# A Perspective Study of Intelligent System for Component based Development

Asif Irshad Khan
Dept. of Computer Science,
FCIT,
King Abdulaziz University,
Jeddah, KSA

Mohammad Shariq
Dept. of Computer Science,
FCIT,
King Abdulaziz University,
Jeddah, KSA

Md. Mottahir Alam
Faculty of Engineering,
King Abdulaziz University,
Jeddah, KSA

## ABSTRACT
Recent developments in the industry show strong inclination of Architects towards agent based software development and component based development. Both these approaches help organizations to utilize the older and experienced programs and interfaces into new products without having to reinvent the wheel; thereby reducing cost and time of production and ensuring high quality with already tested components and interfaces.

Nowadays, researchers envisage an Intelligent Component-Oriented Software Development methodology which is an amalgam of the two approaches resulting in more flexible, reusable and customizable agent components. This helps in pushing forward the development timelines and quality expectations to newer heights. In this paper we mainly analyzed various states of art intelligent component-oriented software development techniques and studied the research gap in the component selection processes. Recommendations for future research direction for Intelligent Component-Oriented Software Development are also highlighted in this paper.

## Keywords:
Multi-agent control, Component Based Development, Agent-based modeling, Self-adaptive systems

## 1. INTRODUCTION
Software Industry in the present Information Technology era, has enormous pressure of meeting the product deadlines with minimum development time and minimum development cost. Reusability of software is an important prerequisite for cost and time-optimized software development.

More and more software companies are adopting Component-based Development (CBD) methodologies to meet the demands of customers to deliver, change faster and at a lower cost. Component-based software engineering (CBSE) is used to develop/assemble software from existing components. Some of the advantages that a company may avail by opting CBD for the SW development are cost effective and meets the tight deadlines[1].

CBD is time saving and productive as software are built by integrating already developed components instead of writing from scratch by using state of art tools.

CBD technologies comprised of implementing a component into a system through its well defined interfaces

[2]. Using well-defined interfaces, a component interact with other components to accomplish a partial function of

the system. The inner structure of the component and the implementation of the interfaces are hidden to the outside. Therefore, CBSE enables a distributed and independent development of components as well as a straightforward replacement of a component by a different component in large-scale systems [3].

Lego, as shown in fig 1, is often taken as an example of a component-based approach. Lego provides a set of building blocks in a large variety of shapes and colors. Lego is sold in boxes that contain a number of blocks that can be composed to make up toys such as cars, trains and airplanes [4].

The paper is organized as follows: Section 1 Describes Introduction of the paper, Sections 2 Discuss Open problems in Interoperability for Component Based Systems, Section 3 Discuss Software Agent and Multi-agent system, Section 4 Multi-agent System Engineering (MaSE) methodology, Section 5 Highlighted related work in this area, Section 6 Discusses challenges related to modeling multi-agent system, Section 7 Conclusion and Future Work.

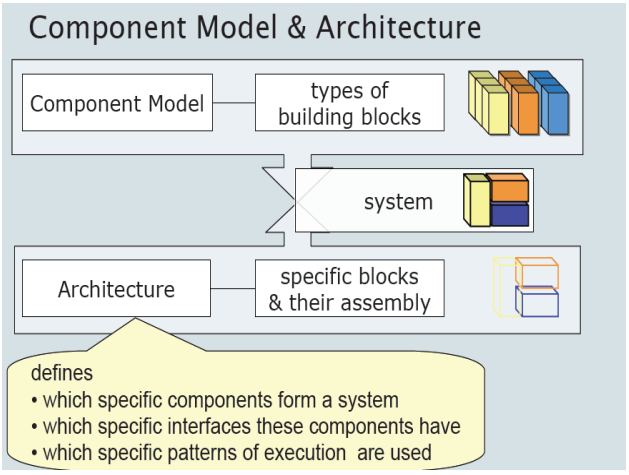


**Fig 1: Concept of Component-based software engineering [5]**

## 2. OPEN PROBLEMS IN INTEROPERABILITY FOR COMPONENT BASED SYSTEMS [6]
It is very difficult to guess how the components behave under different conditions and environments as mostly COTS software comes up as a black box with limited access.

It is also difficult to map user requirement to the component based architecture and generally there is a need for a process which fully customized the component as per the customer requirement.

In order to developed application from components or tailor components to a new situation, efforts are required to build wrappers and the glue between components, since most of the COTS software lacks in plug and play technique and developer has to build wrappers for component integration.

Further, wrappers need to be maintained as the system evolves.

Components are packaged and delivered in many different forms (example: function libraries, off-the shelf applications and frameworks). [6]

Component framework offer varying features (example: component granularity, tailorability, platform support, distributed system support, interoperability). [6]

Most component integration processes suffer from inflexibility by a lack of component evaluation schemes. This problem is often compounded by a lack of interoperability standards between component frameworks and adequate vendor support.

## 3. SOFTWARE AGENT (SA)

Software agents are a software component mainly built to interact with its environment and other agents if needed to accomplish assigned task. For example agent can be used to know the best air fare for a particular route, the job of the agent is to travel to different airlines sever and bring the best price for a particular route.

Software agents offer greater flexibility and adaptability than traditional components, rapid integration of distributed agents provides opportunities to build software systems. Developers can easily develop High-level, flexible enterprise application with the help of agent-oriented software engineering [7].

A software component can be act as an agent if it contains a combination of several of the following characteristics, as shown in Fig 2: [7]

• **Adaptable**: Agent changes his behavior after its deployment based on certain condition, it may be its own learning, user customization, or download new capabilities.

• **Autonomous**: They can act on its user's behalf, mainly independent of messages other agents send, it has its own thread of control.

• **Knowledgeable**: The agent should be smart enough to acquired information, and knowledge about other agents and users to accomplish its goal.

• **Mobile**: The agent should have ability to move from one network to other or among the same network from one PC to other to accomplish his assigned goal.

• **Collaborative**: The agent should interact with other agent to form multi agent societies or work cooperatively with other agents to perform a task.

• **Persistent**: To adopt robustness and retain knowledge in case of any possible runtime failures.

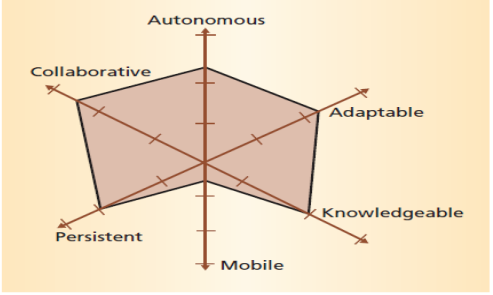


**Fig 2: Components as an agent [7]**

## 4. MULTI-AGENT SYSTEM (MAS)

MAS comprises of several intelligent autonomous software agents who can communicate, cooperate and interact with each other in their environment to solve problems or common goal or common target that are beyond the individual capacities .

Agents are sited in an environment and possibly have knowledge of other agents. The RoboCup challenge is an example of the current state-of-the-art of multivalent systems, in which teams of autonomous agents compete in a simulated soccer tournament. [8]

### AGENTS COMMUNICATION

In Multi-agent system (MAS) communication among the agent is a key to success in many scenarios as communication ability of agents let them share knowledge or request for knowledge from other agent for example an agent may ask another agent for a particular arrival of a vehicle at that agent location so as to dispatched other vehicle from his location.

Communication may be direct with one another or through an interpreter, communicate is usually took place through a language, Knowledge Query and Manipulation Language (KQML) is the most widely used agent communication language (ACL) [22]. Shared vocabularies of words are used in communication which is also known as Ontology. To ensure that two agents are communicating in the same language KQML uses ontologies.

KQML provides a framework for a set of independent agents to communicate and cooperate on a problem using messages called per formatives [8].

– Directives: commands or requests

– Representatives: facts or beliefs

– Commissives: promises or threats

### AGENTS CO-OPERATION

In Multi-agent system (MAS) co-operation among agents is another key factor for success. Individual agent objective is irrelevant in MAS as agent may not cooperate to other agent to achieve the target. It is assumed that agents co-operate each other since all agents want to achieve the same objective. Co-operation among agents allows a community of specialized agents to pool their capabilities to solve large problems [22].

Dividing a problems into sub-problems is also possible in many cases, since every agent have his ability and problem solving capability, a agent cannot solve any problem given to him, he can solve the problem which he is specialized, so, it is always better to divide the problem into sub-problems which further divide into smaller problems so that single agent can solve the assigned problem according to their specific ability in solving large problems.

Co-operation also known as co-operating agents have been applied in the areas of Distributed systems management, electronic commerce and multi agent design systems [22]. Dividing problems into sub-problems so as to specific the problem according to the agent ability raises the following questions [22].

Question: how to choose an agent which is appropriate or suitable to a sub-problem or sub-take? [22]

Question: how to know when the agent completed the task and share the result/

Contract Net Protocol (CNP) is used for assigning the task to agents in Multi-agent system. There are two types of agents in CNP, Agents how has a problem and wish to take the help of other agents, and Participants agents who wish to solve the problem of the above agent, and usually following steps are required in CNP algorithm:

(1) The staring agent who has a problem publicize the problem to all the agents

(2) The participant agents bid for the task

(3) Finally, the starting agents award the task to one of the bidding agent based on his knowledge, suitability and qualification as per the problem.

Result sharing- since the problem is divided into sub-problems it is necessary for agents to share the result, as, some agent depends on the result of other agent to accomplished their assigned task. Usually sharing results is based on two ways

(a) **Proactive:** In this result sharing agent share the result when he believes that other agents may required the result

(b) **Reactive:** In this result sharing agent only share the result when he is asked to share the result.

**AGENTS COORDINATION**

Sometime two or more than two agents in a multi-agent environment depends on each other to accomplished a given task, in such cases coordination relationship is a key to success.

In most of the solutions multi agent based system required agents to coordinate among themselves to reach to the solution of a given task, successful coordination among the agents may result in improve system efficiency, assigned tasks completed in time and proper usage of resources, while on the other hand lack of coordination may result in reduce system efficiency, incomplete assigned tasks and improper usage of resources and system failure [22].

It is very clear that in some situations coordination among the agents is unavoidable, if we consider game Basketball which is a team sport, A team sport is an activity in which a group of individuals, on the same team, work together to accomplish an ultimate goal which is usually to win.

The objective being to shoot a ball through a basket horizontally positioned to score points while following a set of rules. Usually, two teams of five players play on a marked rectangular court with a basket at each width end. A team of basketball players is more likely to win the game with better coordination. [22]

Coordination relationship may be positive as well as negative. Positive coordination relationship benefits both the agent by working together to reach to their assigned goals for example suppose agents are coordinating to switched on a machine if they found machine is off any one agent can switch on the machine to accomplished the common goal (to switched on the machine) , while negative coordination relationship agents cannot complete their assigned task at the same time, for example agents are coordinating to print some assigned job, both of them issue print command but one agent command will be accepted others put in the printer queue.

It is very important from the research point of view to explore this area so as to understand:

What are the possible general coordination problems among the agents in a multi agent based systems?

What are the possible strategies of selecting best coordination mechanisms in dynamically changing environments?

## 4.1 Challenges Of Multi-Agent System (MAS)

Sycara et al. [10] described the following six challenges of multi agent systems

1. How to decomposing problems (Sub task) and allocate tasks to individual agents according to their ability?

2. How to control agent Coordination and communications?

3. How multiple agents act in a coherent manner?

4. How to share result to other agents and the state of coordination?

5. How to reconcile conflicting goals between agents?

6. How to engineering realistic multi agent systems?

# 5. RELATED WORKS

The main objective of component based software is to reduce development costs and efforts, component based software development (CBSD) brings flexibility, reliability, and reusability in software development as components use to build the system already tested and validated in other systems[24]. CBSD approach moves software industry from developing application from scratch to application assembly. Interoperability is one of the key issues of building applications from reusable components is interoperability [6]. Interoperability means ability of two or more entities to cooperate and communicate despite of their different execution environment and implementation language.

System can be model using mathematical constructs such as sets and functions and Model-based techniques are usually used for these purposes. The formal specification of the system can be expressed in term of model-based specification. The use of formal methods for proving “semantic correctness” of components in complex applications remains an active area of research. [6]

Component customization, Selection and integration are the areas of component based software development (CBSD)[23], in the implementation phase of CBSD wrappers are used as a glue to integrate components to make a cohesive system rather than coding from scratch. For greater flexibility and adaptability within this context Software Agents is right choice. High reliable distributed application can be represented and customized with the use of Agent-oriented SE as it provides developers high-level flexible abstractions. [12]

To store and manage reusable components in the Enterprise Java Beans (EJB) architecture, the author [13] proposed a component-based Repository model. [13] Listed many benefits of their component-based Repository model. For example component requirement viewing, adaptation, testing and deploying.

Hutchinson et al. [13] did not mentioned about component version control i.e. Effect of Component Version Releases on System and how to overcome issues related to changes in new release and how to integrate new release taking into consideration of risk analysis of component failure in new release, which is very important function of the repository.

A four-stage component-based development process model was proposed by the Lee et al. [14]. Lee et al. [14] Introduced a new technique to integrate off-the-shelf components with the newly developed components in the then existing CBD process models. However, less rather no attention is paid to integrate the internally developed components. The same motivates the author to propose an improved CBD model.

Anurag et al. [15] proposed Umbrella Model for Component-based Software Development in 2011, This model focused more on testing strategies, How to select the best candidate component based on the requirements and design is not mentioned, Repository is not been used in the Umbrella model which is a very important for CBD [24]. Also, there is a lack to management between design, component selection and testing.

Kung-Kiu et al. [16] Proposed the W Model for Component-based Software Development in 2011, this model life cycle is similar to that Of the Y Model [16]

Using off-the-shelf (OTS) components such as COTS or open source software (OSS) promises to reduce development time and cost while increasing software quality, but it also complicates composition and requires a lot of skills for the selection of development model, component selection procedures, component selection timing, effort estimation, OSS and COTS components modification, defect location, and OTS component knowledge management [17].

It is found that the traditional waterfall model and the evolutionary development model are unsuitable for COTS-based development.2These models can be used with some adaptations (such as RCPEP and V-Model XT ) to integrate OTS components .Also, sufficient knowledge of OTS candidates can make using these adapted processes superfluous.

Integrators should follow formal selection procedures deciding the suitable OTS components. A candidate component should fulfill several quality requirements and follows strict industrial standards. Unfortunately, due to lack of evidence on the possible benefits, integrators are reluctant to use a formal process, which is supposed to be complex and time consuming.

Although researchers have empirically evaluated formal processes for selecting OTS components, the cost benefits and preconditions of using a formal process are often unclear and future research can be based on J.Lie et al. [17].

There's no specific development process phase in which integrators select OTS components. But One important strategy proposed to avoid mismatches between the system requirements and the OTS components' functionalities is to identify and evaluate the component as early as possible [17]. However, integrators must consider certain pitfalls if they wish to select OTS components in a project's early phases.

It is advisable to prefer a formal effort estimation tool over personal experience when estimating the effort required to integrate the components as estimations based on personal experience are usually inaccurate. Some estimation tools, such as COCOTS, 8 account for both the components' technical nature and several of the issues we've identified, including component understandability and vendor response time. Investigation shows that estimation tools should also account for possible requirements changes and component evolution, especially for large projects with long durations [17].

Integrators usually use OSS components in the same way as commercial components—that is, without modification, because changing the source code of OSS components might not be feasible, especially for a long-term commercial system with a possibly long evolution path ahead. Thus, developers must consider application contexts, such as commercial versus noncommercial applications and long-term versus short-term applications, when deciding to choose between OSS and COTS components.

Although problems with OTS components are rare, the cost of locating and debugging defects in OTS-based systems is substantial. Investigation shows that the integrators faced tough time in locating defects in 80 percent of the projects that has been investigated [17]. Results show that organizations have partly managed implicit and explicit knowledge about OTS components via component uncles. However, few centralized external channels exist for OTS users to share experiences between organizations.

External experiences of using certain OTS components are scattered in several COTS or OSS portals, bulletin boards, or mailing lists. Search engines usually yield huge, unwieldy sets of results. A centralized experience portal for sharing OTS component-related knowledge between organizations, probably using a global OTS wiki, 12 could be a solution.

In CBSE approach it is often seen that based on the requirements there might be several COTS products that satisfy the requirements with different degrees, but in few cases most likely none of the several candidates would completely match the user requirements, This high chances of occurrences of mismatches between COTS products and system requirements is very communal and to address this issues A. Mohamed et al. [18] developed a decision support approach known as Mismatch-Handling aware COTS Selection (MiHOS).

Linear programming technique is used to identify near optimal solutions. MiHOS suggests substitute strategy for resolving the most suitable mismatches using appropriate actions, such that the most important risk, technical, and resource constraints are met.

For efficient software development, Thomson et al. [19] uses intelligent agents, to examine the design implementation and runtime algorithms are used with specific input and output.

Agents are used to examine algorithms or each other and broadcast the best algorithm among themselves, this was increases accuracy, efficiency and robustness in the system [19].

The key issues in developing any open multi-agent systems are:

(a) How to design a flexible interchangeable system structure that has the ability to reconfigure itself at runtime.

(b) How to provides agents selection mechanism from a vast range of agents each possessing different capability (services)

(c) How to provides coordination or an execution plan model to allow agents working together to achieve a common goal.

Guo et al. [20] developed architecture for Component based software integration that uses multi agent system for integrating the distributed COTS software through a distributed scripting mechanism. This architecture is three

layered architecture and is represented using the agent UML (AUML).

Guo et al. [20] proposed solutions for the following issues in integrating COTS software

a) How to successfully integrate the distributed COTS software [20] Used wrapper to successfully integrate the distributed COTS software application under windows and UNIX platform.

b) How to demonstrate the feasibility of integrating software.

Multi-Agent distributed scripting system (MADSS) [1] successfully demonstration the feasibility of integration of the system using Mobile agents.

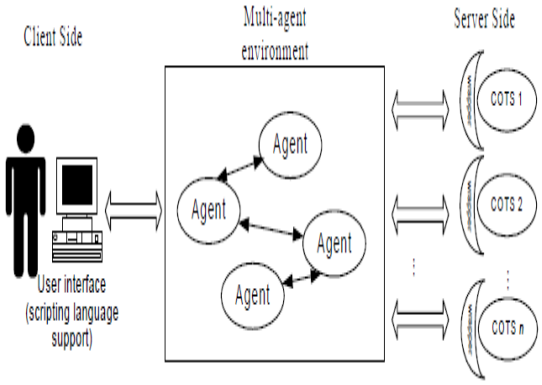


**Fig 4: Conceptual Model of MADSS [20]**

Multi-Agent distributed scripting system (MADSS) is a three tier system in which Agents cooperation is the key for software integration as shown is fig 4. Several distributed wrapped COTS software application exists at the server sides which are wrapped and services are achieved using a service agent.

The role of the Service agent is to maintain the interface of the wrapped COTS software application. Also, when ever news component is added to the server its services and features are also advertised by the service agent to the software agent responsible for elucidation of MDASS agents [20].

While on the other hand to support a user interface to interact with the user to receive jobs written in MADss scripting language client side MADSS uses a client software agent. Client agents do the integration job by generating another mobile slave agents, usually KQML (Knowledge Query and Manipulation Language] is used by agents to communicate with service agent.

# 6. MODELING MULTI-AGENT SYSTEM

Modeling Multi-Agent System from concept to software development brings challenges that have not been studied and traditional ways of analyzing and designing software do not fit the multi agent paradigm, it is quite difficult to consider requirements extract matching with agent properties like autonomy, cooperation, sociability and pro-activeness. Agent dependencies on one another and sociability aspects have to be analyzed in the early stages of the software development process [9].

It is very difficult to select the best modeling method or to evaluate the available modeling methods for MAS project from the several available modeling methodologies applying agent-oriented concepts to software development. MaSE methodology is one such modeling technique for Multi-Agent System.

## 6.1 Multi-Agent System Engineering (MASE) Methodology

The Multi-agent System Engineering (MaSE) methodology helps the designer to set up the initial requirements, analyze models and implement a multi-agent system (MAS), this methodology is independent of any agent's architecture, programming language, or communication framework.

Deloach et al. [11] developed a complete-lifecycle methodology and a complimentary environment for analyzing, designing, and developing heterogeneous multi agent systems. Analysis and design are the two important phases in MaSE.

In Analysis consist of three steps: capturing goals, applying use cases, and refining roles, while the design phase consist of four steps: creating agent classes, constructing conversations, assembling agent classes, and system design as shown in fig 3.

The first step in MaSE analysis is to capture goals and divide them in sub goals or task in hierarchy which generally remain stable throughout the SDLC (software development life Cycle) that state about the system ultimate aim in other words why we are building the system, what system is trying to achieve.

After the goals were defined, use cases are used to identify and represent the functional requirements. Using Use Cases behavior of agents for each situation in MAS is described. Sequences of events of the desired system behavior are represented using Sequence Diagrams.

To support and enforce MaSE, agent Tool system is developed, agent Tool is helpful in implementing all seven steps of MaSE as well as automated support for transforming analysis models into design models [11].

## 6.2 Advantages of Mase

Multi-agent systems has a significant advantage of having capabilities to solve complex distributed problems, earlier by and large these complex problems were normally solved by single applications on single machines, can now be divided and distributed to multiple applications running on multiple machines.

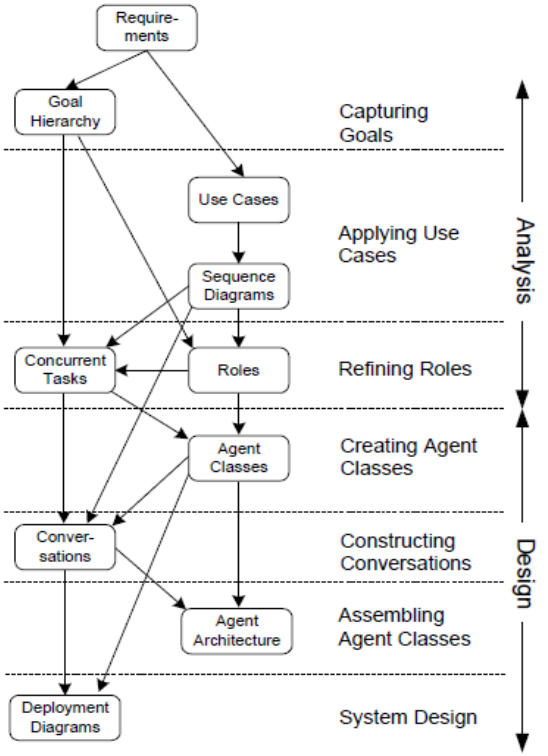


**Fig 3: MaSE Methodology [11]**

Different agents can be used to assigned task in parallel to reach the goal. Another advantage to using multi-agent systems is that it is simple to add a new agent to a multi agent system in order to address new problems without rewriting or redesigning the whole system.

## 6.3 Some General Application Areas of MAS

### 1. Industrial applications

a) manufacturing

b) process control

c) telecommunications

d) transportation systems

### 2. Electronic Commerce

a) Electronic markets / auctions

b) Buying agents (e.g. Jango, shopbot, etc.)

### 3. Business Process Management

### 4. Information Management

a) information gathering

b) information filtering

Present Complex Software Systems require a flexible and powerful framework for developing and managing application dynamically, distributed computing brings challenges of changing requirement at run time, interaction within different networks and different environment. Now a day's researchers focuses their research in the area of software engineering on issues such as automated assembly, adaptively and dynamic reconfiguration[21].

A multi agent based approach which is based on Agent Oriented Software Engineering (AOSE), and component based software engineering (CBSE) address the above issues [21].

# 7. CONCLUSIONS AND FUTURE RESEARCH SCOPES

Our analysis of intelligent component-oriented development strategies shows that majority of the techniques currently in use are hypothetical and speculative without being practically tested. The component selection methods proposed are mostly manual and time consuming which are hard to be automated. As a result, implementing them in large and complex enterprise systems is not practically cost-effective.

Research studies further show that the intelligent techniques fail to adapt to situations where none of the pre-existing components in the repository satisfies a particular business case or when it is more expensive to implement a particular feature using pre-existing components. Our analysis demonstrates that the real enterprise environments encounter multiple such scenarios during its development phases and the issues are mostly left unaddressed by these automatic methods.

Moreover, it was found that most of the techniques do not provide any component integration compliances and regression testing procedures to facilitate the development and integration process. In large enterprise systems, such lack of integration information creates enormous burden on the testing procedures and leads to multiple failures due to missing gaps. Such lacuna in implementation techniques creates more issues when the newly introduced components affect the existing data layer, which other existing components had been utilizing. It is more time consuming and expensive for an external system to determine beforehand whether the "to be introduced" component behavior would affect the functionality of pre-existing components in the target system.

It is therefore concluded that it is practically difficult to map the functional and non-functional attributes of the components with the target system requirements without the presence of facilitating documents.

Considering the above challenges, future component selection frameworks should address the following areas:

• The component selection procedure in the framework should be automatic.

• It should have the functionality to notify the integrator with missing gaps between the requirements and the configurable behavior of the retrieved component.

• It should be able to provide with the integration instructions and ways to configure and control the behavior of the component.

• It should be able to notify on how the retrieved component would be affecting the behavior of the pre-existing components in the system. Further, it should suggest ways to configure the effect on the behavior of other components.

Apart from all these above, the selected component should meet the quality expectations of the target software system.

Our future research will deal with using multi-agent systems to help CBSE in solving some of the unresolved issues such as automation of selection procedure, suggestion of the most suitable component based on the target specification, COTS

component plug and play integration support using scripting tools, automated assembly and dynamic configuration.

## 8. REFERENCES

[1] M. R. Qureshi and S. A. Hussain, 2008. A Reusable Software Component-Based Development Process Model. Ad. Sof. Eng. 39. 2 88-94

[2] H. Hansson, M. Åkerholm, I. Crnkovic, M.Törngren, 2004. SaveCCM – a Component Model for Safety-Critical Real-Time Systems. In: The 30th EUROMICRO Conference (EUROMICRO'04), France.

[3] Research Areas of the Software Engineering Group, Retrieved on November 18th 2011 from http://www.cs.uni- paderborn.de/en/research-group/software-engineering/research/research-areas.html

[4] Basic Concepts of Component-based software, Retrieved on November 20th 2011 from http://www.idt.mdh.se/kurser/cdt501/2008/lectures/book%20Basic%20Concepts%20of %20CBSE.pdf

[5] M.R.V. Chaudron, 2012. Component Models, Technische Universiteit Eindhoven, retrieved on 22nd marches 2012 from http://www.win.tue.nl/~mchaudro/cbse/02_Intro%20CBD_Component%20Models.pdf

[6] M. Madiajagan, B. Vijayakumar, 2006. Interoperability in Component Based Software Development by World Academy of Science, Engineering and Technology

[7] M.L. Griss, G. Pour. 2001. Accelerating development with agent components, In: IEEE Comput. Mag., 34 (5), 37–43

[8] Rafea, A. 2012 Agents and Multi-agent Systems. Retrieved on March 25th 2012 from http://www.cse.aucegypt.edu/~rafea/CSCE485IA/slides/chapter6-mod.pdf

[9] Wooldridge, M. & Jennings, N. 1997. *Intelligent Agents: Theory and Practice*, Retrieved on March 23rd 2012 from http://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html.html.

[10] K. P. Sycara, 1998. Multiagent Systems. In: *AI Magazine* vol. 19(2) 79-92.

[11] Deloach, Scott A., 2001. Analysis and Design using MaSE and agentTool. In: *12th Midwest Artificial Intelligence and Cognitive Science Conference* , Miami University, Oxford, Ohio.

[12] Frederick S., Thomas P. Krishna K., 2004. Multi-Agent System Case Studies in Command and Control, Information Fusion and Data Management, U.S. Government (USG), Informatica 28:2 999–999

[13] Hutchinson, J. Kotonya, G. Sommerville, I. Hall, S. 2004. A Service Model for Component-Based Development. In: Proceding of 30th EUROMICRO Conf., Rennes- France, 162-169

[14] J. Lee, J. Kim, G. Shin. 2003. Facilitating reuse of software components using repository technology, In: Proceeding of 10th Asia–Pacific software engineering conference.

[15] Anurag D. P.C. Saxena . 2011. Umbrella: A New Component-Based Software Development Model, In: International Conference on Computer Engineering and Applications IPCSIT vol.2 IACSIT Press, Singapore.

[16] Kung-Kiu L., F. M. Taweel and C. M. Tran. 2011. The W Model for Component-based Software Development In: Proceeding of 37th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE

[17] J.Li, R.Conradi,O.P.N.Slyngstad,C.Bunse, M.Torchiano, and M.Morisio. 2009. Development with Off-the-Shelf Components: 10 Facts, IEEE Software, 26- 2, 80 - 87

[18] A. Mohamed, G. Ruhe, and A. Eberlein. 2007. Decision Support for Handling Mismatches between COTS Products and System Requirements, ICCBSS'07, Banff, Canada.

[19] Ke. Thompson. 2011. Improving software development and robustness though multi agent systems In: proceeding of the 49th Annual Southeast regional Conference, ACM, New York, USA

[20] Guo-M. F.; Zeng-W. H. ; Jim-M. L. 2005. An architecture for multi-agent COTS software integration systems. In: Parallel and Distributed Systems

[21] M Dragone, David L., Rem C., Gmp O'H. 2009. SoSAA: A Framework for Integrating Components & Agents. In: Proceedings of the ACM symposium on Applied Computing, ACM New York, NY, USA 722–72

[22] Andreas S. J. 2010. Multi-Agent Systems: An Investigation of the Advantages of Making Organizations Explicit. MSc Thesis, Department of Informatics and Mathematical Modeling, Technical University of Denmark.

[23] A. I. Khan and et. al., "A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 2, July 2011, ISSN (Online): 1694-0814, www.IJCSI.org.

[24] A. I. Khan and et. al., "An Improved Model for Component Based Software Development", Software Engineering, Vol. 2 No. 4, 2012, pp. 138-146. doi: 10.5923/j.se.20120204.07