboard and can theoretically access all infrastructure and knowledge available online. Cloud enabled robotics present a greater canvas of applications for designers to work upon and with that arises a need for effective tools to design and model such systems.

Cloud robotics is currently seen as a client server system where robotic clients tap into the online resources through one or many web servers. This approach is useful as many online web centers provide useful knowledge for robots such as maps, image and text based web search and algorithmic support. There is however an interesting opportunity in enabling a peer-to-peer cloud robotic framework where every robot in the system can act as a server offering its hardware and knowledge resources to its associated robots. The dependency of all robots on a centralized server is removed in a peer-to-peer system. This system also allows individual robots to share their resources as a service, thus it's a more effective use of available system resources. By the presence of a mechanism to implement cloud computing business logics in such trade transactions between robots, robots could share their resources in exchange for other resources or an actual transfer of money. This mechanism should be useful for robots working in a predefined team as well as for trade of services between robots of random affiliations. In the current paper we present an agent oriented model for enabling peer-to-peer cloud robotic trade.

The concept of agents is essentially an extension to the concept of objects in object oriented methodology. An object is specified by its class attributes and operations which can be accessed from other objects. This open access to an object's interior functionality compromises its autonomy. What makes agents different from objects is the autonomy an agent has over its own operations. Agents are deployed in various domains with different functionalities and thus there is no consensus on the definition of an agent. In general, an autonomous entity in a system of computing entities interacting with other autonomous entities with a mandate to complete their personal and shared goals is known as an agent [1], [2] and [3].

The concept of agency [4] however explains a wider set of abstractions associated with an agent. Apart from autonomy, an agent should be heterogeneous in design giving its designer independence to design it in any manner irrespective of the other agents and the network administration. Its interactions must protect its autonomy and the communication protocol should not reveal its internal design. The commitments that an agent makes to other agents should be based on a social concept with a debtor, a creditor, action and a context. Unlike components, the receiving party (agent) takes the ownership/responsibility for an action taken when a message is received and not the party (agent) that sends the message. An agent should have mental states, explicit goals and knowledge and none of these should be in public domain. The above characteristics of an agent make it ideal for a business logic implementation where it preserves the autonomy of the entity it is representing. In our view, agents present an excellent methodology for an open and peer-to-peer implementation of cloud computing enabled robotics.

An agent based approach to cloud robotics may be more suited for designing intelligent robotic systems because agent

* 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).

Vineet Nagrath is with Laboratoire Le2i, UMR CNRS 5158, Le Creusot, FRANCE (+33666999529; Vineet.Nagrath@gmail.com)

Olivier Morel is with Laboratoire Le2i, UMR CNRS 5158, Le Creusot, FRANCE (Olivier.Morel@u-bourgogne.fr)

Aamir Saeed Malik is with Department EE, Universiti Technologi Petronas, Perak, MALAYSIA (Aamir_Saeed@petronas.com.my)

M Naufal B M Saad is with Department EE, Universiti Technologi Petronas, Perak, MALAYSIA (Naufal_Saad@petronas.com.my)

Fabrice Meriaudeau is with Laboratoire Le2i, UMR CNRS 5158, Le Creusot, FRANCE (Fabrice.Meriaudeau@u-bourgogne.fr)

systems are essentially one of the two subdomains of distributed artificial intelligence or DAI [5]. Distributed problem solving (DPS) is the subdomain of DAI which deals with the distribution of the process of problem solving while multi agent systems (MAS) [6] deals with the interaction and behavior related complexities in a DAI system. A distributed intelligence system reaches a state of intelligence by various simple competitions and collaborations in its members which give equal importance to their personal and collective goals [7]. Multi agent systems are systems of autonomous computational entities with objectives and roles which are specific, and which work in an environment with other autonomous computational entities which may have different roles and objectives [8]. In our opinion, multi agent system based cloud robotics could be a useful construct for the implementation of peer-to-peer cloud computing business model. The infrastructure that a peer-to-peer cloud computing platform may provide will be ideal for scalable and heterogeneous multi agent systems and could lead to a generation of intelligent robotic systems.

II. MODEL DRIVEN ENGINEERING

A model of a system can be simply defined as a set of statements about the system being studied [9]. The statements that form a model of a system should explain the working of the system or describe its behaviour in various scenarios. A meta-model is a model that describes other models, i.e. the system being studied in a meta-model is a model itself [9]. The statements made in a meta-model thus describe how a model has to be made. The engineering approach in which the designs are developed in high abstraction leaving aside the lower level implementation details is known as model driven engineering or MDE. In MDE, the implementation details are specified at a later phase of the development cycle giving an abstract and rapid prototyping of the higher level design. MDE also encourages a greater client participation in the design process as non-technical parties can easily understand and contribute to an abstract model of the system. The software industry and industries where implementation complexities are high, have recently adopted the MDE

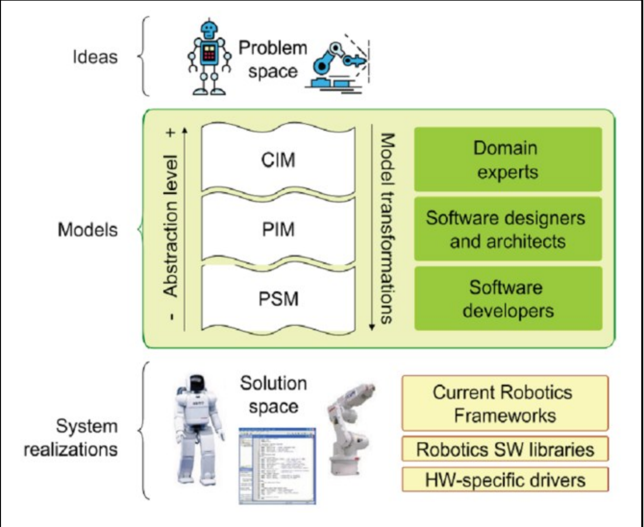


Figure 2. Model Driven Architecture adaptation for the robotic product development in the industry

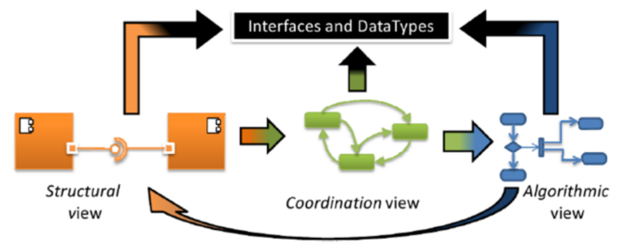
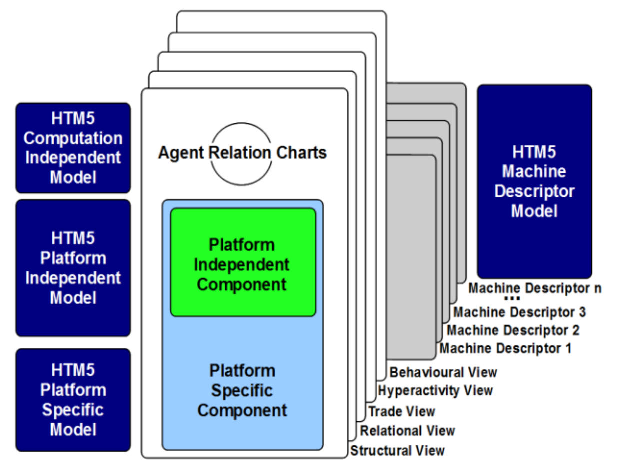


Figure 1. V3CMM: A 3-view component meta-model for robotic software development.

methodology for product development. Multi agent systems in general are an ideal candidate for the use of MDE. Moreover, the implementation level diversities of a multi agent system that is used to represent a peer-to-peer cloud robotic system makes it even more necessary to use a MDE based approach towards its design. In our current work, we have thus decided to take a model based approach towards specification and development of the software system.

In the year 2003, a guide to model driven architecture (MDA) [10] was released by the object management group (OMG), a consortium of computer industry. Since then, MDA has been popular in the industry and is being used for software development of complex systems. The three layers of OMG's MDA offer flexibility and abstraction for different phases and depths in the design process. Platform specific (PSM), platform independent (PIM) and computation independent (CIM) are the three models that specify the three layers of MDA. One of the adaptations of OMG's MDA is presented in the Fig. 1. The example shows how MDA is used for product development in the robotic industry [11]. Ideas from the problem space are converted to a model by the domain experts. This top layer model is computation independent and uses MDA's computation independent model. Later on, the software designers convert CIM to a MDA's platform independent model which is at a lower abstraction layer. In the next step the software developers specify MDA's platform specific model and thus the system model is ready for implementation. This system model is then used to realize the system in the solution space.

Some systems may be so complex that a single model might not give a simplified picture of the system. In such cases, more than one model is used to represent different features of the system. These different models of the same system are known as different 'views' of the system. A multi view system provides better readability of the design and this kind of system modelling is called multi-view modelling. One example of a multi-view modelling is V3CMM [11], a three view meta-model used in software development for robotic systems (see Fig. 2). The structural view of V3CMM is a model for the structural design elements of the robotic system, coordination view models the way the system handles events and the algorithmic view is a model for working logics of system's individual modules. The model being developed by the authors is a five view hyperactive transaction meta-model for multi agent design (HTM5), while the current paper explains the view (sub-model) which specifies the trade behavior of the multi agent system. We present our idea of a peer-to-peer cloud robotic system using the trade-view of the HTM5 meta-model. Section III of the current paper gives a



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

Figure. 3 HTM5: A five view hyperactive transaction meta-model for multi agent system design.

brief overview of HTM5 while Section IV presents the elements and workflow of a peer-to-peer cloud robotic system using the trade view of the HTM5 meta-model.

III. HTM5: A Five View Hyperactive Transaction Meta-Model

HTM5: A five view hyperactive transaction meta-model is a meta-model for designing multi-agent systems (MAS). It is based on the model driven architecture (MDA) guide of the object management group (OMG) [10]. Like MDA (see Fig. 1), HTM5 is a 3 layered model with computation independent, platform independent and platform specific layers (see Fig. 3). The five views (sub-models) of HTM5 are designed to capture different aspects of a multi agent system design and are named structural, relational, trade, hyperactivity and behavioural views.

A. HTM5 anatomy

The structural view models the MAS structure and physical locations of different agents. This view also specifies the hardware (agent hosts) on which a particular agent is hosted, and the connections (kind and name of networks) among various agents. The relational view models agent relationships. Agent interaction and their relative roles in the multi agent system are based on the kind of relationship that exists between them. The trade view specifies the trade of services between agents. The locations of services and the demands associated to those services are specified in this model. The mechanism (e.g. service and demand registry) and data (e.g. cost variables) that governs the trade are also specified in this view. The hyperactivity view models the hyperactivity [12] in a multi agent system, which is an agent's ability to transmit knowledge to agents which are associated to it. Hyperactivity mechanism is a controlled release of an agent's autonomy to provide operational and design flexibility. The behaviour of the multi agent system at various layers is specified in the behavioural view. A step by step sequence of events and its reaction by the system defines the system's respond to a scenario.

In addition to the five main views of HTM5, there are four hyperactivity sub-views (which are placed under hyperactivity view) for capturing hyperactivity in structure, relational, trade and behavioural views. The machine descriptor model (see Fig. 3) for hardware (agent hosts) and agent relation charts (ARCs) [13] (In computation independent layer) are some other components of HTM5. For the current paper, we will limit ourselves to the description of HTM5-Trade view and key elements of ARCs required to present the idea of peer-to-peer cloud robotic implementation using multi agent systems in Section IV.

B. Agents, Merges and Relations

HTM5 model proposes differentiating the agents into categories (see Fig. 4). Agents in HTM5 are entities that represent a software construct (base software) on the cloud. More than one agent may be hosted on a hardware (agent host), and thus a hardware may be represented by more than one agents. HTM5 agent components have a set of control parameters which governs its internal state and decision making logic. A "passive" agent in HTM5 is an agent in which the control parameters are constants while an "active" agent has an update mechanism for the control parameters. The update or activity mechanism of an agent may be based on internal communication with the base software or on the transactions that takes place with other agents in the system. Update of an agent's control parameters based on external transactions does not compromise its autonomy as the update mechanism is internal to an agent. HTM5 proposes the concept of hyperactivity [12] where an agent's autonomy is released through a controlled mechanism. Agents which have a hyperactivity mechanism allowing specific agents (associated agents) to update its control parameters are called "hyperactive" agents. Hyperactivity mechanism gives flexibility to a designer to reduce an agent's autonomy when needed and induce an object like character in an agent.

An HTM5 agent may have managerial functionalities which are meant to keep the multi agent system run smoothly. For example an agent may manage addition or removal of members to the system, spread or combine information coming from different agents. The designer may choose to keep these managerial functionalities of an agent separate from its other responsibilities by placing them in a separate agent. An HTM5 agent dedicated to these managerial functionalities is known as a “merge-agent" or simply a "merge". Similarly, agents dedicated to maintaining a relationship between other agents are called "relational-agents" or "relations". A "relation" agent maintains registry for relationship based trade (collaborations) and holds relationship data parameters. Merges and relations are agents

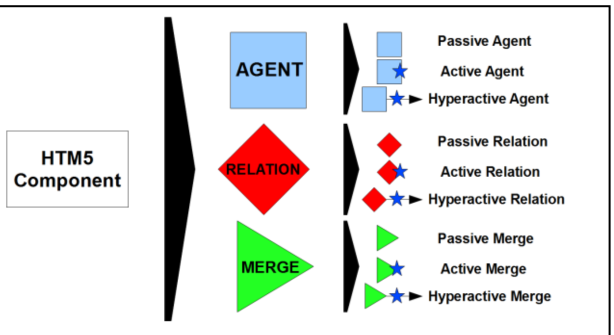


Figure. 4 Different categories of HTM5 agent-components.

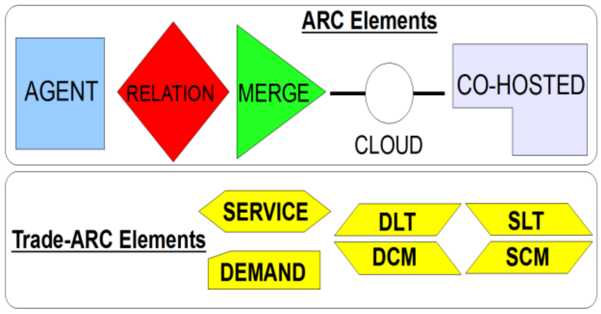


Figure. 5 Design elements of Trade-View Agent Relation Charts

like any other agent in HTM5 based system, and are different only because they are dedicated to specific managerial roles in the multi agent system. Like agents, merges and relations can be passive, active or hyperactive (see Fig. 4).

C. *HTM5 Trade view*

- The computation independent layer of HTM5-Trade view consists of trade view agent relation chart (T-ARC) [13]. Fig. 5 displays elements of T-ARC that are used to model the relation based trade logics of the multi agent system. HTM5 agent components (agents, merge and relations) which are running on the same hardware device are called co-hosted components. A "cloud" is any computer network that establishes a connection between agents. There may be more than one kind of clouds (networks) in the system and HTM5 ARCs represents different kinds of networks by unique numerals.
- A "service" could be any sharable resource that an agent is offering to other agents for use. T-ARC specifies services that are available from individual agents. T-ARC also specifies "demands" that agents have, which are resources that an agent wants to get from other agents in the system. The agents (relation agents in particular) maintain demand and service lookup tables (DLT, SLT). The lookup tables are registry that a relation maintains to facilitate service and demand discovery by other agents. T-ARC specifies the locations where the lookup tables are placed. The service and demand cost metrics (SCM, DCM) are relationship variables maintained by a relation. The lookup tables and cost metrics are open to be used by the designer to store any item (and not just registry and costs) that is required to maintain a relationship. The T-ARC thus offers a flexible toolset to represent the idea as a design model.
- The lower layers of platform independent and platform specific design in HTM5 trade view are a cluster of classes (R: Relational view, S: Structural view, T: Trade view, B: Behavioural view, XH: Hyperactivity in view X) (see Fig. 6) encapsulated together to form a component. The platform independent component contains abstract classes [14] which are later completed in the platform specific layer. Unified markup language (UML) [15] is used to specify HTM5 platform independent and platform specific components.

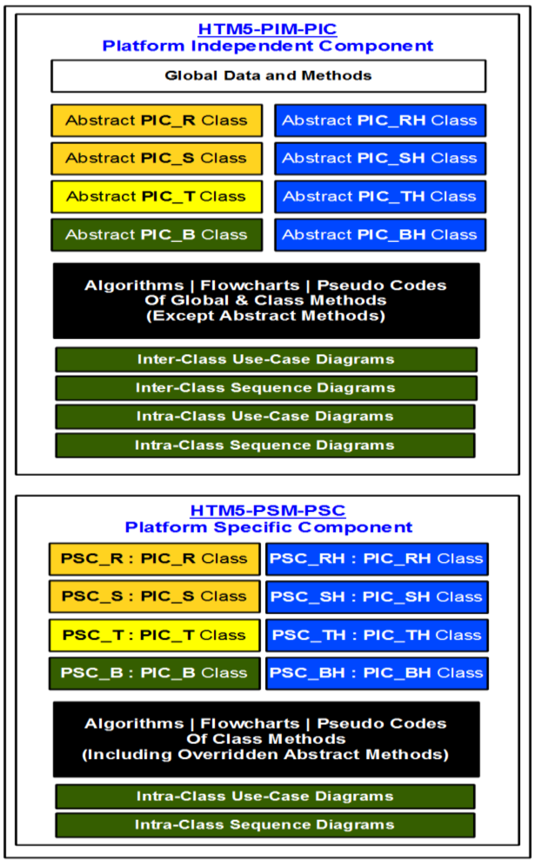

HTM5-PIM-PIC
Platform Independent Component
Global Data and Methods
Abstract PIC_R Class
Abstract PIC_RH Class
Abstract PIC_S Class
Abstract PIC_SH Class
Abstract PIC_T Class
Abstract PIC_TH Class
Abstract PIC_B Class
Abstract PIC_BH Class
Algorithms | Flowcharts | Pseudo Codes Of Global & Class Methods (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

Figure. 6 Class clustering in HTM5 model for platform independent and platform specific agent-component

Details about the structure and format of these lower layers and that of HTM5 machine descriptor model (HTM5-MDM) (see Fig. 3) are not essential for the current paper.

IV. Agent Driven Peer-to-Peer Cloud Robotics

In introduction to this paper we have suggested the benefits of peer-to-peer and agent driven approach to cloud robotics. HTM5 is a general purpose model for multi agent system development, and suited for service oriented systems. In this section we present the idea of peer-to-peer cloud robotics using HTM5's modelling constructs.

A. *System representation*

A cloud robotic system with multiple parties is by definition a geographically and computationally distributed system. It is one of the prime requirements in a model to have a structural representation that locates networks, agents and hardware components. ARCs are computationally independent representations of a system's structure and relationships that exists between agents. Fig. 7 presents an example of two ARCs [13] of a simple peer-to-peer cloud robotic system. Agents, hardware, clouds and managerial agents (merges and relations) are well specified in the model. The names and locations of services and associated demands are modeled in T-ARC along with functional descriptions of merge and relational agents. Lookup tables and cost matrices contains relationship variables used by the relation agents and agents which are a part of the relationship. Fig. 7 is an example of a well-defined closed system and hence it doesn't

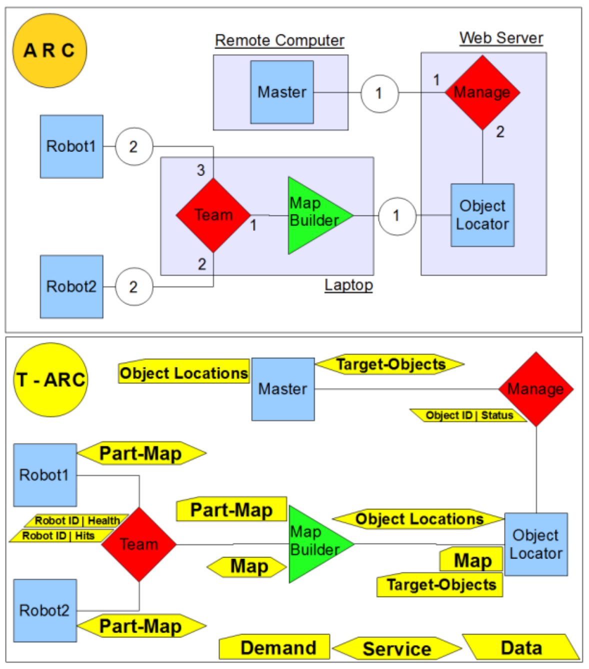


Figure. 7 ARCs for representing system structure, relations and trade structure of a peer-to-peer robotic system.

require a service discovery and matchmaking mechanisms. In a real project, several ARCs may be drawn to form a complete model of the system.

B. Discovery and matchmaking mechanism

We assume most cloud robotics systems will be dynamic with agents having more than one options to get the required services from and where members become online and offline randomly. In such systems there will be a need to have a registry system (like in many service oriented open systems) to publish services and demands of agents. Fig. 8 is an example of one such system where more than one service provider publishes the availability and cost of its services through a lookup table hosted on a relation agent. One possibility for service initiation could be the matchmaking mechanism implemented on the relation. In such scenario it is necessary for the agents to grant authority to the relation agent to decide on their behalf which service provider is allotted to them and on what basis (The relation agent will have to be hyperactive [12] to surpass autonomy of other

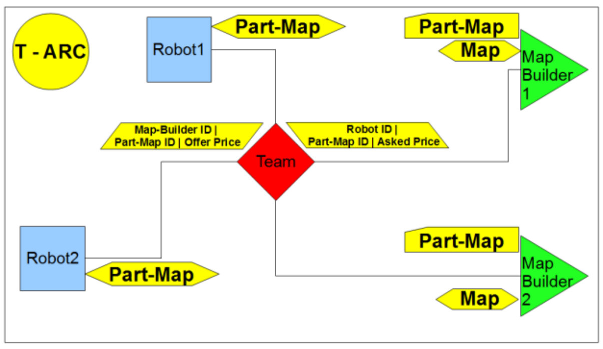


Figure. 8 Trade ARC representing Service discovery and Matchmaking mechanism with Lookup tables hosted on a Relation.

agents). The other possibility could be to allow agents to negotiate the costs of the services. The relation agent in such a case may store and maintain the registry, the trust and quality of service variables of individual relatives.

Agents communicate the updated values (or read requests) for the lookup table entries (e.g. Part-map IDs and Offered/Asked Prices in Fig. 8) to relation agents, as and when required. The relation agent adds/deletes records from the lookup tables when agents become online/offline. Any updates made in relationship variables are reflected in the matchmaking mechanism (on the relation agent) or the negotiation mechanisms (between pair of agents).

C. Relationships and contracts

Relationships are relative positioning of agents into a predefined social construct. In scenarios where we have relation agents dynamically adding or deleting agents to a relationship (via a merge agent, discussed in part D of the current section), the joining agents instantly know their role in a relationship by the kind of relation agent they are attaching to. As each agent has its personal goals and business logic, a pre-defined trade relationships help reduce otherwise chaotic system of agents.

Once agents decide to exchange the services (by matchmaking mechanism or by agent to agent negotiations), the relation agent or the agents themselves may bind themselves to a contract. This is similar to current pay per use cloud computing services available online, the service provider and the user agree upon a cost and quality of service before initiating the service. The quality of service between a pair of agents is monitored by the relation agent. The denial of service by a service provider, or a problem with the quality of service after contract initiation is reported by the client agent and is recorded by the relation agent. Counter Actions could be initiated by the relation agent like termination of the current contract, penalty and/or lowering of the trust parameter for the service provider. Trust parameter of an agent may affect the matchmaking/negotiations for its future deals. Trust is a complex human factor to model, and its implementation may vary from project to project.

In peer to peer (or client server based) cloud robotics, the parties involved in the use of paid services will have to bind themselves in a legal contract. Administrative and banking agents as a part of the cloud computing system could be one of the possible solutions. These measures will be essential in popularizing cloud robotics as a business possibility as relationships and contracts bring reliability and a level of trust in the cloud robotic system.

D. Dynamic Electronic Institutions

The most advantageous step towards bringing peer-to-peer cloud robotics closer to a viable business model, would be to establish mechanism for automated dynamic electronic institution [16] formation. An institution is a representation of a norm based society. Social, business and administrative institutions shape the way humans interact [17]. Electronic institutions are a representation of a norm based multi agent system leading to a sociologically-inspired computing [18]. The norms introduced by electronic institutions or dynamic electronic institutions may be seen as a limiting factor in a system's functionality, but by constraining agent's behaviour they decrease the system complexity and make agent behaviour more predictable.

For dynamic electronic institutions, one proposed lifecycle is a three phase life cycle (3F life cycle) [19]. Formation, Foundation and Fulfillment are the three phases in a digital institution's lifecycle. Agents from an agent community are selected to form a coalition (Formation Phase) followed by establishment of norms of an institution. Once the norms are finalized and a particular kind of institution is chosen, the agents form and participate in the institution (Foundation phase). Re-formation and re-foundation may occur as and when required to keep an institution effective. On completion of an institution's mandate, the institution is dissolved (Fulfillment phase). The 3F approach has one application in agent based business ecosystem development [19]. Cloud robotics is one such ecosystem, and an agent based peer-to-peer approach towards cloud robotics requires an institutional framework to find acceptance as a business model. The digital electronic institution concept can be implemented in HTM5 trade model by using merges for managing formation, foundation and fulfillment phases, and relational agents for re-formation, re-foundation and managerial logic constructs.

V. Conclusion

We have discussed the possibilities of extending the current client server based cloud robotics to a peer-to-peer structure. We believe a real-life cloud robotic system will be a network of computers, web-servers, smart phones, ambient intelligent devices and robots working together as one system. The use of agents to represent robots and other computational units and to treat cloud robotics as a multi agent system ecosystem was the key motivation behind the presented idea. The Agents we talked about could be any entity in the cloud, i.e. any network enabled computational device (Including Robots and Human Actors).

We introduced model based engineering and its growing popularity in the design on complicated software systems including distributed intelligent systems. Elements and concepts behind the five view hyperactive transaction model were discussed with special attention to the trade view. We presented scenarios and mechanisms using HTM5's trade model by which the idea of agent driven peer-to-peer cloud robotics may be actualized. Mechanisms of service discovery and matchmaking, relationship and contract based trade and dynamic digital institutions were discussed with respect to peer-to-peer cloud robotics. We presented a computational independent description of peer-to-peer cloud robotics and suggested HTM5 trade model as one of the tools for its development. No claims were made to present HTM5 as a better or the only meta-model for implementation of the presented ideas. HTM5 is an open and flexible meta-model for multi-agent system development, and in our opinion has necessary tools for cloud robotics ideas presented here.

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] "Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook." Editors: Federico Bergenti, Marie-Pierre Gleizes, Franco Zambonelli ISBN: 978-1-4020-8057-9 (Print) 978-1-4020-8058-6 (Online)

[5] 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.

[6] 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.

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

[8] 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.

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

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

[11] 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.

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

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

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

[15] "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

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

[17] Douglass C. North, "Economics and Cognitive Science," Economic History 9612002, Economics Working Paper Archive at WUSTL, 1996.

[18] D. Hales and B. Edmons, "Sociologically Inspired Engineering". In AgentLink News, Agent Research Overview, pp 11-13. September 2004.

[19] Eduard Muntaner-Perich, Josep Lluís de la Rosa Esteva, "Using Dynamic Electronic Institutions to Enable Digital Business Ecosystems", Coordination, Organizations, Institutions, and Norms in Agent Systems II, Lecture Notes in Computer Science Volume 4386, 2007, pp 259-273