ISSN No. 0976-5697

Volume 5, No. 7, September-October 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Interoperability in the JVM and CLR Engines for Cross Languages Application Developments

Sonar Sanjay Bhagwan
Ph.D Scholar Rai University,
Saroda, Dholka Taluka, Ahmedabad, Gujarat, India

Abstract: Engines of the Java Virtual Machine (JVM) and Common Language Runtime (CLR) are designed on the basis of multi-platform mechanism and as well as compatible of the cross languages interoperability based on the pure object orientation for both JVM and CLR related languages. Both engines languages are compatible for application domain specification, requirements elicitation, Genericity in all objects specification, designing mechanism of pure object orientation and Methodological implementation. The paper proves common mechanism and methodological design-to-implementation steps by pure object orientation, to design database and application using JVM and CLR languages. Paper also proves genericity in relationship between compatible mechanism of the CLR and JVM platform. Both engines has compiled and interpreted mechanism, cross languages feature, as well as multi-platform mechanism. The JVM platform is java based languages as Java, JRuby and Scala; and CLR platform is VS.NET based languages as vb.net, c#.net and vc++.net and other. There are no business logic is applied on the design-to-implementation steps for designing the system in both engines, but common tools is mechanized and sequentialised to design the applications and common database for JVM and CLR related languages.

Keywords: Interoperability, compatibility, domain, genericity, Cross-Platform, actor, Use Cases.

I. INTRODUCTION

Interoperability in the pure object oriented Cross-Languages, like as Java Virtual Machine based and Common Language Runtime based languages. The pure object oriented cross language as well as Platform independent Languages are purely object dependable. Each task of the application is treated as objects and here object itself is generic. The object is atom as well as object attribute either it is based on instance or non-instance, from the specific domain, then faced with object classes, that is abstracted from the business logic or SRS from the domain specification; also reusability, genericity, polymorphism, class binding, and generalization are mechanized by the pure objects orientation design[1,3]

In both engines are designed on the basis of pure object orientation as well as both engines are also compatible and interoperable for pure object orientation. The object is treated as tangible unit as static and dynamic, faces in the class, operations, and methods, therefore the object in pure object oriented design, treated as generic itself. Also both engines languages mechanism of developing applications and database should be common as compatible, so generic domain identification and object identification should be more specific and genericity, for both engines languages as common database design and common application design, also object classes, methods, operations as well as component design and collaboration design would be generic for common use for both platforms; and therefore it should be tightly coupled with both platforms. Paper prove the common domain and all object specification with specific object build in domain/sub domain as based on the commonness for design the application and database in both platform languages[1,8].

A 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 [1, 6].

II. PROBLEM DEFINITION

Significant drawbacks as in present object-oriented design methods only deal with the design of specific application on selected platform and selected language; And does not facilitate the design of commonness for CLR and JVM engines as the based on the pure object orientation for cross languages. And also no interface oriented object system design for software implementations and database implementation on the pure cross-languages platforms, also no any specification of the object limitation, across the system on the based on the object paradigm for both engines languages. Also present tools and methods/tasks of the object oriented system are not incorporated and interface oriented as common in both engines languages, also not step next tools, to design the system from design-to-implementation for the Cross-Platform software development in both JVM and CLR languages. 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. In present object oriented systems for developing system in pure cross languages platforms; the requirements to design the common system for Pure Object Oriented Cross Languages of JVM and CLR engines, are pointed as below.

a. Object Identification and Specification

b. Single object/class objects/ control object/ link object/ attribute objects/atom objects
c. Non-instances and instances objects.
d. Classification of the objects by domain.
e. Domain Specification as area of work and decomposition of domain by objects

f. Specification in actor and Use Case Scenario for domain specification.
g. Specification of object limit in the domain.
h. Indentify common Non-instances and instances objects. Common interface oriented architecture of the objects
i. Step next tools to precede design-to-Implementation and interfaces components Collaboration of the entire system design.
j. Compatibility in commonness in cohesion and coupling for both engine languages.

A. Previous Research on Commonness:

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 [3].

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 [3, 9].

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

III. COMPARISON OF JVM AND CLR ENGINES

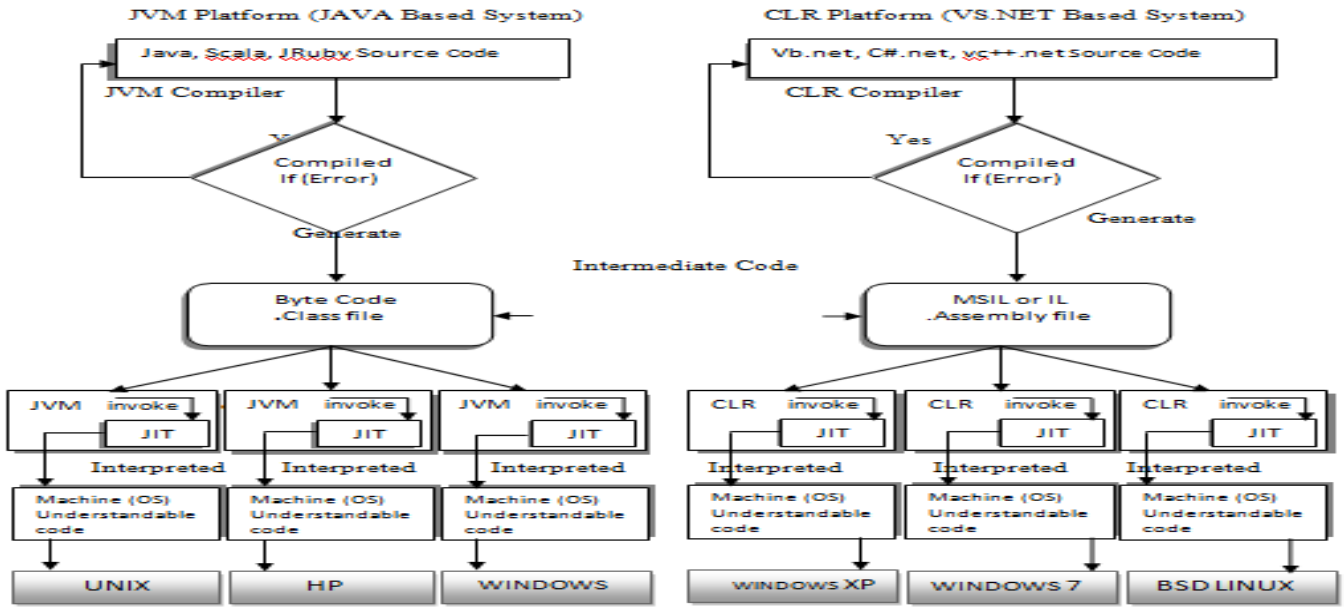

Figure 1. Comparison of JVM and CLR engines

Design for the comparison between java based and micro-soft Visual Studio based platform. Both platforms generate intermediate code for platform independence and language independence; both engines have its own JIT (Just In Time) Compiler, get invoked by the JVM and CLR engines. JIT generate Native code as own machine (OS) understandable code for interpreted as output. The CLR developed for UNIX and LINUX Operating System by UN Berkeley's *Berkeley System Distribution* (BSD) [7, 9, 10].

IV. METHODOLOGICAL IMPLIMENTATION

The following six steps are mechanized of the pure object orientation for the developments applications with database in both JVM and CLR related cross languages. The following design engineering steps are interface oriented from domain specification to Step next tools to precede design-to- Implementation and interfaces components Collaboration of the entire system design for developing application in both platforms.

A. Genericity in domain identification:

This is first step of the Methodological Implementation, Using this step the generic domain would be identified for both platforms. The common area is justified and abstracted. As well as area wise all objects is mechanized for further specification of Collaboration model and component interface design.

B. Domain Specification and Limitation:

After the domain specification, Using this step, identify and specify all common generic objects like instance/Non-instance objects, control objects, attribute objects, atom, anti-atoms, object classes, Major and sub classes modules

and groups etc. This specification is based on the genericity of the objects but not get from requirement elicitation and specified business logic. The domain specification and limitation cover the all generic area of the domain for all object identification, that is designed-to-implementation for java based and dot.net based applications with common databases. The Actors, Use Case and Scenarios tools are used for the solution of the domain specification and limitation.

C. Requirements Elicitation Engineering:

Requirements Elicitation identifies the common Actors, Scenarios (General instance, directly applied to application), Use Cases Refinements (Specify common scenarios), identify the relationship among Actors and Use Case Instances, and indentify common initial objects as well as functional and non functional requirements.

The first step of requirements elicitation is identification of actor. An actor can be human or external system or software agents that are directly concerns with system specification. Actor also defines the boundaries of the system and to find all perspectives from which the developers need to consider the system [5, 6].

Scenarios are the concrete, focused information description of the single features of the system from the viewpoint of the single actor. Scenarios focus specific instances, also Scenarios provide Common tools for the requirements elicitation engineering. The numbers of scenarios are used for common requirements elicitation as like As-is-Scenarios, Visionary Scenarios, Evaluation Scenarios. A Scenario is the instance of the use case that is a use case specifies all Scenarios for a given piece of functionality. A use case is initiated by the actor, as well as use case represents the complete flow of the events through the system in the sense that it describes a series of related interactions that from its initiation.

The relationship among Actors and Use Case Instances to reduces the complexity of the model and increasing its understandability.

After the Indentify and specify the Actors and related use case instances, the initial objects can be abstracted from them. For that the participating objects are abstracted from each use case. The design engineer identifies the names of the object from the work of use case. Also identify the non-functional requirements that are not directly concerns with system.

D. Generic object designing:

Decomposition of the problem into objects depends of the Heuristics or judgment and nature of the problem. The common object identification for the both technology is based on the pure object oriented systems. Using pure object identification, the object itself is generic as like instance or non-instance object, control or link object, atom or anti-atom object, attribute or variable object etc. and collection of the objects are object class or class as major or sub classes. Also in pure object orientation the JVM and CLR both engine are treated object as class and class treated as object as well as atom is treated as variable and it treated object for instance. There is no genericity in the instance now, the genericity concerns with only objects and object class with attributes, operations, methods, Multiplicity, link and associations, rolling, ordering, aggregation. Generalization and qualification.

An Example of **common** class model for CLR and JVM application object specification of the company's employees with employees related other objects, based on pure object orientation [4, 8, 11].

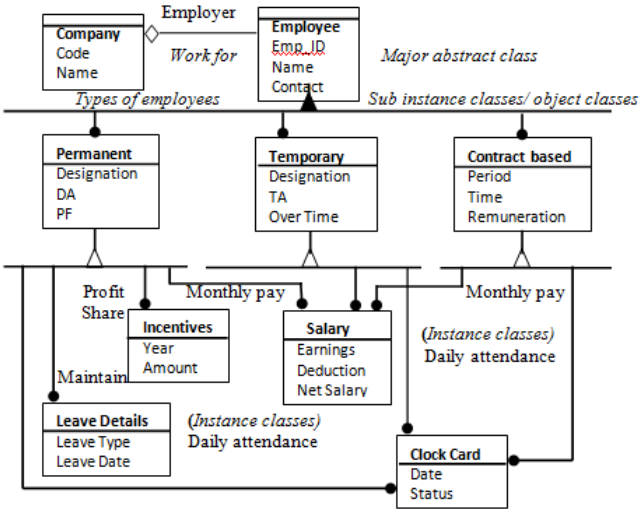

Figure 2. Common generic object class specification.

The following common factors are treated and operated by the Generic object designing.

a. The Clean room of the object and/or object class and/or object as attribute and/or instance/non-instance.
b. Specify that objects treated as if class/abstract major class, class (non-instance)/major class/sub classes link (Relationship) class.
c. Multiplicity of the class object as how many instances of one class may related to each instance of the another class.
d. Identify object domain/sub domain and Link and association among the object class
e. Role names of the association with qualification of the each objects.
f. Aggregation and Generalization of the each objects. Candidate key as object's minimum attributes that uniquely identifies as object of link to other objects.

E. Commonness on Coupling and Cohesion:

The commonness between coupling and cohesion as designed as loosely coupling and tightly coupling, the system is divided into number of subsystem. And two or more subsystem is loosely coupling and tightly coupled, depending on the nature of the system. The one subsystem is modified the impact on the other subsystem.

The Cohesion is the number of dependences within a sub system. The cohesion specifies the all subsystem domain objects are interacted with each other as the based on the tightly and loosely coupled. Each subsystem contain number of unrelated objects, The class model and Use case model are tools for domain object cohesion and coupling [2].

F. Common models of Unified Modeling Language:

Unified Modeling Language has nine diagrams for modeling the system, but the following models are clarifying commonness design for both JVM and CLR related languages.

a. ***Class/Object Model:*** The Class diagram describes the system in terms of objects, classes, and members of the class. Also depicts the interface among the classes with above pure identifications, and relations among them. The objects in the class share a semantic purpose, based on the common attributes and operations [2, 6, 12].
b. ***Use Case Model:*** This Model is used for purely during requirements elicitation and analysis of the functionality of the system**.** The model provides the external visible behavior of the system. The model designed for actor and system performance scenario here actor describe entity that interact with the system. Along with what the system does in response. The Use case diagram is basic tool for domain /sub domain specification, object selection in specific domain/sub domain and object division by supporting the actors [2, 13].
c. ***State Diagram:*** This diagram designs by states and events, when events is received the next state depends on the current state as well as event, state changes is caused transition. A state represents a particular set of values for an object, This Diagram is used for common operations, method and states of the each objects of the both engines languages [4].
d. ***Collaboration Model:*** this model interaction with emphasis on relations between objects. Collaboration model represents of the combination of the information taken from class and use case models, and describes the static and dynamic behavior of the object classes. Also Collaboration model show the message flow as well as association between objects. Basically Collaboration Model specifies the domain and co-domain for object distribution [7, 10].
e. ***Component Model:*** This model describes the major architecture of the system. The component diagram depicts the deployable units of the system as applications, components, and data stores. As well as design the interface and associations among the components. This model also specifies the domain and co-domain for object distribution as well as actor and use case specification [7, 8, 12].

V. CONSISTENCY BETWEEN OBJECTS DESIGN ENGINEERING AND INMPIMENTATION

When performing the common pure object oriented analysis and design for cross languages we should care about the consistency where there will be a relation among the object designing and implementation. The design should be common for both platforms for application implementation and database implementation. The above six methodological implementation steps are interface oriented and sequentialised for the commonness object design, for implementation application and database in both platform languages. Therefore the object in pure object oriented system treated as generic itself it has many faces for operation statically and sequentially. Also both engines languages are mechanism of developing applications are compatible, so generic object identification is vital for all related objects (instance/non-instance, control, attributes, atomic, link etc.),object class, methods, operations as well as component design and collaboration design would be generic; and therefore it is tightly coupled with both platforms for design database and application object orientation designing.

VI. CONCLUSION

Drawbacks as in present object-oriented design methods only deal with the design of specific application on selected platform and language. And does not commonness for CLR and JVM cross languages design as the based on the pure object orientation and database design. Also present tools and methods/tasks of the object oriented system are not incorporated and interface oriented and not specifically step next tools, to design the system from design to implementation for the Cross-Platform languages of both JVM and CLR engines. The above methodological implementations tools are focused on the commonness design pure object orientation for developing applications in both platform cross languages because object in pure object oriented system treated as generic its self for different faces. The comparison depicts the interoperability and comparability of the JVM and CLR cross platform independence languages commonness. The paper also proves the consistency in design engineering to common database design and pure object oriented application development in both engines languages.

VII. ACKNOWLEDGEMENT

I would like to thanks Dr. Samrat O Khanna, Head of the Departments of ISTAR institute of the Anand-Gujarat, to inspire and Encouraged me to perform my best. And I also Thanks to all Software firms who gave me the best support to abstract my goal.

VIII. REFERENCE

[1]. Gamma, Erich; Helm, Richard; Johnson, Ralph; and Vlissades, John. “Design Patterns”, 2nd Edition, Pearson Education 1994.

[2]. Bernd Bruegge, Allen H. Dutoit. “Object Oriented Software Engineering, Using UML, Patterns, and Java”, Second Edition, Pearson Education, 2010

[3]. 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.

[4]. James Rumbhaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, “Object Oriented Modeling and Design”, Prentice- Hall India Edition, 3rd Edition, 2001

[5]. 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.

[6]. 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), pages 71-76.

[7]. www.it-ebooks.info/book (E-book) Scala/JRuby Programming.

[8]. James J, Odell, "Advanced Object Oriented Analysis and Design by UML", SBN- 9780521648196, Cambridge University Press, 1998

[9]. Deitel & Deitel, Listfield, Nieto, Yaeger,Zlatkina, "C# How to Program", 3rd Edition, Pearson Education, 2005.

[10]. Deitel & Deitel, "Java How to Program", 3rd Edition, Prentice Hall American Edition, 1995.

[11]. Sonar Sanjay B, Dr. Samrat Khanna, "A UML Model for Automation of Counseling System Using Pure Object Oriented Approach", Journal of IAEME, "International Journal of Computer Engineering and Technology", Volume 4, Issue 5, September-October 2013, pp. 15-22.

[12]. Code P. Yourdon. E. "Object Oriented Design", 2nd addition, Yorden press, Englewood Cliffs, N J 1991.

[13]. Schneider Winters, "Applying Use Cases", 2nd addition, Pearson Education .2008.