Object Oriented Graphic Frameworks

Dr. Hari Ramakrishna, Professor
Department of Computer Science and Engineering
Chaitanya Bharathi Institute of Technology
Gandipet, Hyderabad, A.P, INDIA

Abstract

This Paper presents an object oriented graphic framework architecture named as white box graphic pattern-frame. Few applications of the model and sample code segments along with few result screen shots of applications are presented. Graphic object basic functionality and domain specific graphic component behavior management is demonstrated. These frameworks uses several object oriented primitive patterns such as dynamic polymorphism for managing multiple behaviors. A method of packing and exporting the framework to different clients using Microsoft middle ware frameworks like Document view architecture, ATL and ActiveX frameworks is demonstrated. This paper presents applications of such framework in building intelligent graphic simulation application where graphic components implements semantic behavior of domain specific objects.

Keywords- graphic frameworks, pattern-frames, semantic graphic components, White box graphic pattern frames.

I. INTRODUCTION

Object oriented application frameworks can significantly increase software quality and reduce development effort. However, a number of challenges to be resolved in order to present the failure of such frameworks. Some of the important aspects to be considered in this regard are development effort, learning curve, maintainability, validation and defect removal, efficiency, and standards definitions. The following issues should be considered carefully while redesigning any object oriented graphic framework.

i) Development effort of the domain application developer should be simplified

ii) Learning curve for understanding such frameworks should be simplified

iii) Integrating the framework with different application environment should be considered

iv) Framework maintenance activities include modification and adaptation of the frameworks should be carefully considered

v) Generalization of behavior graphical components for abstraction should be handled carefully

vi) Framework exhibits Inversion of control handling of such software need special efforts

vii) Efficiency of the code, dynamic binding nature of frameworks should be carefully handled

viii) Standards should be well defined

Some of the expectations of such frameworks are listed below:

i) Integration of design patterns and defining pattern frames

ii) Defining pattern languages

iii) Defining procedures for integrating frameworks with state of art middle ware development environments like Microsoft development environment such as Application Frameworks, ATL, Active X , .NET or WPF.

This paper presents a model lightweight computer graphic framework which can be used for configuring to build several domain specific graphic components. These are useful for several software applications developed user modern middleware development frameworks for supporting simulation and graphic requirement. These models use several primitive object oriented patterns in addition to some domain specific pattern-frames. [1-6].

II. TYPICAL FRAMEWORK SAMPLES

This section presents few Framework samples, starting from a simple hello world application using java frameworks. These are known as middleware development frameworks. Figure 1 presents the UML building block for representing a framework.

Consider a hello world program in Java. An example can be implemented using java Applet. The Java Applet is a part of Java Framework. This in turn depends on AWT and Java language frameworks. The following

Figure 2 represents a hello world component structure in UML using Java frameworks.

From this diagram, it can be observed that Hello- world applet depends on Java Applet. The HTML client or Java frame which includes the Hello-World Applet gets Hello-World Interface through the Java Applet. Message from client application will invoke the Java Applet that in turn sends them to the Hello-world Applet. For displaying the hello world message the Hello-world applet implementation again depends on graphic library, which is part of Java framework. The class diagram of the hello world example is displayed in the following Figure .3

Consider another sample of frameworks from Microsoft Document View Architecture and MFC. Microsoft provides Application Wizard for using the framework and class wizard for managing the applications. A simple MFC based application structure in UML is presented in the following Figure 4.

This UML Diagram represents a logical structure of Document View Architecture. The application class of the client module is inherited from CWinApp, a class of Microsoft MFC framework. In fact the Application wizard decides from which class of CWinApp class group the MyApplication class should be inherited, depending on the requirement of the client specified through application wizard. The user requirements are collected in six steps at the time of creating an application framework in VC++ through application wizard. The type of project workspace also changes the aggregation-combination, depending on how the user is exporting functionality.

The ATL technology of Microsoft also provides similar frameworks for supporting Automation layer; component technology and web based computing. Some of the frameworks presented in this thesis also depend on these Microsoft frameworks. These frameworks are reffered to as Middleware integration frameworks.

The above application represented in Figure 2.4 has an Application class aggregating Frame class. The frame class inturn aggregates View and Document classes. The Application, Frame, Document and View classes are inherited from CWinApp, CMainFrame, CDocument and CView classes as shown in Figure 2.4. The CWinApp class manages the Windows application functionality. The CMainFrame class aggregates a set of view and document objects. The view manages device context. It can also manage GUI required functionality.

In addition to creating a Windows based Application with automatic code generation and aggregation with MFC framework, Microsoft frameworks also support class wizard for managing client applications. The message maps are managed through this class wizard. The resource sub-system helps user in building Menu items, Tool bars, Dialog boxes and Accelerator keys. They also support simple graphic primitives through CDC class, which is an abstract class. CClientDC is a sub class of CDC that can be instantiated from any class inherited from CWin class. This rule is automatically imposed by making CWin pointer, which is an argument for constructor of CClientDC class.

The OLE framework is also integrated to Visual studio such that a simple object oriented programmer also can use OLE features in these applications. The requirements for such facilities are also collected through the application wizard by asking the user, whether the application is an OLE server or OLE container etc.

At present versions of Visual studio DOT NET has several additional features such as multiple language support, Web based distributed and object management facilities. WPF also presents a new model development environment bringing revolution in the presentation layer of windows and web applications.

III. ARCHITECTURE OF WHITE BOX FRAMEWORKS

The intent of White box frameworks is to generate object library for configurable generic domain specific classes. All the framework patterns discussed in the previous sections support required reusable-modules and functions for managing graphic components. But they do not manage any graphic components. White box frameworks manage graphic components.

Name: White Box pattern-frame
Intent: Managing generic reusable and configurable graphic component.
Motivation and Applicability: In general generic modules are not efficient. But code reuse will be more. This affects the cost of the application. But most of the code for managing any graphic component irrespective of the application is common. They change only in Geometry and Behavior. It is observed that creating a

configurable class for graphic element can reuse the common code for managing graphic components.
Structure: Figure 5 presents the Architecture of White box pattern- frames.
Generic Interface: It defines generic behaviour of a graphic component.
Generic graphic class: It implements the generic behaviour of a graphic component.
Specific Interface: It defines specific interface of a selected graphic component.
Specific Graphic class: It implements the specific interface of a graphic component.

The generic behaviour of a graphic class is implemented by the Generic graphic class. This in turn depends on specific behaviour of selected graphic element. This specific behaviour is implemented by the specific graphic object. The client can reuse the generic as well as specific behaviour of the graphic object. This pattern-frame enables a generic graphic class managing different types of graphic elements. This supports generic behaviour and allows the specific object to define specific behaviour.

IV. IMPLEMENTATION AND SAMPLE CODE

The following graphic class namely CGraphicElement implements a generic graphic framework, which can be configured to graphic components of different domains. This model will not affect the performance of the element. This decreases cost and complexity of the application.
Though this model is based on the Microsoft MFC framework, it can be implemented using any object oriented language with minimum graphic support. The following are observations made on the above graphic class.
i) The above Graphic class is an abstract class as it has a pure virtual function. The clients need to implement or attach geometry to the above graphic base class for building full-fledged graphic component.
ii) The graphic class implements several common graphic object management functions such as Show Key points, Serialize, Locate etc. These functions are used to make an object persistent and for locating and editing the object. The client as per the requirement of his/her graphic element can configure these functions. For example, the key-point function by default shows the range of the graphic element, and will display right bottom, left top and center point of a rectangle, which is range of the graphic element. User can Move the element or scale the element as he likes irrespective of the geometry of the graphic element. If user wants to redefine this function, he can configure it as per his requirement. For example locating a line of a line-component is different from a Rectangle-component. User can implement his own Locate algorithm for locating his element as per his requirement. Without implementing this algorithm, the framework itself will support an algorithm for locating the element.
iii) This framework will decrease the cost of implementation of a graphic-element. These frameworks are more useful for the graphic applications where the numbers of graphic-element types to be managed are more.
iv) Other observation in this model is that the above framework class uses Foundation class framework for defining geometry.
The Line and Ellipse classes are typical simple graphic elements created using the above framework class. The two classes are using default implementation for locating the elements and for editing the elements. What all these classes require to implement is two functions which are specific to application. The first one is a unique Id of the element, which the user is giving to the element in his application. This function is suggested for referring the element. The next function is Geometry of the element that is specific to that element. The following element namely Rectangle configures Locate function. The definition for Rectangle class is presented in Table 2
The IsLocated function can be configured by the client to attaching object specific behavior. For example, if the user does not want to locate a rectangle with three key points (which is a default procedure provided by the framework), and he wants to implement procedure for locating a rectangle with reference to its border, he can adopt this method. Similarly for managing complex graphic components one can change other behavioral futures. The following Polygon definition changes other behavior of framework class namely CGraphElement, which is a generic graphic base-class.
The polygon in Table 2 manages a Point list with the help of a Foundation class for managing Polygon geometry. This will affect behavior of the element. For making the geometry persistent, it implements Serialize method. For supporting Translation operation it is required to implement Move-method. In fact, this function again directs the message to Foundation class, which is supposed to manage the Graphics. Even Serialize is directed to Foundation class. The foundation class can support any number of operations as discussed in function class frameworks. Domain specific function classes can be built as separate tool for managing geometry of different graphic elements

and aggregated with the above graphic class for building domain specific applications. [7-12]

In the above Framework Graphic base class design is visible to the client, and implementation alone is encapsulated. These framework classes are named as white box frameworks. These concepts can be implemented in other languages, like Java that support Object oriented primitive patterns.

This application uses the White-box frameworks for managing the graphic elements, and Foundation class framework for managing Line Attributes. Function-class framework can be used for managing algorithms for locating elements of different shapes. This allows the client to configure the application with his own locating algorithms.

Users: A model VC++ Application view using white box frameworks is shown in Figure 6 and a PCB model is shown in Figure 7

V. CONCLUSION

The object oriented graphic framework presented in this application is implemented in Microsoft development environment. It is tested with different client applications. An Active X control can be built over this framework to use it in web enabled applications and to port the functionality through VB Automation layer. These frameworks can be used for simulation applications. It is tested for a PCB (printed circuit board) testing application. Some more pattern-frames are added to make the graphic components mimic like original PCB components. Figure 7 presents a PCB component designed using this framework while integrating with other pattern frames using dynamic display files. These components are called as semantic graphic components as they implement semantic behaviour of PCB components.

ACKNOWLEDGMENT

The author acknowledges all the professional advisors who motivated for developing graphic frameworks patterns.

REFERENCES

[1] Dr.Hari Ramakrishna, "Design Pattern for Graphic/CAD Frameworks", Ph.D thesis submitted to Faculty of Engineering Osmania University March 2003,

[2] Dr.Hari Ramakrishna and Dr.K.V Chalapathi Rao, "Pattern Methodology of Documenting and Communicating Domain Specific Knowledge", CVR Journal of Science and Technology Vol 2. June 2012 ISSN 2277-3916.

[3] Christopher Alexander, "An Introduction for Object- oriented Design", A lecture Note at Alexander Personal web site www.patternlanguage.com

[4] Pattern Languages of Program Design. Edited by James O. Coplien and Douglas C. Schmidt. Addison-Wesley, 1995

[5] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, "Design Patterns: Elements of Reusable Software Architecture", Addison-Wesley, 1995

[6] LNCS Transactions on Pattern Languages of Programming http://www.springer.com/computer/lncs?SGWID=0-164-2-470309-0

[7] Hari Ramakrishna (1995). Three – dimensional interactive computer graphics package for civil engineering applications, Proceedings of the National Conference on Civil Engineering Materials and Structures, Hyderabad, India.

[8] Hari Ramakrishna (1996). "Applications of Computer Graphics in flooring and wall paper Patterns" First National Conference on Computer Aided Structural Analysis and Design, Hyderabad, Jan' 1996.

[9] Hari Ramakrishna (1996). "Applications of Computer Graphics in Interior Design", Annual proceedings of Institutes of Engineers at Hyderabad, Nov' 1996.

[10] Harrington, S. (1987), Computer graphics: programming approach, McGraw-Hill International, second edition.

[11] Hearn, d. and Baker, M.P (1992). Computer graphics, Prentice Hall, second edition.

[12] Newman,W.S and Sproul, R.S (1981),Principles of interactive computer graphics McGraw-Hill International, second edition.

AUTHOR PROFILE

Dr.Hari Ramakrishna was awarded B.E in Computer Science and Engineering in 1989 by Osmania University, Hyderabad ,A.P INDIA, M.S in Computer Science by BITS PILANI,INDIA and Ph.D. in Computer Science and Engineering by the Faculty of Engineering Osmania University in "Pattern lanaguages for graphic /CAD frameworks". He worked in Software Industry for several years and developed Graphic, CAD /GIS products using Microsoft environment. He has about 15 years of teaching experience . Presently he is working as a Professor for last 7 years in the Department of Computer Science and Engineering at Chaitanya Bharathi Institute of Technology, Hyderabad INDIA.

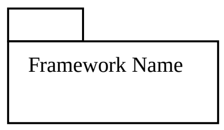


Figure.1: Framework in UML

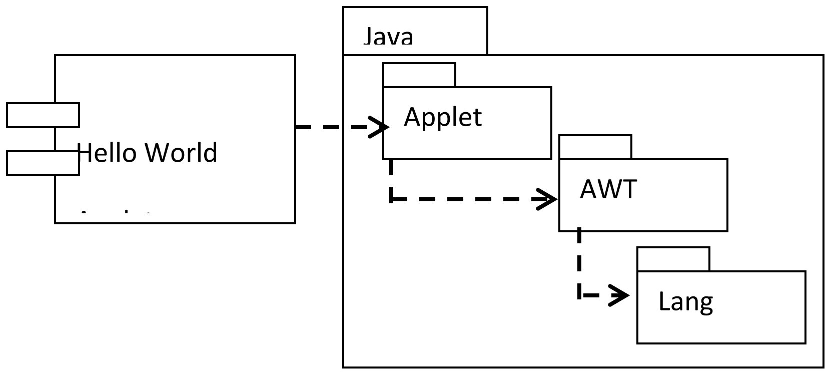


Figure 2: Structure of Hello-World Applet with Java Frameworks

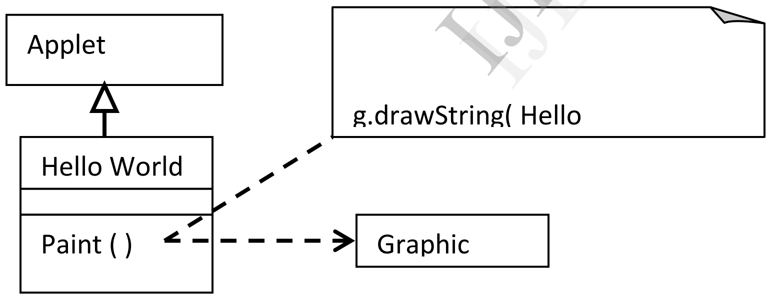


Figure 3: Class diagram of Hello world class

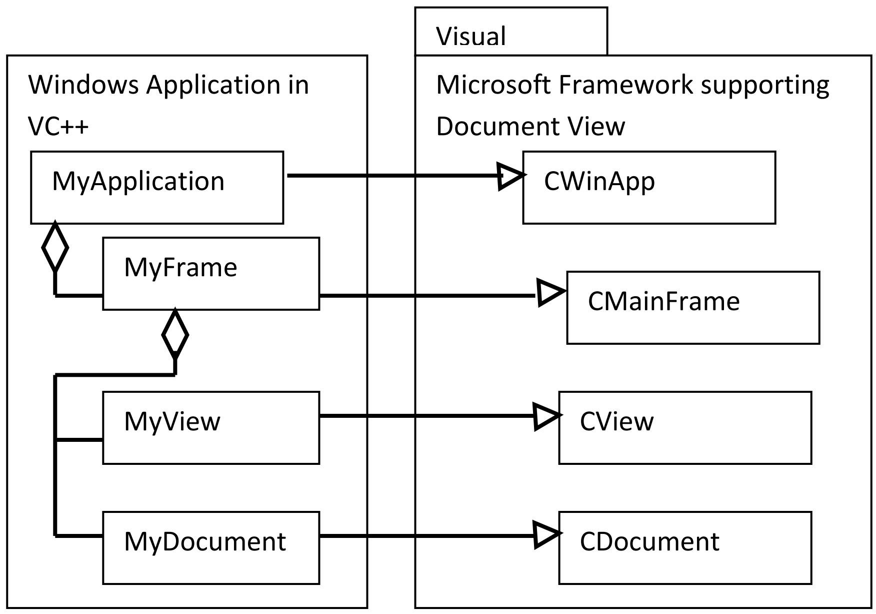


Figure .4: MFC based VC++ Windows's application

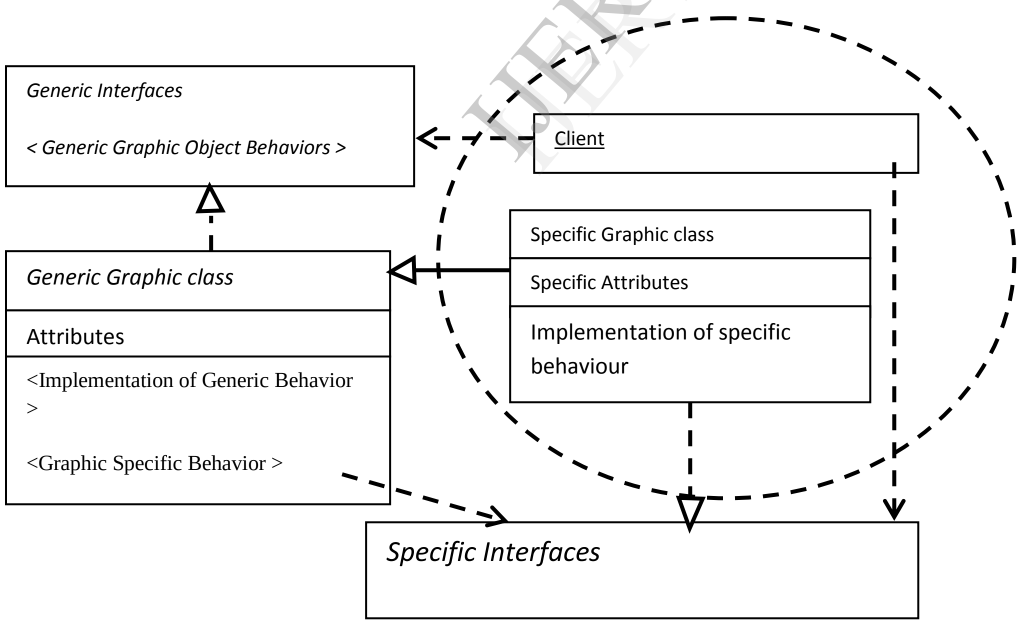

Generic Interfaces
< Generic Graphic Object Behaviors >
Client
Specific Graphic class
Specific Attributes
Implementation of specific behaviour
Generic Graphic class
Attributes
<Implementation of Generic Behavior >
<Graphic Specific Behavior >
Specific Interfaces

Figure .5: Structure of White box Framework

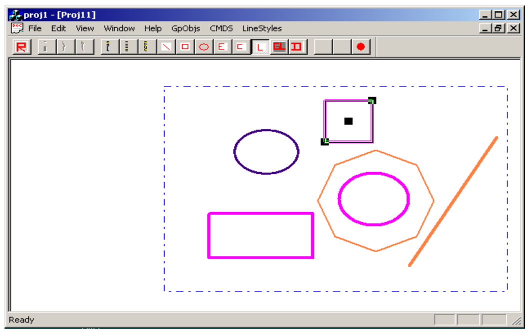
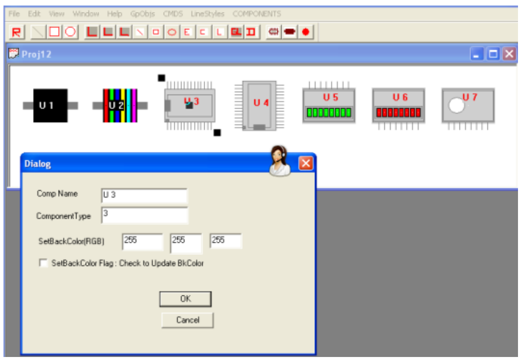


Figure 6 A VC++ client using white box graphic frameworks for managing Graphic element.


File Edit View Window Help GpObjs CMDS LineStyles COMPONENTS
Proj12
U 1
U 2
U 3
U 4
U 5
U 6
U 7
Dialog
OK
Cancel

Comp Name U 3
ComponentType 3
SetBackColor(RGB) 255 255 255
SetBackColor Flag : Check to Update BkColor

Figure 7 A VC++ client using white box graphic frameworks for PCB component design.

Table: 1 Graphic framework class definitions

```
class AbstractGraphic   /* Defines generic Graphic behavior */
{
public:
          void      virtual GetColor(void) = 0 ;
          void      virtual SetColor(COLORREF Rgb) = 0;
          void      virtual SetStyle(int i) = 0;
          void      virtual SetWidth(int i) = 0;
          void      virtual Set_Points(ULONG,ULONG,ULONG,ULONG) = 0;
          void      virtual Mark(CDC*)= 0;
          void      virtual Draw(CDC*)= 0;
          void      virtual Paint(CDC*,COLORREF) = 0;
          void      virtual   ShowKeyPts(CDC*)= 0;
          void      virtual   Move(CDC*,CPoint) = 0;
          void      virtual   Serialize(CArchive& ar);
          BOOL      virtual Locate(CDC*,CPoint,COLORREF) = 0;
          BOOL      virtual IsLocated(CPoint) = 0;
          BOOL      IsInRange(CRect) = 0;
}
class CGraphElement: public AbstractGraphic /* implements Generic graphic behavior */
{
public:
          CGraphElement(void);
          virtual ~CGraphElement(){};
protected:
          CPoint m_pPoint1, m_pPoint2;
          CLineAttribs        m_cLineAttribs;
public:
          void virtual        Show(CDC*) = 0;
          int virtual         GetType(void) = 0;
 };
class CLine :public CGraphElement  /* specific graphic element
{
public:
          void      Show(CDC*) ;
          int       GetType(void);
          BOOL      virtual IsLocated(CPoint);
};
```

```
class CElle :public CGraphElement /* specific graphic element
{
public:
        void    Show(CDC*) ;
        int     GetType(void);
};
```

Table 2: Domain specific Element sample configuring Locate behavior

```
class CRectangle :public CGraphElement
{
public:
        void    Show(CDC*) ;
        int     GetType(void);
        BOOL    virtual IsLocated(CPoint);

};
```

Table 2: Domain specific Element sample configuring domain specific behavior

```
class CPoly :public CGraphElement
{
public:
        CPointList      m_pList;
        CPoly(CPoint point);
        ~CPoly();
        void    Show(CDC*) ;
        int     GetType(void);
        void  virtual   Move(CDC* dc,CPoint pt);
        void    virtual Serialize(CArchive& ar)
};
```

Table 3: Domain specific component class declaration using graphic framework

```
#include "GraphicFramework.h"
class CCmp :public Component
{
public:
        CCmp(void);
        CCmp(int i);
        bool markflag;
        void    Show(CDC*) ;
        int     GetType(void);
        BOOL    virtual IsLocated(CPoint);
        void virtual Design(void);
};
```