RESEARCH ARTICLE OPEN ACCESS

The Common Interface Oriented Architectural Framework to Improve Compatibility of the Pure Object Oriented Cross Languages Interoperability

Sonar Sanjay Bhagwan
Ph.D. Scholar
[MCA, M.PHIL-CS]
Rai University, Saroda
Dholka Taluk, Ahmedabad - 382260
Gujarat - India

ABSTRACT:
Traditional object-oriented design methods only deal with the design of specific applications and do not facilitate the design of common framework. A framework provides a highly effective mechanism for software reuse within an application domain; the framework captures the features that are common across the product line. The Pure Cross-Platform Object Oriented Languages are purely atom/Variable/object dependable. As well as many aspects and requirements are based on the specific variable or Object(s) or Atom. In these languages, the object is atom as well as Variable (Tangible Unit) abstracted from the problem also reusability, genericity, polymorphism and inheritances are specialized and generalized from the objects. The basic problem is that, No any Common compatible and interoperable architectural framework and interface oriented object oriented system design for software implementations on the pure cross-languages platforms, also No any specification to identification of the object limitation, across the system on the based on the object paradigm. Also present tools and methods/tasks of the object oriented system are not incorporated tools as well as not specifically step next tools, to proceeds the system from identification of need to system design for the Cross-Platform software development in both JVM and CLR engines. The research prove the Common Compatible and Interoperable Architectural Framework to develop system to any pure cross languages platform as Java Virtual Machine technological languages like Scala, Ruby, Java and Microsoft Visual Studio languages like vb.net, c#.net, j#.net, asp.net and .cobol.net. The Architectural Framework provides the common platform to design the system in any pure cross languages compatibility. The Architectural Framework precedes the system from Object elicitation to clean room software system for the pure cross languages development.

Keywords:- domain, Atom, Common object identification, genericity, Pure Object Orientation, Common compatible architectural Framework, elicitation, clean room system , Interoperability, CLR, JVM, Scala, Groovy, Dot Net, abstraction, object paradigm, automation

I. INTRODUCTION

An object-oriented framework is a set of abstract classes that together comprise a generic solution to a family of related problems drawn from a specific domain. Reusable frameworks typically emerge as the result of an iterative, evolutionary process during which they successively address a range of different requirements. The present and future systems are being developed by the Pure Cross-Languages Platform like in Java Virtual Machine and Common Language Runtime in Microsoft Visual Studio framework. The pure Cross-Platform independent Object Oriented Languages are purely Atom/Variable/Object dependable. As well as many aspects and requirements are based on the specific variable or Object(s) or Atom. In these languages, the object is atom as well as Variable (Tangible Unit) abstracted from the business logic **or** customer requirements and System Requirement Specification (SRS); also reusability, genericity, polymorphism and inheritances are specialized and generalized from of the objects

The first basic problem is that, No any specific compatible and interoperability interface oriented system design or common architectural framework for the any pure cross-languages platforms software implementations, also No any

specification to identify and precede the **object limit**, across the system on the based on the **object paradigm**. Also present tools and methods/tasks of the object oriented system are not incorporated tools as well as not specifically **steps next tools** to precede the design from identification of needs to structural algorithms for the designing of the Cross-Platform software development in either JVM or CLR engines. Basic drawbacks in present object oriented systems for developing system in pure cross languages platforms; the problem pointed out by the following tasks.

1. Clarity of object/domain identification from the requirements elicitation of the system.
2. Specification of object limit in the domain.
3. Step next tools to precede design
4. Compatibility and interoperatibility in the objects and domain.
5. Integrations and interface between objects.
6. Association rules and fewer anomalies in links of objects.
7. Clustering in the objects.

The Research proceed an **automation**, that the design a common compatible and interoperable Architectural Framework, this common Framework applied on any pure cross languages platform to Java Virtual Machine technological languages like Scala, Groovy, Java and Microsoft Visual Studio languages like VB.NET, C#.NET, J#.NET, VC++.NET and ASP.NET to develop the applications or products. The common automation architectural framework provides the common tools to design the systems/applications/products in any pure cross platform independent languages compatibility by the identification, specification, abstraction, limitation, integration, association, cauterization, and elicitation of the objects. Framework automates the system requirements elicitation to clean room software system.

Common Architectural Framework covers the following cross languages, for designing the applications.

1. JVM Platform (Java based System)
   Combined languages like, Java**,** Scala and Ruby.
2. CLR Platform (VS.NET based System)
   Integrated Languages: **VB.NET, C#.NET, J#.NET and ASP.NET**

II. OBJECTIVES

During the past years, the need for software reuse and commonness has become evident. Object-orientation has provided a means to increase the reusability of code, by introducing standard interfaces and inheritance. Class libraries have provided well defined and tested reusable components, but using class libraries mainly implies reuse of code and little reuse of analysis and design.

To increase the potential of reuse and commonness in the cross platform languages, the **common automated interface oriented architectural object-oriented frameworks** have been suggested. This framework is intended to capture the functionality common to several similar applications of cross platforms. The following objectives are summarized for the common automated interface oriented architectural object-oriented framework.

1. Design the common automated interface oriented architectural object-oriented frameworks.
2. Common compatible and interoperable architectural framework to directly support for cross platform independent languages.
3. Interface Oriented (automation of tools) architectural framework.
4. Step-to-Next architectural framework.
5. Domain Specification, for identification of common objects.
6. Identifications of generic domains.
7. Domain is based on the atom, variable, entities, tangible unit(s) and enti-atom.
8. Common Framework support for both JVM and CLR engine/compiler.

9. Genericity in all domain objects for commonness.

III. TARGET GROUPS

The framework is supported to the following groups

1. Software organizations in need for reuse technique.
2. Developers/analysts to design pure object oriented system.
3. Students/ Scholars interested on pure object oriented techniques.

BACKGROUND OF PROPOSED RESEARCH WORK

The Basic research is based on the system development by the present pure Cross Platform independent Object Oriented Languages. Many software firms and groups are developing the present software's by the cross platform independent languages on different platform.

But significant drawbacks are, No any common object oriented architectural framework as automation to identify and proceeds steps next methods to implementation application in the cross platform languages. The research prove that the common compatible and interoperable architectural framework is based on the automation tool, applied on any pure cross platform independent languages in both JVM and CLR engines.

The requirement is common framework for developing the system in the pure cross platform languages as java virtual machine platform and Microsoft Visual studio platform. As same, I want to design the common architectural integrated automation models also called specific compatible and interoperatibility interface oriented system design, to design and implementation of software implementations in both JVM and CLR engines.

IV. PREVIOUS RESEARCH

PREVIOUS RESEARCH ON OBJECT ORIENTED COMMON ARCHITECTUREAL FRAMEWORK (PAST STATUS)

Present work from **Alan Snyder** at **Hewlett-Packard** on developing a common framework for

object-oriented terminology. They defined several comparing criteria and performed an extensive comparison of these methodologies. The results were presented in a set of tables. The goal of this effort is to develop and communicate a corporate-wide common language for specifying and communicating about objects. We next look into the research activity at Hewlett-Packard, led by Dennis de Champeaux.

De Champeaux and Faure at Hewlett Packard Laboratories have initiated a systematic comparison of OOADMs, by surveying more than ten object-oriented analysis methodologies; de Champeaux compared the common features and the major differences of the chosen methodologies. His article provides an excellent tutorial for object-oriented analysis, but his comparison of the methodologies is somewhat abbreviated.

De Champeaux is developing a model for object-based analysis. His current research focuses on the use of a trigger-based model for inter-object communications and development of a top-down approach to analysis using ensembles. We then survey two research activities that prescribe the design process.

The above research has influenced our decision on the selection of OOADMs to use in our comparisons and the criteria we should use to compare these methodologies. However, we take a different research path by building a formal representation for each methodology and comparing these methodologies based on the uniform representation. In contrast, no uniform and formal basis was developed in previously published research. The comparisons were based on the informal description of the methodologies.

The technique we use to construct the formal representation of an OOADM is meta-modeling.

Meta modeling has been used to develop theoretical foundations for information systems. Brinkkemper has adopted this technique to build a framework for the formalization of information system modeling the analysis and design of information systems can be viewed as the conceptual modeling of the systems, and the development of an information system is the modeling process that starts with the Universe of Discourse and ends at the information system. Hence, various analysis and design methodologies can be viewed as specific modeling processes that apply a set of predefined techniques. He proposes a formal framework for information system modeling and addresses the issues of what is the best way to construct a model and how one particular model is related to the other models of an information system. Due to the lack of a common core of theory in information systems.

Rebecca Wirfs-Brock from **Tektronix** has been developing an object-oriented design method that focuses on object responsibilities and collaborations. The method includes graphical tools for improving encapsulation and understanding patterns of object communication. **Trygve Reenskaug** at the Center for Industriforskning in **Oslo**, Norway has been developing an object-oriented design method that focuses on roles, synthesis, and structuring. The method, called Object-Oriented Role Analysis, Syntheses and Structuring, is based on first modeling small sub-problems, and then combining small models into larger ones in a controlled manner using both inheritance (synthesis) and run-time binding (structuring).

Then present investigations by **Ralph Johnson** at the University of Illinois at Urbana-Champaign into object-oriented frameworks and the reuse of large-scale designs. A framework is a high-level design or application architecture and consists of a suite of classes that are specifically designed to be refined and used as a group. Past work has focused on describing frameworks and how they are developed. Current work includes the design of tools to make it easier to design frameworks.

Object Oriented Design with Applications by Booch

Brooch's method is mainly intended for the design stage of a project. Booch describes a number of general properties of well-structured complex systems. Systems built with the OODA methodology should satisfy these properties. In OODA the problem domain is modeled from two different perspectives: the logical structure of the system and the physical structure of the system. For each perspective both static and dynamic semantics are modeled. OODA describes Appeared in the Proceedings of the 26th Hawaii International Conference on System Sciences, January 1993 Volume IV, pp. 689-698 various techniques to accomplish these tasks, and provides a rich set of graphical notations.

Designing Object Orientation by Wirfs Brock.

This methodology covers mainly the analysis phase of the systems development life cycle. Two major concepts, abstraction and encapsulation, are used to manage the real world complexity. The DOOS methodology describes the problem domain as a set of collaborating objects. A system is developed in two stages. During the initial exploratory phase objects, their responsibilities and the necessary collaborations to fulfill these responsibilities are identified. The detailed analysis phase streamlines the results of the first phase. Two graphical techniques are introduced for the second phase. One technique is to show classes and class structures and the other is to depict classes, subsystems and client-server relationships.

Finally, we present some results from the research group in object-oriented software engineering at **Northeastern University**, led by **Karl Lieberherr**. They have been working on object-oriented Computer Assisted Software

Engineering (CASE) technology, called the **Demeterm system**, which generates language-specific class definitions from language-independent class dictionaries. The Demeter system includes tools for checking design rules and for implementing a design.

V. PRESENT RESEARCH ON OBJECT ORIENTED COMMON ARCHITECTUREAL FRAMEWORK (PRESENT STATUS)

A unified OO modeling method emerged in the mid-1990s. It was based on the methods and work of three authors (with their coauthors): OO Analysis and Design by Grady Booch [Booch, 1994], Object Modeling Technique (OMT) by James Rumbaugh and his colleagues [Rumbaugh, 1991], and OO Software Engineering (OOSE) by Ivar Jacobson and his colleagues [Jacobson, 1992]. The ‘‘three amigos’’ (as the three distinguished authors became known) gathered in Rational Software Corporation and started working on the Unified Method. However, they quickly realized that a method should consist of a modeling language, which should be standardized as a common means of communication, and a process of developing software, which should not be homogenized, but should be customizable for particular problem domains, applications, organizations, development teams, and so on. This is why they decided to work only on the language itself, which they named the Unified Modeling Language (UML).

In the present, following **methods** and **tools** have been used, for the developing object oriented applications, But No any domain specifications and interface oriented architectural framework to develop applications in any Pure Cross-Languages Object Oriented Platform.

1. Object Oriented Paradigm.
2. Object modeling Techniques (OMT)
3. Unified Modeling Language (UML)
4. Object orientation using UML
5. System Engineering by Dr. Pressman
6. Component based system using divide and conquer Methods.
7. Object association by Cohesion and coupling
8. Software Matrix.
9. Agile system in Object Oriented System.
10. Requirements Elicitations.
11. Object Constrain Language (OCL).
12. System development Paradigms & Rationale Management.

VI. MATERIAL AND METHOD

To achieve these, start study from the requirements **Object domain(s)** from the business logic, customer requirements elicitations as well as identification of need and System Requirement Specification, and after these, Design the integrated interface tools and automation tool(s) using following tasks and methods for integrate the **common compatible and interoperable architectural framework** to develop the system in cross platforms languages.

VII. EXPERIMENTAL STUDY

- **Research method with justification:**

Developing frameworks introduces new aspects when dividing the work. The main idea of a framework is to capture generalities of a domain or a set of applications within a domain. Finding generalities requires a good overview of the domain and the system respectively. In domain Analysis, the following two types of domain objects are to be identified / specified from the problem.

- System Model for Solution Domain
    1. Base Objects
    2. System Objects

- Application Model for Problem Domain
    1. Problem Domain Objects:
    2. User Interface Objects:
    3. Control Objects:

The research prove automation, that the design a common compatible and interoperable

Architectural Framework, this common Framework applied on any pure cross languages platform as Java Virtual Machine technological languages like Scala, Ruby, Java and Microsoft Visual Studio languages like VB.NET, C#.NET, J#.NET. And ASP.NET, to developing the applications. The architectural framework provides the

Common platform to design the system/application in any cross languages compatibility from abstraction and elicitation of the objects. The framework precedes the system from Object elicitation to clean room software system

- **Object/Variable/class/instances can be Generic:**

A service can have different implementations (different code) for different objects, which can produce observably different behavior (although sharing some common intent). A client can uniformly issue requests for the service (the requests identify a common operation); an appropriate implementation is selected for each request. There is no limit to the number of different implementations of a given service.

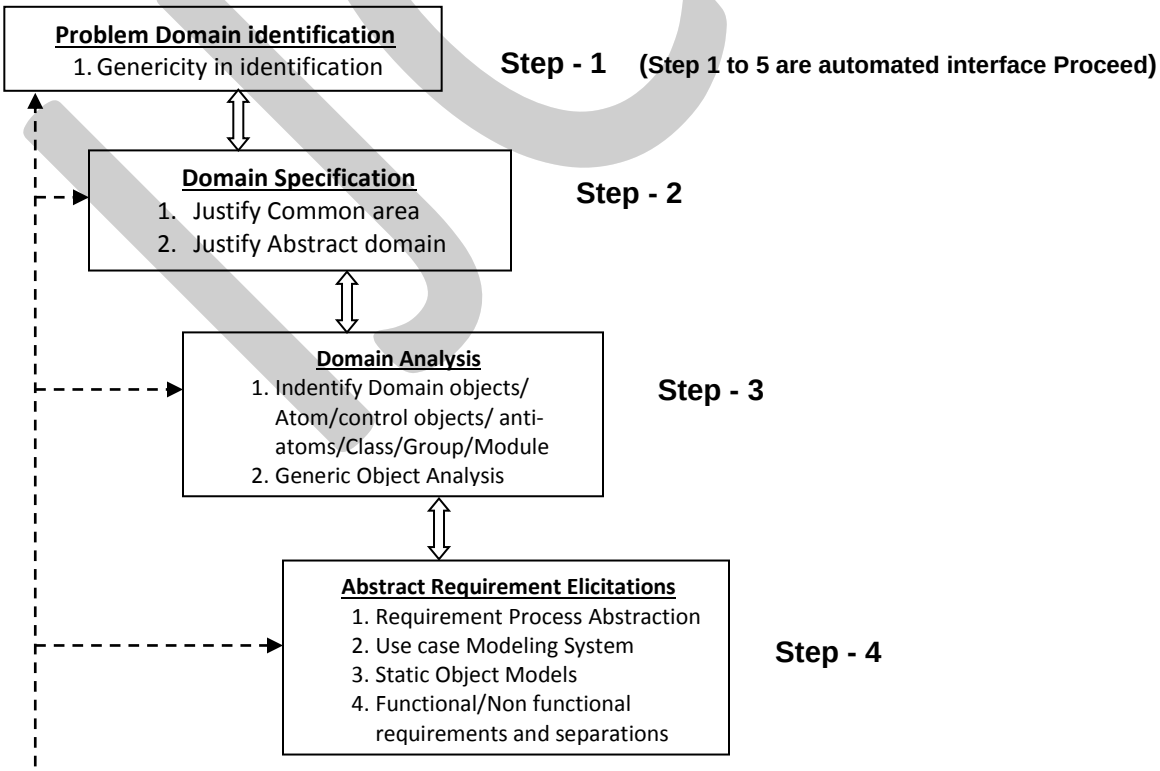
An operation with multiple implementations is a generic operation. Clients that request generic operations may themselves be generic in the sense that they can perform a common activity on different kinds of objects.

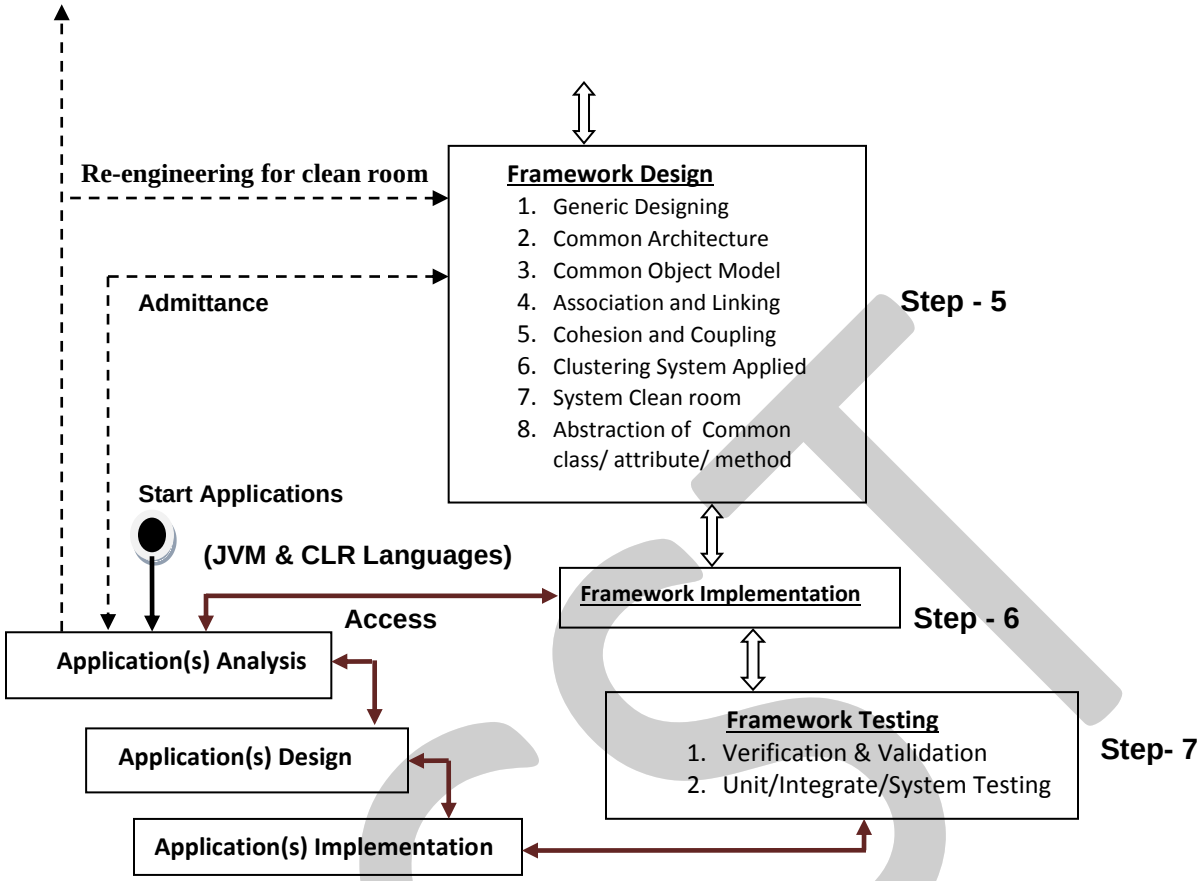
The selection of code to perform a service (binding) is based on the objects identified in the request. In general, the identification of the objects occurs when the request is actually issued, so the selection of code would happen at that time (dynamic binding). Code selection is sometimes based on factors that are known prior to execution, so that the code can be selected during program compilation or linking (static binding).

The importance of the common Object Oriented Architectural Framework is the **automation** system (interface oriented) to develop the system in any cross platform languages in both JVM and CLR engines.

VIII. RESULT AND DISCUSSION

Common Automated Interface oriented architecture framework for developing pure cross platform languages as JVM and CLR engines.





XI. CONCLUSION

The research prove the **Common Compatible and Interoperable Architectural Framework** to develop system to any pure cross languages platform as Java Virtual Machine technological languages like Scala, Ruby, Java and Microsoft Visual Studio languages like vb.net, c#.net, j#.net, asp.net and .cobol.net. The Architectural Framework provides the common platform to design the system in any pure cross languages compatibility. The Architectural Framework precedes the system from Object elicitation to clean room software system for any pure object oriented application development.

ACKNOWLEDGEMENT

I would like to thanks Dr. Samrat O Khanna, Head of the Departments of ISTAR institute of the Anand-Gujarat, I also Thanks Dr. N N Jani, Director of the S.K Patel Institutes of Gandhinagar-Gujarat to inspired me Encouraged me to perform at my best. And I also Thanks to all Software firms who gave me the best support to abstract my goal of research to reach my destination

REFERENCES

[1]. De Champeaux, D. and Faure, P., "*A Comparative Study of Object Oriented Analysis Methods*," Journal of Object-Oriented Programming (JOOP), March/April, 1992, pp. 21-33.

[2]. Grady Booch. *Designing an Application Framework*. Dr. Dobb's Journal 19, No. 2, 1994.

[3]. Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissades, John. *Design Patterns*.

[4]. Ralph E. Johnson. *How to develop frameworks*. Notes for OOPSLA '95, 1995.

[5]. James J, Odell, publication year 1998, *Advanced Object Oriented Analysis and Design by UML*, ISBN- 9780521648196

[6]. Bernd Bruegge, Allen H. Dutoit. Object *Oriented Software Engineering, Using UML, Patterns, and Java*, Second Edition, Pearson Education

[7]. Dr. Roger S, Pressman, *Software Engineering*, A practitioner's Approach, Fifth Edition, McGraw Hill international edition, Computer science series.

[8]. M. Mattsson and J. Bosch. *Object oriented frameworks*: Composition problems, causes and, solutions. In Building Application Frameworks: *Object-Oriented Foundations of Framework Design,* pp. 467-487, M. Fayad, D. Schmidt, R. Johnson editors, Wiley Press, 2000.

[9]. Ian Sommerville, *Software Engineering*, 6th Edition, Pearson Education

[10]. Matteo Golfarelli, Stefano Rizzi, *Data Warehouse Design, Modern Principles and Methodologies*, Tata McGraw-Hill edition

[11]. James Rumbhaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, *Object Oriented Modeling and Design*, Prentice- Hall India Edition

[12]. Heninger K.L., *Specifying software requirements for complex systems*. New techniques and their applications. IEEE Transactions on Software Engineering 6 (1), p. 2-13, 1980.

[13]. Johan Larsson. *Object oriented frameworks*.REBOOTConsortium, 1992.

[14]. N. Bouassida, H. Ben-Abdallah, and F. Gargouri, A. Ben-Hamadou: *A stepwise Framework Design Process*, IEEE International Conference on Systems Man and Cybernetics, 07-09 October, Hammamet, Tunisia, 2002.

[15]. I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*, Addison-Wesley, 1999.

[16]. Wirfs-Brock, R. and Wilkerson, B. *Object-Oriented Design*: A Responsibility-Driven Approach. In Proceedings of OOPSLA '89 Conference. SIGPLAN Not. (ACM) 24, 10, (New Orleans, Louisiana, October 1989), 71-76.

[17]. B. H. L Betlem R. M Van Aggele, J. Bosch; J. E Rijnsdorp, "*An Object Oriented Framework for Process Operation*", Technical Report, Dept. of Chemicals Technology, University of Twente, 1995

[18]. Papers and articles of the IEEE, Elsevier ACM, World Scientific and Springer publishers