www.irbas.academyirmbr.com May 2017
International Review of Basic and Applied Sciences Vol. 5 Issue.5

Object Oriented Paradigm vs. Agent Oriented Paradigm

NAZIR AHMAD
Assistant Professor; Government Postgraduate College Sahiwal, Pakistan.
Email: nazir313@gmail.com
Tel: 00923046241104

MUHAMMAD TAHIR MUMTAZ
MS Scholar; PMAS- Arid Agriculture University Rawalpindi, Pakistan.
Email: tahirmscs@gmail.com
Tel: 00923467490248

ZIA-UR-REHMAN
MS Scholar; PMAS- Arid Agriculture University Rawalpindi, Pakistan.
Email: zia120rehman@gmail.com
Tel: 00923045434212

GHULAM ALI
University of Central Punjab Lahore, Pakistan.
Email: ghulamali@ucp.edu.pk
Tel: 00923016010309

MUHAMMAD TAYYIAB MUMTAZ
University of Education Lahore, Pakistan.
Email: tayyabjutt128@gmail.com
Tel: 00923087371224

Abstract

In this research we compare the two development techniques, agent oriented development and object oriented. We have discussed the strengths and weaknesses of both techniques which are also shown in a tabular form. Further, we have developed a risk-based framework to facilitate the developer to decide which technique he has to choose. Some critical decision factors have been discussed through polar chart. Object oriented paradigm has great impact on software development through its features like inheritance, data encapsulation, data abstraction and reusability but agent oriented approach has improved development process by its features like autonomy of actions, parallelism, modularity, and adaptability. It has been analyzed that agent oriented development can be given higher level of rank than object oriented approach due to its key features like true parallelism and own thread of control along with autonomy of actions. However, to reduce risk of adapting new paradigm, mixture of both techniques can be used.

Keywords: *Agent Oriented, Object Oriented, Risk-Based, Agent, Object.*

Introduction

Both paradigms have strength and weaknesses.

Strengths of Object Oriented Paradigm

Object-oriented development makes the promise of reusability, modular architecture, increased quality, improved reliability and flexibility, real world modeling http://www.dotnetperls.com/oop. Following are the strengths of object oriented development.

Inheritance

Inheritance is one of the primary strengths of object oriented paradigm. Inheritance means the data members and methods defined in a parent class can be inherited into subclass. The use of inheritance increases the reusability, because additional functionality can be added to a class without altering the existing class.

Encapsulation and data abstraction

Encapsulation is another primary strength of object oriented paradigm. Encapsulation is a data hiding technique, which enable the designer to syntactically separate the interface and implementation. Whereas data abstraction is a technique of representing the essential properties without including the semantic details. Due to data abstraction data is not visible to the outside world. Only the methods defined in that class can access these data members.

Reusability

The use of OO approach increases reusability. We can add new code at the top of existing code without changing the existing. The code reusability heavily depends on the structure of the underlying system http://www.dotnetperls.com/oop. The existing software modules can also be reused to develop new application. But the reusability of existing modules is not easy, it requires planning and investment http://www.ajsquare.com/Publications/object-oriented-development-ood.php.

Modular Architecture

An OO system is composed of different modules like objects, subsystems, libraries and frameworks. This modular architecture facilitates the developers to modify the existing system without interfering the usurers of the systems and other components of the system http://www.ajsquare.com/Publications/object-oriented-development-ood.php. Further, new modules can easily be plugged- in to the existing system.

Increased Quality

Increased quality is largely related to the reusability. If ninety percent components of the new system are being reused, it will automatically increase the quality of the system. Because these components are already tested and verified, it will also reduce the testing effort. Then only the remaining part have to be tested and verified http://www.ajsquare.com/Publications/object-oriented-development-ood.php.

Improved Reliability and Flexibility

The Object-oriented systems are more reliable, because new functionality can be derived from existing objects. The objects are dynamically crated and invoked and new objects can be created at any time. These newly created objects inherits the data members from many other objects, also inherits the behaviors of the parent classes. So, new behavior can be added to the system without affecting the existing system's functionality http://www.dba-oracle.com/t_object_oriented_approach.htm.

Weaknesses of Object Oriented Paradigm

Complex Structure

The compilers available for the object oriented languages do not produce as much efficient machine readable code as compare to the compilers available for structured languages. This is due the complex structure of the object oriented programming languages which also makes the execution of object oriented programs slower.

Dynamic Memory Allocation and De-allocation

Dynamic memory allocation and de-allocation is one of the major weaknesses of object oriented programming languages. More than half million allocations are needed to for a simple Othelo game applet. This makes it less efficient as compare to traditional languages.

Centralized Structure

Commonly objects are considered as centrally organized entities, because the methods of an object are invoked by the other components of the system. There are some situations, which required self organized entities.

Multiple and dynamic classification

In OO languages, once an object is created, it can never change its class or it becomes an instance of multiple classes. So, it can never change its behavior. But in certain situations we need more flexible approach.

Strengths of Agent Oriented Paradigm

AO paradigm includes all the capabilities of other paradigms such as object oriented, knowledge engineering and service-oriented) and even more abilities.

Autonomous

The primary strength of agent oriented paradigm is autonomy of actions. An agent is totally independent entity and has control over its internal state and behavior (Castelfranchi 1995). The invocation of actions is internal to agent, so they can take initiatives and respond to a massage in any way yes or no. They can watch their internal set of capabilities. In other words agent can decide when and how to act when a request came from another agent. Agent can also dynamically change its behavior according the change in the environment. So, agents can perform their actions without external intervention.

Better Maintenance

The use of agent oriented paradigm provides the ease of system maintenance. Because new agents can be added to the existing system without any modification and existing agents can be modified without interfering with other components of the system. It will also reduce the overall maintenance cost of the system.

Analogies from nature

Agents can demonstrate the properties of real world characters. Agent oriented systems tend to model the real world in a more complete fashion than object oriented approach. Agents are organized as societies;

these societies can demonstrate the political and organizational behaviors. In other words nature provides a rich trove of ideas for multiagent system development.

Modularity

The modular nature of agent architecture increases efficiency of task execution, increase code reusability, and provides better separation of business logic and infrastructure, faster development, reduces communication overhead and provides high degree of flexibility.

Parallelism

The agent technology provides true parallelism. Each agent has its own thread of control along with autonomy of actions. The communication among agents is asynchronous which increase the parallel processing.

Adaptability

Agents have the ability to adapt the dynamic changes in their environment. According to the changes in environment agents can modify their behavior. An agent can drop its goals and can adapt new goals dynamically. The agent oriented system may introduce new agents which are previously unknown to the system or can modify the existing agents.

Social Ability

The agents of the system cooperate with other agents of the system to achieve common goals. A computation consists of these agents informing, requesting, offering, accepting, rejecting, competing, and assisting one another.

Weaknesses of Agent Oriented Paradigm

Complex Architecture

Due to dynamic behavior of agent oriented applications the agent architecture is more complex than object architecture. A large number of agent architectures have been developed; due to a vast variety of agent architectures it is very difficult for developers to select a single best general architecture for the development of agent oriented applications.

High Development Cost

The development of agent oriented systems costs higher as compare to traditional paradigms, due to the lack supporting tools.

Large Scale Applications

The existing agent oriented programming languages, design and development tools are not suitable for the development of large scale applications.

Complex Interactions

The timing, sequence and number of agent interactions is unpredictable. First, the agent decides at run time that which of its goals requires interactions in a certain situation, and what kind of interaction requires to

achieve that goal, and when these interactions will occur. Secondly, if one agent sends a request to other agent, it is very difficult to predict that his request will be accepted, rejected of modified through some social interchange.

Table 1: Strengths of OOP & AOP

Features	Object Oriented Paradigm	Agent Oriented Paradigm
Inheritance	Yes	No
Encapsulation and data abstraction	Yes	No
Reusability	Yes	No
Modular Architecture	Yes	No
Increased Quality	Yes	No
Improved Reliability and Flexibility	Yes	No
Autonomous	No	Yes
Better Maintenance	No	Yes
Analogies from nature	No	Yes
Modularity	No	Yes
Parallelism	No	Yes
Adaptability	No	Yes
Social Ability	No	Yes

Table 2: Weaknesses of OOP & AOP

Features	Object Oriented Paradigm	Agent Oriented Paradigm
Complex Structure	Yes	No
Dynamic Memory Allocation and De allocation	Yes	No
Centralized Structure	Yes	No
Multiple and dynamic classification	Yes	No
Complex Architecture	No	Yes
High Development Cost	No	Yes
Large Scale Applications	No	Yes
Complex Interactions	No	Yes

Home Grounds

Agent Oriented

Following is a list of interesting issues to be considered when using agent oriented approach to develop a system (Message 2000).

Diverse Types of Communication

Must use the agent oriented approach when the system requires diverse types of communication.

Emergent Behavior

Agent oriented approach is more useful in situation when the system has to perform better where the behavior of the system cannot be specified on a case by case basis.

Social cooperation, negotiation and competition

Agent oriented approach is more suitable in situation where a number of different entities involves in social cooperation, negotiation and competition.

Autonomous actions

Use the agent oriented approach when the system has to perform autonomous actions.

Maintenance

Agent oriented approach is more suitable in situations where the system will be expanded, modified and the system requirements and behavior may change over time.

Parallelism

Must use the agent oriented approach where application requires parallel processing and asynchronous communication.

Complex Distributed Application

Agent oriented approach is more suitable for the development of complex distributed applications.

Object Oriented

Development Cost

Developing object oriented system involves lesser cost than using agent oriented paradigm, due to the support of well-defined methods and tools.

Hardware and software resources

In situations when hardware and software resources are limited object oriented approach is more suitable.

Analysis and Design Skills

Use of object oriented approach requires less analysis and design skills as compare to agent oriented approach.

Faster Development

Object oriented paradigm have vast variety of IDEs and CASE tools (e. g. Visual Studio), which provides ease of development, the application take lesser time to be implemented.

Cost of Acquiring the Methodology

It is the cost related to adopting the development approach. It also includes the cost incurred for the training of the development staff. Object oriented approach costs lesser on the training of development staff; because OO is evolved over the decades, and most probably the engineers already have the basic knowledge of object oriented approach.

Centralized

Object oriented approach is more suitable in situations where system requires centralized organizational structure.

Passive Entities

Object oriented approach is more suitable in situations where system is composed of passive and dependent entities, the actions of entities are invoked externally and the object must have to respond the calling entity.

Critical Decision Factors

The selection of a suitable approach for the development of software is a challenging task. Below is a set of critical decision factors that must be considered while selecting an appropriate approach for the development of any type of software.

1. Degree of Interactions
2. Maintenance Effort
3. Development Cost
4. Analysis and Design Skills
5. Degree of Parallelism

Comparison and Contrast

The comparison and contrast of the above defined critical decision factors is presented in Figure1.

Degree of Interactions

The scale of degree of interactions is defined over following four levels of interactions.

Interactions between objects (OI)

The interaction between object have very low degree of interaction as compare to agents.

No direct interactions between agents (NI)

The degree of interactions between those agents that do not interact with each other directly is higher than the degree of interaction between objects. They just perform their responsibilities without directly interacting to their neighbors. But they can learn from their neighbors without direct interactions with each

other. For example ant colony agents, the objective of ant colony agents is to search for food and carry it to nest. When an agent fined a food location it sends a message on a shared communication media, that it has found a new location where food is available. The other agents update their food locations mental state according to this information. So the agents learned from the other agents without direct interactions.

Simple interactions between agents (SI)

The degree of simple interactions between agents that directly interacts with their neighbors is higher than the degree of NI. These agents negotiate with their neighbors and other agents of the society. The agents discuss and share their knowledge and experiences. They enhance their capabilities by negotiating with other agents of the system. They compete with the other agents of the society.

Complex, or conditional, or collective interaction between agents (CI)

The degree of complex interaction between agents is higher than all the above mentioned types of interactions. The complex and conditional interactions required between those agent which have emergent behaviors, they have social norms such as resolving conflicts, trust and self-regulation. These agents may involve in many types of interactions simultaneously.

Maintenance

Maintenance is defined as the percentage of application modified due to no-corrective maintenance. The agent oriented systems are easy to maintain as compare to object oriented system. Here we define the maintenance as the percentage of application modified due to no-corrective maintenance. It means the modification of existing agents and addition of new agent costs lesser than objects.

Development Cost

Development cost is the percentage of development effort required for debugging. The development cost of agent oriented system is higher than object oriented system. Debugging agent oriented system takes most of the part of development efforts which make the development of agent oriented system more costly and difficult. This is due to the lack of support of debugging tools in IDEs for agents. The percentage of development effort required for debugging is the ratio of debugging effort by total development effort.

Analysis and Design Skills

Object oriented analysis and design skills

The development of object oriented system requires only object oriented analysis and design skills.

Object oriented plus structured analysis and design skills

At the second level of analysis and design skills the developer has the knowledge of both paradigms structured and object oriented.

Agent oriented analysis and design skills

To develop an agent oriented system the developer must have the skills of agent oriented analysis and design along with structured and object oriented.

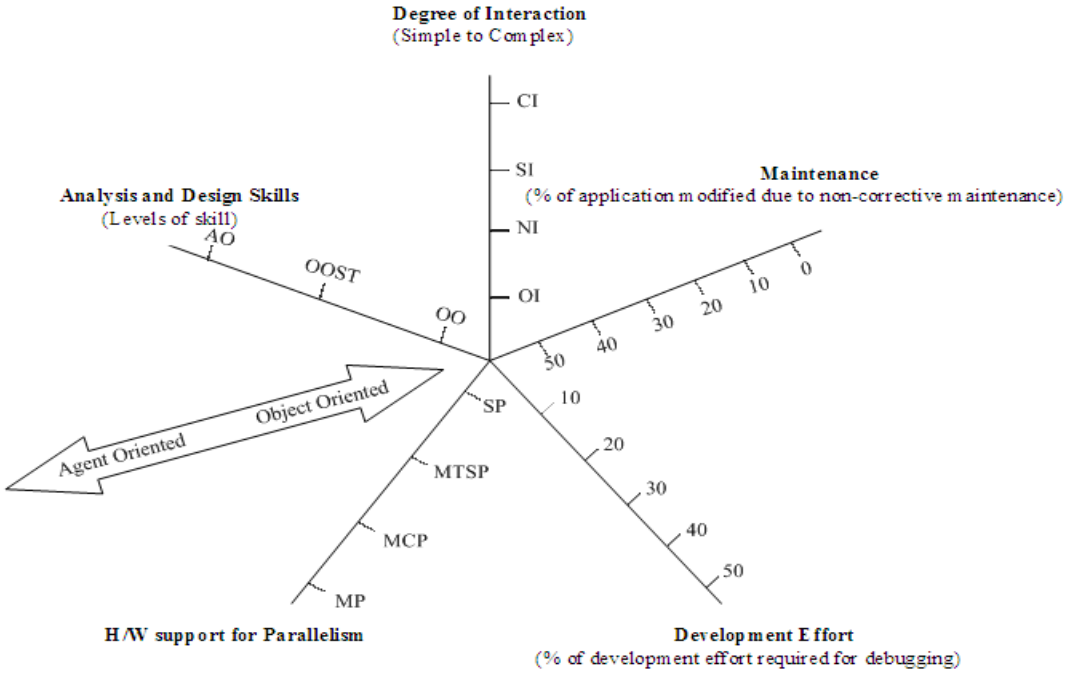

Figure 1. Home ground polar chart to compare the agent oriented and object oriented development technique.

Hardware support for Parallelism

No parallelism

The application requires no parallel processing. Object oriented technique is better in such situations. Because no need to make extra investment in purchasing hardware.

Multithreaded single processor

This scenario supports parallelism but at a low level.

Multi core processor

The multi core processors support the parallelism at higher level as compare to previous two architectures.

Multi processors

To achieve the true parallelism we need multi processor support to run the multiple agent at different location which negotiate with each other asynchronously.

Risk-Based Framework

The proposed risk-based framework consists of risk analysis, risk comparison, architecture analysis, tailor development technique and execute and monitor steps. The flow of these steps is depicted in "Figure 2". In risk analysis step the rating of the software's environmental object oriented and agent oriented risks is computed, if the risk ratings are uncertain then collect the data and perform analysis. Repeat this process until we are not certain about the risk ratings of the system. In step 2 the risk ratings are compared, if risks

of either approach dominate then proceed with other risk-based approach. If neither approach dominates then go to step 3.

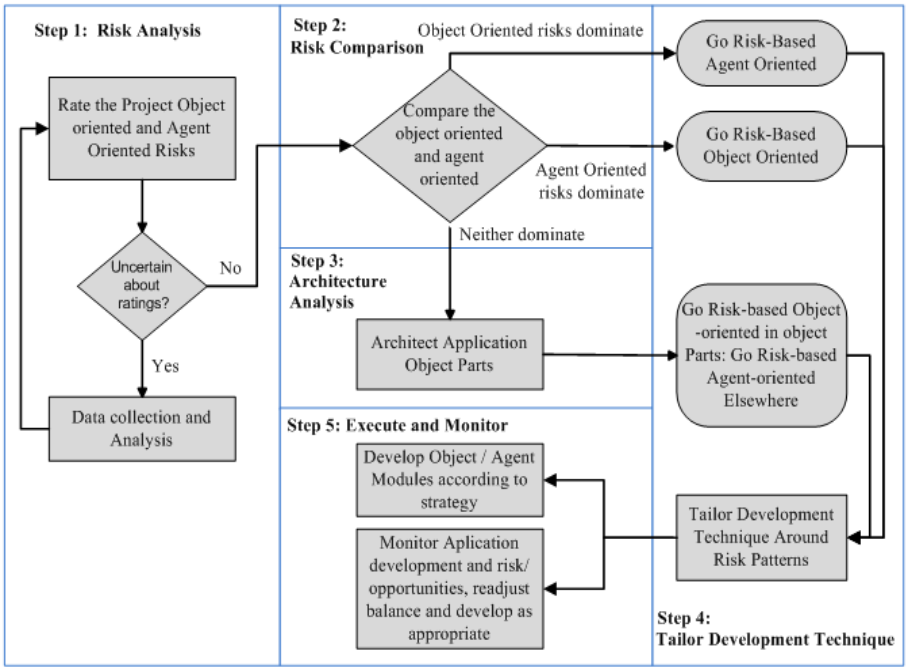

Figure 2. Risk-Based Framework

The architecture analysis can be done by using the "Quality Attribute Workshop (QAW)" and "Architecture Tradeoff Analysis Method (ATAM)". The QAW method can be used to develop a logical architecture of the system regarding the critical decision factors defined in section 3. Then the developed architecture can be verified that how well it fulfill our requirements, and how well it handle the involved risks and tradeoffs in requirements by using the ATAM method.

Tailor development methodology, when neither risk-based technique dominates, otherwise use the dominated technique. By tailoring the development technique we can get the advantages of both the techniques. Build the self-organizing, parallel and emergent behavior parts as agents and develop the ramming part as object oriented. The combination of both the techniques provides flexibility, preserves the rigidity and inefficiencies of other techniques and reduces the risk of adapting new paradigm.

The execution and monitoring step is defined to develop the system according to the defined criteria and monitor the development, readjust the development plan and balance the development of combined technique according to the system requirements.

Conclusion

This paper discusses the comparison between object oriented and agent oriented software development techniques including their strengths and weaknesses. Agent oriented development can be given higher level of rank than object oriented approach due to its key features like true parallelism and own thread of control along with autonomy of actions. However, mixture of both techniques can be used to reduce risk of adapting new paradigm.

References

Agents and objects are compared.
Agent-Oriented Software Development.
Agent Oriented Software Engineering and Other Programming Paradigms.
Agent Oriented Software Engineering and Other Programming Paradigms.
An Efficient Memory System for Java.
Analysis of Factors Affecting Testing in Object oriented systems.
A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance.
C. Castelfranchi. A point missed in multi-agent, DAI and HCI. In M. J. Wooldridge and N. R. Jennings, editors, Intelligent Agents - Theories, Architectures, and Languages, number 890 in LNAI, pages 49–62. Springer, Verlag, 1995.
Determining When to Use an Agent-Oriented Software Engineering Paradigm
Efficient implementation of neural networks in an object oriented programming environment
Firth, R., et al.: A Classification Scheme for Software Development Methods. Software Engineering Institute Technical Report 87-TR-41. Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA, 1987. Robert Firth, William G. Wood, Richard D. Pethia, Lauren Roberts, Vicky Mosley
http://www.dotnetperls.com/oop
http://www.dba-oracle.com/t_object_oriented_approach.htm
http://www.ajsquare.com/Publications/object-oriented-development-ood.php [Adhikari, 1995; Taylor, 1995]:
http://www.hit.ac.il/staff/leonidm/information-systems/ch06.html
Message: Methodology for Engineering Systems of Software Agents – Initial Methodology. Eurescom Participants in Project P907-GI 2000.
Objects and Agents compared.
Programming Multi-Agent Systems.
SEI Architecture Analysis Techniques and When to Use Them.
Types and priorities of multi-agent system interactions.