Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 3 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Software Agent Reusability Mechanism at Application Level

By Deepti Aggarwal & Aarti Singh
*Chandigarh University, Gharuan*

*Abstract -* The usage of already available software agents plays a vital role in the process of development of application specific software. This reuse also leads to software development cost benefits as well as may ensure the timely delivery. This paper lay down an idea that for reusing reactive multi-agents systems two factors are to be considered i.e. (i) abstract description of agent in application independent way and (ii) reuse of such systems through adoption in specific domain[25]. For such a development main requirement is the effective reusable software abstractions. In present study the role of abstraction level and dependence level is analyzed for intelligent agents.

*Keywords : software reuse, software agents, software abstraction.*

*GJCST-C Classification : D.2.13*

SOFTWARE AGENT REUSABILITY MECHANISM AT APPLICATION LEVEL

# Software Agent Reusability Mechanism at Application Level

Deepti Aggarwal [α] & Aarti Singh [σ]

*Abstract -* The usage of already available software agents plays a vital role in the process of development of application specific software. This reuse also leads to software development cost benefits as well as may ensure the timely delivery. This paper lay down an idea that for reusing reactive multi-agents systems two factors are to be considered i.e. (i) abstract description of agent in application independent way and (ii) reuse of such systems through adoption in specific domain[23]. For such a development main requirement is the effective reusable software abstractions. In present study the role of abstraction level and dependence level is analyzed for intelligent agents.

*Keywords : software reuse, software agents, software abstraction.*

## I. Introduction

The reuse of software components refers to the process of using software artifacts for the development of new software. Since the evolution of computational systems, the reuse of software is in practice. Reuse of the already existing code to develop new software or developing the software that can be reused is one of the prominent areas of research. The present study mainly focuses on the intricacies involved in reuse of multi agent software systems.

A **software agent** is a computer program that symbolizes a user. It implies that an agent has the capacity to make the appropriate decisions.

Using software agents for Domain-oriented component design method is a newly proposed reuse approach in software engineering and it starts from the process of acquiring business knowledge within a common application domain. After having the knowledge of the application domain, a collection of business logic is mapped into components which can be reused in the future deign. This method increases the functional completeness of the software component and makes it reusable to a lower extent. Since the business requirements of different organizations are diverse and are changing, so a reusable knowledge base that can be adaptive and flexible are yet to be provided by current domain component design methods. In the following section the major design issues of the reusable software components are addressed.

*Author α : Assistant Professor, University Institute of Computing, Chandigarh University, Gharuan, India.*
*Author σ : Associate Professor, M.M. Institute of Computer Technology & Business Management, M.M. University, Mullana, India.*

## II. Design Issues

The major issues to be considered in the development of agent-based software systems include [3], [11], [12].

- Scheduling of tasks and their synchronization
- Prioritization of tasks by the agent
- Assignment of tasks by the agents
- Representation of agents in different environments, and storage of their internal state
- For heterogeneous platform what are the Behavioral changes of the agents
- How message passing can be facilitated and communication can be established
- Usage of hierarchies of agents

Apart from the above stated issues following issues are critical issues in reusability of the software agents:-

- Usage of software agents on diverse platforms
- Sharing or reconstructing ontology for software agents being reused

## III. Software Abstractions

The abstraction of software artifacts has to be used in every method of software reuse. A software abstraction is high-level, in the sense that attributes corresponding to one or more realizations of facts in more detailed level are represented. Some attributes describe what and how abstraction is done [19]. For the clear understanding, comparison and selection of appropriate software artifacts, the small abstractions are needed and these abstractions can be used in reuse process. The clear understanding of user interface has to be there so that a set of software artifacts can be composed and this should be depicted in the abstraction specification. Every artifact plays an important role in application development and it may not be concluded that the final deliverable i.e. the software product satisfies the user requirements or not.

An abstraction is composed of a fixed, a variable and a hidden part. Only the fixed and variable part is visible in the abstraction specification. The fixed part represents the invariant features and the variable part symbolizes the variable features of the abstraction. The hidden part consists of the realization details.

The ***cognitive distance*** is the measure of the effectiveness of the reuse technique. It is the effort required to transfer software system from one phase to another in terms of intellect. The goal of the reuse technique should be to minimize the cognitive distance between the abstraction and the final software product. To minimize this distance the abstraction specification should more specifically represent the abstractions which are used for application domain. Finally the mapping of the specifications should be made partially or fully automated [10].

## IV. Multi-Agent Application Engineering

"Multi-Agent Application Engineering" is a domain oriented research towards reducing software complexity and increasing productivity. This can be accomplished via techniques and tools that aid software reuse in Multi-Agent Software Engineering.

In Application Engineering, the software abstractions that can be reused are created. Application Development is the process of developing domain specific applications using software abstractions that can be used again and again [12].

In the following section the model for developing agent based application specific software is presented. A set of activities and various tools and libraries that can be used is also discussed for developing high level software abstractions.

### *a) Developing Multi-Agent Specific Applications*

A multi-agent specific application is made using the constitution of a group/assortment of reusable agent frameworks available in the library of the development environment as shown in Figure 1. These frameworks are realizations of high-level software abstractions in the library.

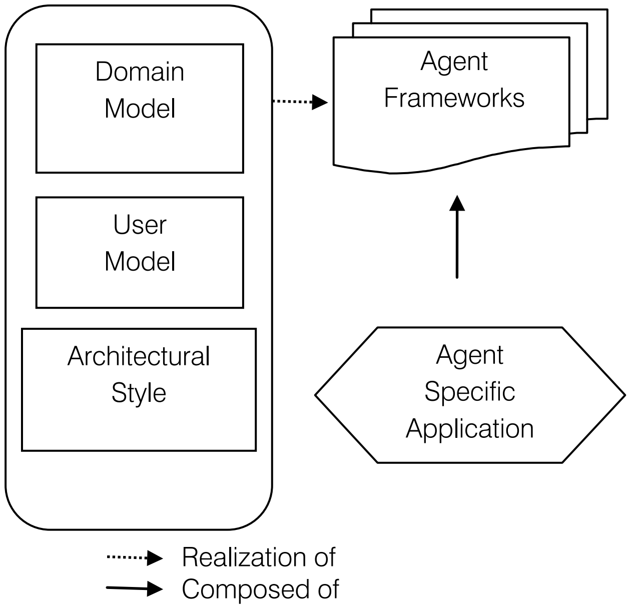

*Figure 1 :* A model for Agent-based Application Development [23]

Particular requirements of a multi-agent specific application are used, to map the specification level of a domain model into a realization satisfying such needs. The realization should have associated a set of frameworks, which are agent-based solutions to those requirements. Requirement analysis in Agent-based Application Development should also consider particular preferences of users of the multi-agent specific application. Therefore, these user profiles are used to map the specification level of a user model into a realization satisfying these preferences or user needs. The realization should have associated a set of frameworks, which are agent-based solutions to those needs. The choice of mapping to select best frameworks depends on the fact that which particular style of agent architecture is to be used for the design.

### *b) Developing High – Level Software Abstractions*

The reusable agent-based software abstractions are illustrated and described considering their level of abstraction and their dependence level from the application or user domain: domain models, user models, agent-oriented architectural styles, agent-oriented design patterns, agent-oriented frameworks and software agents.

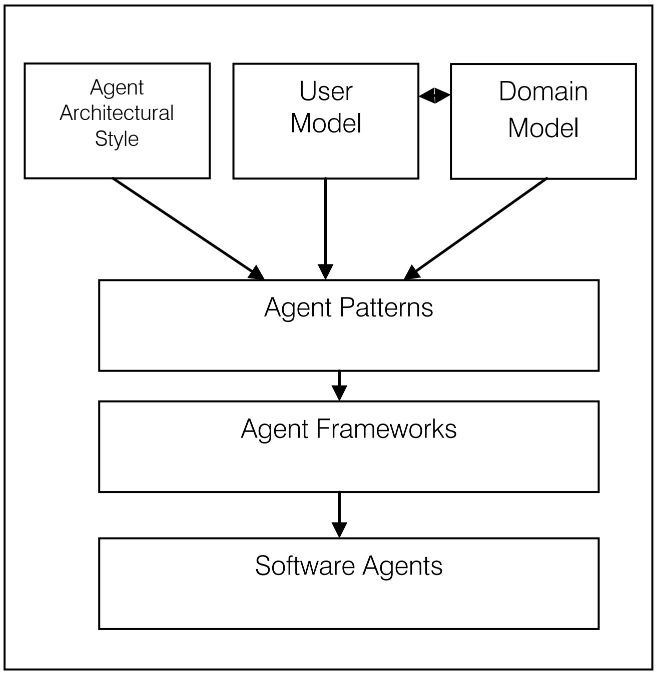

*Figure 2 :* Agent Based Application Development Process

### *c) Requirement Analysis*

The next phase is to extract the requirement specifications of the domain of the application. The application domain, the user model and the interactions of domain model and the user model result in a reusable software product.

The domain model specifications are depicted at a high level of abstraction. This model represents the conception of problem. The language represents the definitions necessary for elements, processes and their relationships in the system. The user model specifies the needs and requirements of the end user.

*d) Design of the Model*

The outcome of the design phase is the reusable design attributes of Agent – Based Application Engineering. It consists of the agent based architectural styles, software patterns and the frameworks.
Architectural style is the set of designing rules which will specify the type of elements and coupling which can be used to constitute the system and subsystems.

The software pattern identifies and specifies the problems can commonly occur at conceptual and architectural level. These problems generally originate from architectural styles being used, domain model and the user model. A software pattern has a set format so that it can be easily propagated. This format asserts the problems that have been depicted by the pattern and forces the action to be taken to resolve the problems. For making use of such pattern there should be a framework that validates the pattern and provides a probable solution to the problem. If it was previously used in some application, that has to be mentioned in the framework. Some agent based software patterns have already been proposed that can be used at architectural level or later stages in the agent design [2, 17]. The basic design guidelines are provided by the architectural styles and the agent patterns so that agent oriented frameworks can be developed [3, 5, 20]. The participating agents are chosen from the bank of agents which facilitates domain dependent or domain independent functionality.

## V. Conclusion

A new horizon of reuse based software engineering using agent paradigm has been introduced. With the introduction of one more layer of abstraction at the software level, the present study may be used for the development of reusable software components across various platforms. Thus it proposed an effective model for agent based reuse.

## References Références Referencias

1. Alves da Silva Filho, J. H. and Girardi R. (advisor). (2001). SIMCAP: A Multi-agent System for Filtering Scientific Publications from the Web.
2. Aridor Y. (1998) Agent Design Patterns: Elements of Agent Application Design. In: Proceedings of Autonomous Agents 98.
3. Bradshaw J. M. (1997), Book on Software Agents. Published by The AAAI Press, 1997.
4. Brugali D. (2000) Towards Agent-oriented Application Frameworks. ACM Computing Surveys. Vol. 32 no. 1.
5. Burmeister, B. (1996) Models and Methodology for Agent-Oriented Analysis and Design. In: Proceedings of KI'96 Workshop on Agent-Oriented Programming and Distributed Systems.
6. Diniz, A. and Girardi R. (advisor) (2001) ABARFI: An Agent-Based Software Architecture for Information Access and Filtering.
7. Fleming M. (1999) User Modeling in the Design of Interactive Interface Agents. Proceedings of Seventh International Conference on User Modeling. Pages 67-76 Springer – Verlag New York Inc. Secaucus NJ USA.
8. Girardi R. (1997) Software Architectures to Improve the Effectiveness of Reuse Techniques. 8th WORKSHOP ON SOFTWARE REUSE (WISR8).
9. Girardi R. (1996) Towards Effective Software Abstractions for Application Engineering. Position Paper, NASA Focus on Reuse Workshop, George Mason University, Fairfax, Virginia, Sept. 24-27, 1996.
10. Girardi R. (1992) Application Engineering: Putting Reuse to Work In: Object Frameworks. Dennis Tsichritzis (ed.) University of Geneva.
11. Gopal Shankar, V.M. Thakare (2009), A Road Map of Autonomous Software Agent Architectures. Proceedings of 3rd National Conference INDIA Com-2009, Computing for Nation Development Feb 26, 27 2009, BVICAM New Delhi
12. Hyacinth S. N. (1996) Software Agents: An Overview. Knowledge Engineering Review, vol. 11, no. 3, pp. 205-244.
13. Jennings N. R. (2000) On Agent-based software engineering. Artificial Intelligence, 117, pp. 277-296.
14. Jennings N. R. (1998) A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems. Vol.1 No.1. Page No.7-38.
15. Kendall, E. A. (1998). Patterns of Intelligent and Mobile Agents. In: Proceedings of Autonomous Agents' 98.
16. Krueger C. W. (1992) Software Reuse. ACM Computing Surveys, Vol. 24, N. 2.
17. Lima Costa Ferreira, S. and Girardi R. (advisor) (2002) Agent-based Software Architectures in Information Access Systems. Final Degree Dissertation, CGCC-UFMA. (In Portuguese)
18. Maamar Z. (2000). An Overview of Agent-based Framework. ACM Computing Surveys, Vol.32, n. 1.
19. Ribeiro de Oliveira, I. and Girardi R. (advisor), (2001). An Analysis of Software Patterns for Agent-based Development. Final degree dissertation, CGCC-UFMA. (In Portuguese)
20. Rosario Girardi, Reuse in Agent – Based Application Development.
21. Shaw M. (1996) Software Architecture: Perspectives on an Emerging Discipline. Prentice-Hall.

22. Sodre A. and Girardi R. (advisor). (2001) MADS: A Multi-Agent Software Development Methodology. Master dissertation CPGEE-UFMA. (In Portuguese).
23. Tahara Y. (1999). Agent system development method based on agent patterns. In: Proceedings of International conference on Software Engineering, p. 356-367.
24. Valente, B. and Girardi R. (advisor). (2001). An Experience with Agent-based Application Development. Final degree dissertation. CGCC-UFMA. (In Portuguese).
25. Wood, M. (2000) An Overview of the Multi-agent Systems Engineering Methodology. In: Proceedings of the First International Workshop on Agent-Oriented Software Engineering (AOSE-2000).
26. Wooldridge, M. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems, v. 3.