www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 6 June, 2013 Page No. 1900-1908

COMPARISON OF DIFFERENT UML TOOL: - TOOL APPROACH

Tincy Rani, Sushil Garg

CES Dept, RIMT-IET, CSE Dept, RIMT-IET.

Tisha11sep@yahoo.com,sushilgarg70@yahoo.com

Abstract :-

The Unified Modeling Language (UML) is becoming widely used for software and database modeling, and has been accepted by the Object Management Group as a standard language for object-oriented analysis and design. In This paper We compare different UML tool and there Pros & Cons with case study

.

1. Introduction :-

UML is a general purpose modeling language. It was initially started to capture the behavior of complex software and non software system and now it has become an OMG standard. UML provides elements and components to support the requirement of complex systems. UML follows the object oriented concepts and methodology. So object oriented systems are generally modeled using the pictorial language.UML diagrams are drawn from different perspectives like design, implementation, deployment etc.At the conclusion UML can be defined as a modeling language to capture the architectural, behavioral and structural aspects of a system.Objects are the key to this object oriented world. The basic requirement of object oriented analysis and

design is to identify the object efficiently. After that the responsibilities are assigned to the objects. Once this task is complete the design is done using the input from analysis. The UML has an important role in this OO analysis and design, The UML diagrams are used to model the design. So the UML has an important role to play.

2. Argo Tool:-

ArgoUML was conceived as a tool and environment for use in the analysis and design of object-oriented software systems. ArgoUML is free and open source UML modeling
software. It supports all UML diagram like class diagram, use case diagram, activity diagram, sequence diagram and deployment diagram.

ArgoUML was written in a java. This makes any platform with java 5 or java 6. Argo tool generate XMI files. XMI is a standard file format for UML designs [2]. That will be supported by other tool like SD Metrics tool.

2.1 Feature of ArgoUML:-

2.1. 1 UML Diagram support:- The following diagram types are supported by AgroUML:-

- Class diagram
- State chart diagram

- Activity diagram
- Use Case diagram
- Collaboration diagram
- Deployment diagram
- Sequence diagram

2.1. 2 XMI Support: - XMI is an xml based exchange format between UML tools. ArgoUML uses this as standard saving mechanism so that easy interchange with other tools and compliance With open standards are secured. Additionally, exporting the model to XMI is possible. XMI Version 1.0 was used for UML 1.3. ArgoUML 0.20 imports the UML 1.4 formats XMI 1.1 and 1.2. [3]

2.1. 3 Code Generation:-ArgoUML provides code generation for java, c++, c#, PHP4 and PHP5. Other languages may be added since the code generation is a modular framework. The java code generation works with the java reverse engineering to provide basic round-trip engineering.

Reverse Engineering:-ArgoUML provides a modular reverse engineering framework. Currently java source code is provided by default and there are modules for java jar and class file import.

2.1. 4 Diagram editing:-ArgoUML supports many diagrams editing feature that help you edit UML diagrams.

2.1. 5 Internationalization:-ArgoUML Internationali –zation to American English, British English, French, German, Italian, Portuguese, Spanish, Russian, Norwegian Bokmal and Chinese.

2.1. 6 Several diagram export formats:-Diagrams can be saved as GIF, PNG, PostScript, Encapsulated PS, XMI, PGML and SVG.

2.2 Example of ArgoUML: - Argo tool contains ArgoUML.jar, gef.-0.9.c jar, manifest.mf, nsuml.jar, xerces.jar etc file. Figure 1 show the class diagram of customer's order which is made in Argo tool.

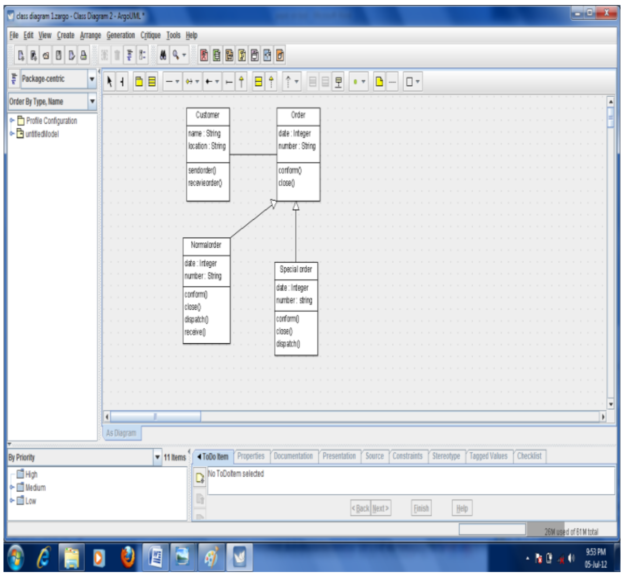


Figure1:-Class diagram of customer's order

2.3 Pros:-

- ArgoUML includes a number of features [1] that supports the cognitive needs object-oriented software designers and architects.
- ArgoUML supports open standards extensively- UML, XMI, SVG, OCL and other.
- ArgoUML is a 100% Pure Java application. This allows ArgoUML to run on all platforms for which a reliable port of java2 platform is available.
- ArgoUML is an open source product, Which allows extending or customizing.

2.4 Cons:-

- Not fully supports UML 2.0.
- Can't Undo! Developers of argoUML must be so optimistic that people (especially Software Developers) never do mistakes.
- Written in Java, so run comparatively slower than starUML.
- Lack of formatting options.

3. **StarUML:** - StarUML is an open source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32 platform. The goal of the StarUML project is to build software

modeling tool and also platform that is a compelling replacement of commercial UML tools such as Rational Rose, Together and so on.

a. Features of StarUML:-

i. Accurate UML standard model: - StarUML strictly adheres to the UML standard specification specified by the OMG for software modeling. Considering the fact that the results of design information can reach 10 years or more into the future, dependence on vendor-specific irregular UML syntax and semantics can be quite risky. StarUML maximizes itself to order UML 1.4 standard and meaning, and it accepts UML 2.0 notation on the basis of robust Meta model**.**

ii. Open software model format: - Unlike many existing products that manage their own legacy format models inefficiently, StarUML manages all files in the standard XML format. Codes written in easy-to-read structures and their formats can be changed conveniently by using the XML parser. Given the fact that XML is a world standard, this is certainly a great advantage, ensuring that the software models remain useful for more than a decade.

iii. True MDA support: - StarUML truly supports UML Profile. This maximizes extensibility of UML, making modeling of applications possible even in areas like finance, defense, e-business, insurance, and aeronautics. Truly Platform Independent Models (PIM) can be created, and Platform Specific Model (PSM) and executable codes can be automatically generated in any way.

iv. Applicability of methodologies and platforms: - StarUML manipulates the approach concept, creating environments that adapt to any methodologies/processes. Not only the application framework models for platforms like .NET and J2EE, but also basic structures of software models (e.g. 4+1 view-model, etc.) can be defined easily.

v. Excellent extensibility: - All functions of the StarUML tools are automated according to Microsoft COM. Any language which supports COM (Visual Basic Script, Java Script, VB, Delphi, C++, C#, VB.NET, Python, etc.) can be used to control StarUML or develop integrated Add-In elements.

vi. Software model verification function: - Users can make many mistakes during software modeling. Such mistakes can be very costly if left uncorrected until the final coding stage. In order to prevent this problem, StarUML automatically verifies the software model developed by the user, facilitating early discovery of errors, and allowing more faultless and complete software development.

vii. Useful Add-Ins: - StarUML includes many useful Add-INS with various functionalities: it generates source codes in programming languages and converts source codes into models, imports Rational Rose files, exchanges modeling information with other tools using XMI, and supports design patterns. These Add-Ins offer additional reusability, productivity, flexibility and interoperability for the modeling information.

3.2 Example of StarUML: - StarUML is mostly written in Delphi. However, StarUML is multi-lingual *project* and not tied to specific programming language, so any programming languages can be used to develop StarUML. (For example, C/C++, Java, Visual Basic, Delphi, JScript, VBScript, C#, VB.NET ...). . Figure 2 show the class diagram of customer's order which is made in StarUML tool.

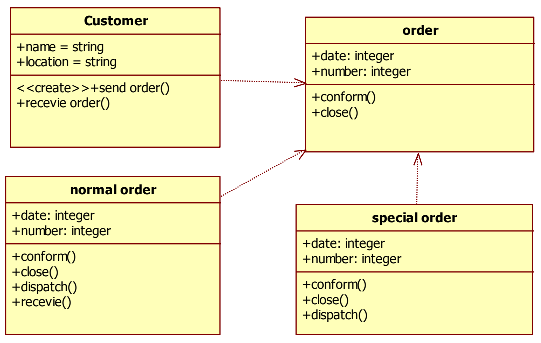


Figure 2 Class diagram of customer's order

3.3 Pros:-

- Supports most of the diagrams specified in UML 2.0.
- Very rich feature set and formatting options.
- Ability to generate source code from the UML diagram.
- Reverse engineer the existing code into UML diagrams.
- Supported languages: C++, C# and Java.
- Fast load time/execution time compared with other UML tools.
- Familiar *Visual Studio* like user interface.
- Supports exporting diagrams into JPG / XMI formats.

3.4 Cons of StarUML:-

- Does not support exporting diagrams into SVG format.

4. Umbrello Tool :-

Umbrello UML Modeller is a UML diagram tool that can support you in the software development process. Especially during the analysis and design phases of this process, UmbrelloUML Modeller will help you to get a high quality product. UML can also be used to document your software designs to help you and your fellow developers.[4]

Having a good model of your software is the best way to communicate with other developers working on the project and with your customers. A good model is extremely important for medium and big-size projects, but it is also very useful for small ones. Even if you are working on a small one man project you will benefit from a good model because it will give you an overview that will help you code things right the first time.

UML is the diagramming language used to describing such models. You can represent your ideas in UML using different types of diagrams. Umbrello UML Modeller 1.2 supports the following types:

- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- Use Case Diagram
- State Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram

4.2 Feature of Umbrello Tool :-

4.2.1 Copying objects as PNG images

Apart from offering you the normal copy, cut and paste functionality that you would expect to copy objects between different diagrams, Umbrello UML Modeller can copy the objects as PNG pictures so that you can insert them into any other type of document. You do not need to do anything special to use this feature, just select an object from a diagram (Class, Actor, etc.) and

copy it (**Ctrl-C**, or using the menu), then open a KWord document (or any program into which

you can paste images) and select **Paste**. This is a great feature to export parts of your diagram as

simple pictures.

4.2.2 Exporting to an Image

You can also export a complete diagram as an image. The only thing you need to do is select the diagram you want to export, and then the option **Export as Picture** from the **Diagram** menu.

4.2.3 Printing

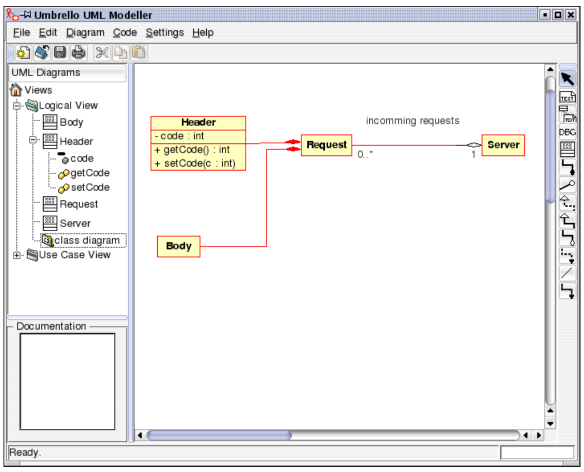
Umbrello UML Modeller allows you to print individual diagrams. Press the **Print** button on the application toolbar or selecting the **Print** option from the **File** menu will give you a standard KDE Print dialog from where you can print your diagrams.

4.2.4 Logical Folders

To better organize your model, especially for larger projects, you can create logical folders in the Tree View. Just select the option **New**! **Folder** from the context menu of the default folders in the Tree View to create them. Folders can be nested, and you can move objects around by dragging them from one folder and dropping them into another.

4.3 Example of Umbrello Tool (class diagram) :-

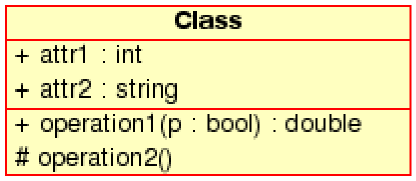
Class Diagrams show the different classes that make up a system and how they relate to each other. Class Diagrams are said to be "static" diagrams because they show the classes, along with their methods and attributes as well as the static relationships between them: which classes "know" about which classes or which classes "are part" of another class, but do not show the method calls between them.



Umbrello UML Modeller showing a Class Diagram

A Class defines the attributes and the methods of a set of objects. All objects of this class (instances of this class) share the same behavior, and have the same set of attributes (each object has its own set). The term "Type" is sometimes used instead of Class, but it is important to mention that these two are not the same, and Type is a more general term.

In UML, Classes are represented by rectangles, with the name of the class, and can also show the attributes and operations of the class in two other "compartments" inside the rectangle.



Visual representation of a Class in UML

Attributes[5]

In UML, Attributes are shown with at least their name, and can also show their type, initial value and other properties. Attributes can also be displayed with their visibility:

- + Stands for *public* attributes
- # Stands for *protected* attributes
- - Stands for *private* attributes

Operations

Operations (methods) are also displayed with at least their name, and can also show their parameters and return types. Operations can, just as Attributes, display their visibility:

- + Stands for *public* operations
- # Stands for *protected* operations
- - Stands for *private* operations

Templates

Classes can have templates, a value which is used for an unspecified class or type. The template type is specified when a class is initiated (i.e. an object is created). Templates exist in modern C++ and will be introduced in Java 1.5 where they will be called Generics.

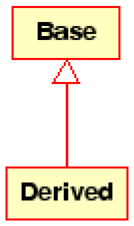
Class Associations

Classes can relate (be associated with) to each other in different ways:[6]

Generalization

Inheritance is one of the fundamental concepts of Object Oriented programming, in which a class "gains" all of the attributes and operations of the class it inherits from, and can override/modify some of them, as well as add more attributes and operations of its own.

In UML, a *Generalization* association between two classes puts them in a hierarchy representing the concept of inheritance of a derived class from a base class. In UML, Generalizations are represented by a line connecting the two classes, with an arrow on the side of the base class.



Visual representation of a generalization in UML

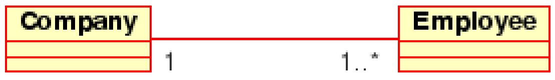
Associations

An association represents a relationship between classes, and gives the common semantics and structure for many types of "connections" between objects.

Associations are the mechanism that allows objects to communicate to each other. It describes the connection between different classes (the connection between the actual objects is called object connection, or *link*.

Associations can have a role that specifies the purpose of the association and can be uni- or bidirectional (indicates if the two objects participating in the relationship can send messages to the other, of if only one of them knows about the other). Each end of the association also has a multiplicity value, which dictates how many objects on this side of the association can relate to one object on the other side.

In UML, associations are represented as lines connecting the classes participating in the relationship, and can also show the role and the multiplicity of each of the participants. Multiplicity is displayed as a range [min..max] of non-negative values, with a star (*) on the maximum side representing infinite.

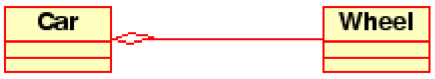


Visual representation of an Association in UML

Aggregation

Aggregations are a special type of associations in which the two participating classes don't have an equal status, but make a "whole-part" relationship. An Aggregation describes how the class that takes the role of the whole, is composed (has) of other classes, which take the role of the parts. For Aggregations, the class acting as the whole always has a multiplicity of one.

In UML, Aggregations are represented by an association that shows a rhomb on the side of the whole.

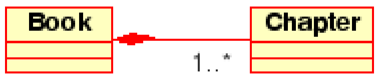


Visual representation of an Aggregation relationship in UML

Composition

Compositions are associations that represent *very strong* aggregations. This means, Compositions form whole-part relationships as well, but the relationship is so strong that the parts cannot exist on its own. They exist only inside the whole, and if the whole is destroyed the parts die too.

In UML, Compositions are represented by a solid rhomb on the side of the whole.



Other Class Diagram Items

Class diagrams can contain several other items besides classes.

Interfaces

Interfaces are abstract classes which means instances can not be directly created of them. They can contain operations but no attributes. Classes can inherit from interfaces (through a realisation association) and instances can then be made of these diagrams.

Datatypes

Datatypes are primitives which are typically built into a programming language. Common examples include integers and booleans. They can not have relationships to classes but classes can have relationships to them.

Enums

Enums are a simple list of values. A typical example is an enum for days of the week. The options of an enum are called Enum Literals. Like datatypes they can not have relationships to classes but classes can have relationships to them.

Packages

Packages represent a namespace in a programming language. In a diagram they are used to represent parts of a system which contain more than one class, maybe hundreds of classes.

5. Rational Rose:-

Rational Rose is a commercial case-tool software. It supports two essential elements of modern software engineering: component based development and controlled iterative
development. Models created with Rose can be visualized with several UML diagrams.

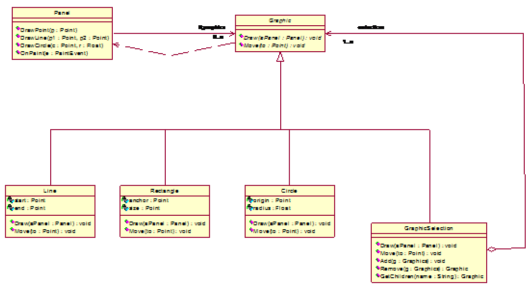
Rose also supports Round-Trip engineering with several languages.

5.1 **Why and where was Rational Rose used[7]**

The usage of Rational Rose was due to a subgoal of the project *Smart*. The goal was to become familiar with several products of Rational Software Corporation. The retrieved knowledge was also used to hold a presentation on this seminar about reverse engineering.

Rational Rose was used as a case-tool in the project *Kahvinheitin* where the idea was to
create a software for a microprocessor based coffee maker. Project *Kahvinheitin* can be
considered as a subproject of the project *Smart*. In the project Rose was used visually to create class-, state- and packet diagrams. Rose's Round-trip engineering capabilities were also examined.

5.2 Example of Rational rose Tool:-



5.3 **Pros:-** Team Development[8]

- One of the main advantages of Rational Rose is that it facilitates team development by providing full team support. It easily allows users to work with their own unique version of the model in their own workplace, without moving from one place to another.

Development Process

- The software can easily be used throughout the whole software development process, unlike other software. Rose can be used at any stage during the development process, as well as using it to help uncover and prevent potential serious mistakes in the future.
-

Model Management

- Managing model changes is also made simple by Rational Rose. Changes made to a model can be made available to others by using a configuration management and version control (CMVC) system. This allows easy integration of changes into the model without interfering with any developmental stage.

Legacy Problems

- Rational Rose addresses bad legacy problems; it lets you go back and correct mistakes and flaws within the legacy application. This is useful when facing software that doesn't fit the users' needs.

Project Documentation

- Rational Rose allows the user to save on creating additional project documentation by using the models created in the software as a basis for design and development. This is a good way to avoid poor documentation practices. It includes ready-built frameworks for different models, as well as a set of reusable components. In addition, it provides templates for creating new models, something that many users enjoy.

Add-Ins

- One of the advantages of Rational Rose is the add-in feature. This allows the user to install programming languages in order to generate necessary codes. Several add-ins can be installed, such as C++, PowerBuilder, Forte, Java, Visual Basic, Oracle 8/9 and XML. Add-ins in the form of nonlanguage tools can also be installed, such as the Microsoft Project. To manage model changes, the add-in feature can be used to install Rational's ClearCase and Microsoft's Visual Source Cafe. A variety of add-ins are available, and the great advantage is that the user can deactivate any of the add-in features he does not need while working on a model.

Configuration

- One of the great advantages about Rational Rose is that the user can configure the interface and tailor the application to suit her needs. Rose uses a graphical user interface (GUI) that includes a browser, diagram and document windows, as well as standard and diagram toolbars. It always makes for a better work environment when the user feels comfortable with her interface and application.

5.4 Cons: -

- At first the tool seems to be quite complex.
- Some minor bugs were found.
- Separate tool had to be used (and learned) to reverse-engineer files.
- Layout manager could have been a bit more effective.
- Generated code was a bit obfuscated

6. Conclusion

In this paper we have described features of different Uml Tool (Argo UML, Star UML, Umbrello, Rational Rose) with case Study .
In Future we developed XMI Convertor tool and using that Convertor tool we can Measure Coupling of UML diagram.

7. References :-

.[1] Beck, K. and Johnson, R. Patterns generate architectures. Proc. European Conf. on Object-Oriented Programming(ECOOP'94). Bologna, Italy. 1994.

[2] Bonnardel, N. and Sumner, T. Supporting evaluation indesign: the impact of critiquing systems on designers ofdifferent skill levels. ActaPsychological. vol. 91. 1996. pp.221-244.

[3] Chun, H. W. and Lai, E.M.-K. Intelligent critic system for architectural design. Trans. Knowledge and DataEngineering. July 1997.

[4] K. Toth, "Software Product Evolution in the Classroom," Proc. American Soc. Eng. Education/PSW Conf., California State Univ., Fresno, 2002.

[5] K.C. Toth, "Simulating (Software) Product Evolution in the Classroom," Proc. 6th Western Canadian Conf. Computing Education (WCCCE 2001), Nelson, 2001, pp. 45&ndash,49.

[6] W.S. Humphrey, A Discipline for Software Engineering, Addison-Wesley, 1995.

[7] http://www.rational.com/rose --Information on IBM Rational Rose,® A commercial UML modeling tool.

[8] http://www.rational.com/xde --Information on IBM Rational XDE,® a commercial UML modeling tool that is integrated with IBM's Eclipse development platform.

[9] OMG Unified Modeling Language Specification, ver. 1.5, OMG Unified Modeling Language Revision Task Force, Mar. 2003, www.omg.org/technology/documents/formal/uml.htm.

[10] Rational Rose Family, IBM/Rational Software Corp., 2003,www.rational.com/products/rose/index.jsp.

[11] Rational XDE, IBM/Rational Software Corp., 2003, www.rational.com/products/xde/index.jsp.

[12] S. Mellor and M. Balcer, Executable UML: A Foundation for Model-Driven Architecture,Addison-Wesley, 2002.

[13] G. Sunyé, et al., "Using UML Action Semantics for Executable Modeling and Beyond," Advanced Information Systems Eng.: 13th Int',l Conf. (CAiSE 01), LNCS 2,068, Springer-Verlag, 2001, pp. 433-447.

[14] XSL Transformations (XSLT) Version 1.0, W3C, www.w3.org/TR/xslt, 2003.

[15] D. Milicev, "Domain Mapping Using Extended UML Object Diagrams," IEEE Software, vol. 19, no. 2, Mar./Apr. 2002, pp. 90-97.

[16] Agrawal, G. Karsai and F. Shi, "A UML-Based Graph Transformation Approach for Implementing Domain-Specific Model Transformations," to be published in Int',l J. Software and Systems Modeling, 2003.

.