Towards Development of Agent Class Diagrams as an Integrative Approach for AUML Extension

GHULAM ALI, NOOR AHMED SHAikh, ABDUL WAHID SHAikh
Department of Computer Science
Shah Abdul Latif University, Khairpur
PAKISTAN
ghulam.ali@salu.edu.pk, noor.shaikh@salu.edu.pk, awahid.shaikh@salu.edu.pk

Abstract: - The relatively new concepts of agents and multi-agent systems have emerged as a promising approach to simplify the intricacies of developing huge and complex software applications. However, methodologies for effectively applying agent technology in such complex applications are still under development. Most of the well-documented specifications wander around the two same underlying research directions: Reuse and extend UML or develop a new methodology from scratch. This paper strive to facilitate the developers and researchers, interested in multi-agent systems, to make an effort for designing effective intelligent agent systems by using the semantics of Agent Class Diagrams, proposed in this paper as an extended version of AUML Class Diagrams.

Key-Words: - Agent, Multiagent systems, UML, FIPA, Class Diagram, AUML.

1 Introduction

Multi-agent System (MAS) is a group of agents interacting and collaborating with each other, while being possibly engaged in a shared activity. In such environments, the communication among agents is done with the help of a standard Agent Communication Language (ACL). ACL delivers structured messages in an asynchronous manner. MAS are extensively recognized as being suitable for service-oriented information processing such as applications of web services, where systems are constructed from a number of components dispersed over the Internet. However, methodologies for effectively applying agent technology in such areas are still under development [1]. Analyzing, designing and implementing systems as a collection of interacting and autonomous agents represents a potential point of departure for conventional software engineering. The relatively new concepts of agents and multi-agent systems have emerged as a promising approach to simplify the intricacies of developing huge and complex software applications.

Installing Agent-oriented technology successfully in the development of industrial scope applications has to be supported with industrial-strength methods and consortiums, like the UML specification for Object-Oriented software development. The current UML specification lacks modeling constructs for most of the core agent related concepts like, goals, behaviors, dynamicity and multiplicity of roles, communicative acts etc. We can have following viewpoints of work being done towards Agent-Oriented Software Engineering [2].

- Extending UML while introducing new profiles,
- Extending existing UML profiles to include new constructs, or
- Working towards a new methodology or set of specifications

We opted to go for extending the UML, while becoming the proponents of the AUML’s definitions. The two advantages that are associated with AUML’s approach are [3]

- The approach is an incremental one, which extends known and trusted technology,
- Integration with explicit engineering tools that support industry-accepted methods of technology deployment

2 Meta Object Facility

The Meta Object Facility provides set of metadata services and a metadata management framework to facilitate the development and interoperability. The World Wide Web or Internet entails numerous data sources, located across the globe and connected within a huge and complex network. In most of the situations these data hubs connected to the internet are required to share and exchange data amongst them. The foremost problem that hinders the process
is the incompatibility of protocols of exchange, and standards of storage. Hence a need was felt for such a consortium, which should standardize the set of services and frameworks to be followed by all. Meanwhile, conforming to this metadata-driven interchange the OMG group adopted this MOF standard for its technologies, like UML, CWM, SPEM, XMI and various other profiles. The latest specification of MOF is v2.0. It is a result of a thorough revision of the previous version, and provides a platform independent metadata framework for OMG. The significant factor that is eminent after this release is the successful attempt to unify the modeling concepts in MOF2.

Now coming back to context of this paper, we found a fundamental yet misperceived concept, without which, the extension seemed to be impossible. Primarily, it is a misconception that OMG is bound by a “Four-Layered Meta-model architecture.” So stated simply, we are using the flexibility of MOF and UML to use packages, while having any number of Meta-Layers; making the process independent of any extraneous constraints [4].

3 OMG UML

The Object Management Group (OMG) is a non-profit group to produce computer industry specifications for interoperable enterprise products and applications. It has adopted an under-growth conceptual architecture named, Model Driver Architecture (MDA) that will promote a model-driven implementation methodology in typical development of the software.

UML was created with a broad scope, ensuring applicability in a large and diverse set of application domains. Though, the OMG admits its usability in numerous development environments but not all of the environments and hence they blessed the specification with a concept of compliance levels. Altogether, there are four compliance levels, according to the increasing level of capability. At the very beginning it is an empty UML package, i.e. combined with Basic Package of UML Infrastructure that contains fundamental UML concepts like Class, Package, Data Type, Operation, etc. Figure 1 is the package diagram at level 0.

Finally, Level 3 (figure 2) incorporates the full UML definition. The notable point is again the flexibility integrated within the underlying structure of UML, i.e. the ease of UML extension based on packages. Hence to substantiate the need of extending the

![Figure 1: Level 0 package diagram](image)

4 FIPA & UML

FIPA is standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. Anyways as stated previously the task of successful deployment of Agent-Oriented Software Engineering paradigms in general is dependent on two factors.

a) Extension of known and trusted methods/technology
b) Explicit engineering tools that support industry-accepted methods of technology deployment.

Most of the stakeholders of software development and engineering community see multiagent systems as extension of the Object-oriented (OO) paradigm, since it is natural to look to the tools normally associated with OO development when designing agent systems. UML was developed with a task of incorporating all characteristics related to the OO methodology. However, by observing various documents and discussions we realized that by migrating from objects to agents, we can exploit certain capabilities and features that are not observable within the UML Standard. Hence this need was felt by FIPA and consequently an agent-based unified modeling language (AUML) is being developed [6].

So apparently, this study is limited to observation of any documents and propositions that are consistent with the first factor, i.e. Extension.
Figure 2: Level 3 top-level package diagram

Figure 3 Agent Class Diagrams Abstract Syntax
5 Agent Class Diagrams
Two documents have been officially released by AUML Group:
  a) FIPA Modeling: Agent Class Diagrams
  b) FIPA Modeling TC: Agent Class Superstructure Meta-model

Agent class diagrams are based on UML Class diagrams, which are also known as static views of a system. To avoid difference between an object and an agent, Agent Class Diagrams were developed to include any classifiers that are essential for modeling of Agents and their interactions.

There are various types of agents, and hence the most difficult part is the complexity that hinders in identifying the common characteristics of all agent types [7].

5.1 Agent Classifier
An agent classifier categorizes the set of agents on the basis of common characteristics. It broadens the idea of a UML Classifier and therefore explains followings:
  a) It is a Namespace (from Kernel) whose members can include features.
  b) It is a Type (from Kernel), thereby making it possible to define generalization relationships to other Agent Classifiers.
  c) It is a RedefinableElement (from Kernel), meaning that it can be redefined more specifically or differently in the context of another classifier that specializes (directly or indirectly) the context classifier.

5.2 Relationship between Agent Classifier and Agent
The primary modeling elements for agents based systems are agents and agent classifiers. Agent Classifiers provide the core features as well as roles for all agents in the system. Because Agent Classifier is an abstract class, hence its instances may be either of Agent Role Class type or Agent Base Class type.

There is a many-to-many relationship between Agent Classifier and Agent. The second part of the statement might be confusing for some readers but it is evident from the fact that to completely describe an agent, both its basic features and its roles should be defined. The basic features will be described by an Agent Base Classifier and roles will be defined by Agent Role Classifier. Therefore each agent can have more than one classifier.

6 Challenges and Future Research directions
Being proponents of agent-oriented software engineering, we anticipate a gradual departure of conventional object-oriented approaches that were extensively used in certain systems (mostly distributed applications); despite of the deficiencies and lack of capabilities of UML. This need as we observed and documented, has been felt and consequently numerous methodologies and specifications and have proposed, as well as maintained. However enough work has to be done to unify all the underlying concepts and definitions, while striving to remain consistent with the constraints of industrial software development. Above all, there is a need of a consortium, which should be acceptable by all the stakeholders of the industry as well as the huge community of professors and students researching on development of agent-based systems. In short, whatever would be the part of the system must ensure a wide applicability in software engineering practice. Secondly, we are required to turn agent-oriented software abstractions into practical tools and utilities for facing the complexity of modern application.
areas [8].

7 Conclusion
OMG UML Specification was a resultant product of a unifying process, which synthesized all the relevant work done by numerous personals and organizations. Undoubtedly, its evolution was the most complex responsibility ever taken by the software development community. The difficulties involved resolution of conflicting specifications and proposals, multiple viewpoints, and biasness of ideas. When we introduced ourselves to this study of agents and their applications, we were overwhelmed with a huge indefinite number of proposal and work issues. The development of agent-oriented software engineering consequently and within small frame of time has been entitled and reproduced in those diverse frameworks and methodologies. Although only few of them were seemingly inconsistent or not properly presented, but most of the well-documented specifications wander around the two same underlying directions:

• Reuse and extend UML, or
• A new methodology from scratch.

With all the knowledge we developed and the conversations amongst the community members that we observed, we decided to remain persistent with Agent UML. One can argue over the fact that to gain popularity and adaptability, the AUML group or its proponents are biased over extending the most extensively used standard, i.e. UML. But we found out that UML is not a specification that was produced by a few personals, rather it is a fusion of concepts and practices that were adopted prior to its first official release by OMG.

Henceforth, within the context of static views for Agent-based systems, we compared the proposition of Agent-Class Diagrams with others, and reproduced the same specification with a few relevant changes. We proved the consistency of the additional classifiers within the AUML Class Diagrams as relevant by simulating a case study, based on the semantics of the model package.

References:


