Integrated Visual Programming Environment

Hari Om Prakash, R. Phani Bhushan, S. Venkataraman, and Geeta Varadan

***Abstract*—The process of programming includes the creation, compilation, and execution of programs to perform a specific task. The Visual Programming Environment (VPE) provides visual approaches to programming for writing programs graphically, such type of programming platform benefits in rapid prototyping, visual representation of data flow - understanding the logic of the program, and visual debugging. Our approach improves integrated development by way of platform independence combining modules developed using various environments including Microsoft .NET, JAVA, Windows Shell, Win32, COM, MATLAB and Web Services. This paper explores an Integrated Visual Programming Environment (IVPE), including concepts of a network as a system and icon based programming with utility in domain-specific applications such as image processing, algorithm performance measurement, etc. Our newly developed IVPE, Visual Sequencer (VIsual SequencER - VISER) provides an environment for integrating run-time components such as .NET (managed DLL assemblies), Java (class files), MATLAB (DLLs generated by mcc matlab compiler), Win32 (unmanaged DLL libraries), COM objects, and Web Services (WSDL supported). We have also presented some test cases of image processing which utilizes various processing modules from various programming environments.**

***Index Terms*—Visual programming environment, data flow programming, rapid Application development, windows shell, Microsoft, NET framework, java, Win32, COM, MATLAB, windows services, image processing.**

I. Introduction

Program is a sequence of instructions or lines of codes and is required to perform specific task(s). A program can have *n* number of linked libraries, references or components helping to achieve a required task. In this paper we propose an Integrated Visual Programming Environment for representing the flow of execution of various components derived from multiple platforms like Microsoft .NET, JAVA, Windows Shell, Win32, MATLAB, COM, and Web Services. We have developed a platform known as VIsual SequencER (VISER), a VPE where programs are created graphically in a WYSIWYG fashion. In VISER, program refers to a paradigm that is provided by the VPE. With the increase in programming language environments and their syntactical approaches, one should be aware of the syntax.

There are various programming environments for building programs without knowing syntax but one should be having a logical approach which can be an easy process for writing programs.

Visual programming is a solution where we need not be aware of the syntax. The process of visual programming includes both the visual creation of programs and the of programs executing [1]. Key benefits of using VPE are rapid prototyping, requires less programming skills (main focus on program logic than syntax), visual representation of data flow, visual representation of all the activities going with data (like data input, outputs, functions, debugging, execution, etc). With the help of VISER, one can test and measure the performance of algorithms (Section V) and one can find the best performed algorithm among various algorithms.

Manuscript received January 7, 2013; revised April 18, 2013.

The authors are with the Advanced Data Processing Research Institute (ADRIN), Govt. of India, Department of Space, 203, Akbar Road, Tadbund, Secunderabad-500009, Andhra Pradesh (A.P), India (e-mail: hariom@adrin.res.in, phani@adrin.res.in, kalka@adrin.res.in, director@adrin.res.in).

II. Integrated Visual Programming Environment

Shu (1988) defines Visual Programming as the use of meaningful graphic representation in the process of programming [1]. Visual programming is programming in which more than one dimension is used to convey semantics [2], [3]. The process of programming includes both the visual creation of programs and the visualization of programs executing [1]. Benefits of using VPE are its rapid prototyping, visual representation of data flow from creation to its execution (including task debugging, exception handling, what others programming can do). There are various VPEs like VisiQuest [4], Synopsis [5], Ariane [6], and Cantata [7]. VisiQuest provides various VPE features but integration of JAVA class objects, DLLs are not supported. Synopsis has a support of integration of DLLs only not for JAVA Class objects. None of the above existing VPEs provides integration of various components to provide a platform independent environment having support of Microsoft .NET framework, Web Services, JAVA, Win32, COM, MATLAB and Windows Shell.

A. Viser Framework

Fig. 1 shows VISER framework which provides an integrated platform independence model of an application or integrated system functionalities built on any platform including Microsoft .NET, Web Services, JAVA, Win32 APIs, COM, MATLAB and Windows Shell based host scripts. The core component of the framework is its Virtual Execution Environment (VEE) where program flow is parsed and individual activity is executed on their respective platforms.

Responsibilities of VEE are listed as below:

VISER runtime environment: Provides various facilities for running and managing the VISER flows.

Implicit data type conversion: Provides various implicit facilities to convert data from one type to another type. E.g.,

String to Integer, Integer to String, Integer to Long, etc.

DOTNETHelper: Provides various facilities for .NET code functionality extraction and their execution.

JVMHelper: Facilities for running and managing java byte code. JVMHelper engine provides a bridge between .NET and JAVA environment through a Java Native Interface (JNI).

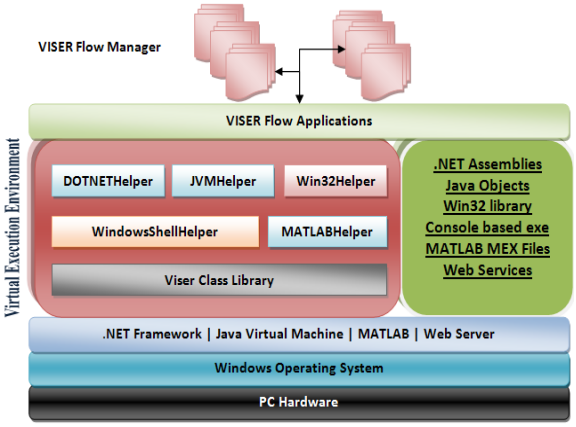

Fig. 1. Viser framework.

Win32Helper: Engine provides interoperability between Win32 libraries and .NET. C/C++ interface has been implemented which is integrated in VISER VEE. Win32Helper is responsible for the management of Win32 function extraction and execution of extern or publically declared functions.

WindowsShellHelper: WindowsShell engine provides the facilities to use host scripts (Shell based executables) to enable successful run at command prompt but the feature provides to use scripts in a visual programming fashion like adding a visual activity (element), assigning executable to it, adding and passing parameter.

MATLABHelper: For running and managing MATLAB MEX binaries. Engine provides an interface to run MATLAB MEX DLLs compiled by *mcc compiler*.

Viser Class Library: Provides various features including user defined Viser flows, data and data access, cryptographic libraries, data base connectivity, network communications, etc.

- Automatic file path handling.

B. Viser Flow Manager (GUI)

Fig. 2 shows VISER GUI which has been developed on WPF (Windows Presentation Foundation), which is a next-generation presentation system for building Windows client applications with visually stunning user experiences [8].

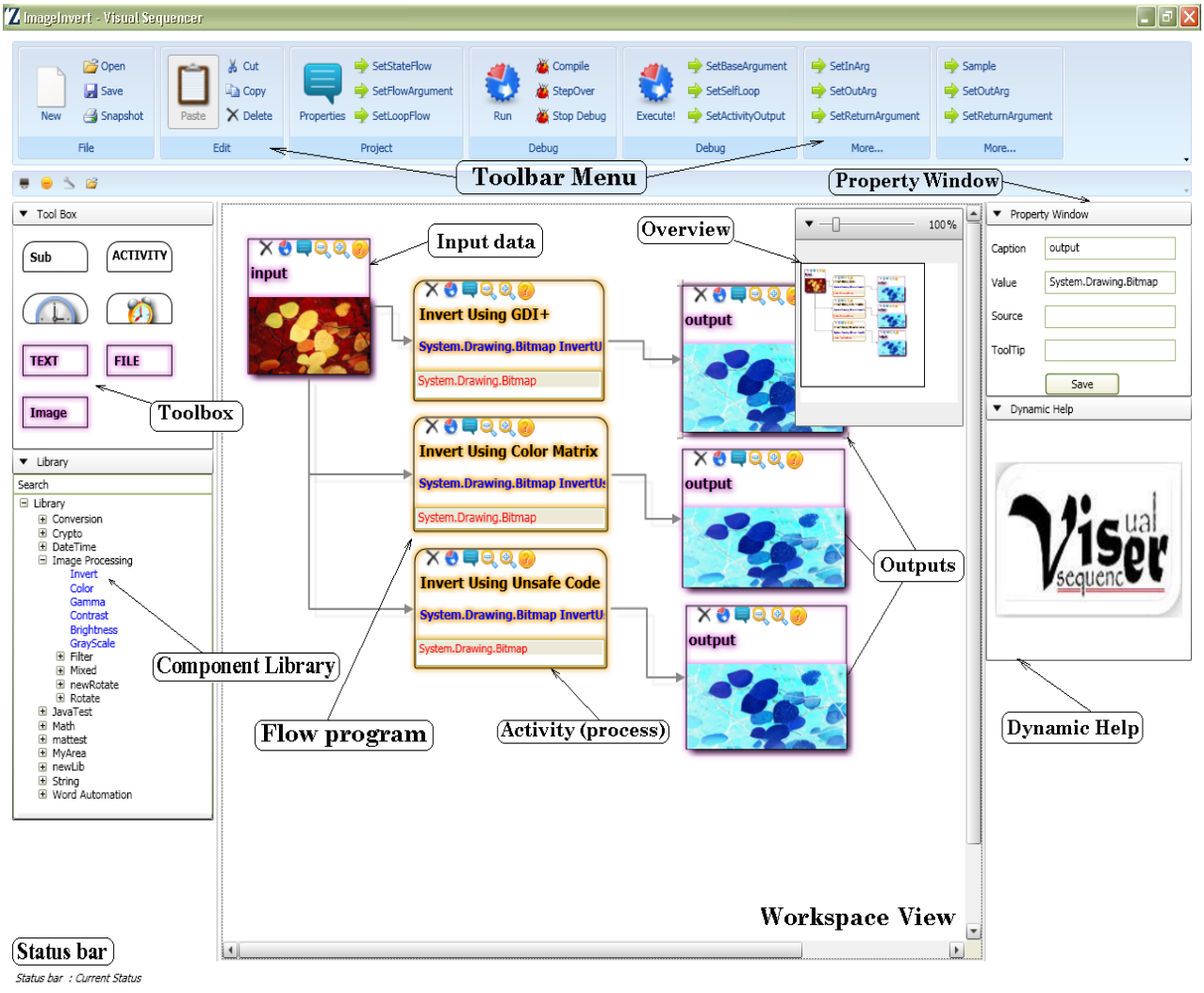

Fig. 2. Viser user interface.

In data flow programming (VPE), all the operations can be performed visually like visually creation of flow programs, argument passing, cut, copy & paste of visual elements. Properties can be set using a property window to individual or selection list visual elements.

III. Programming with Viser

In Fig. 3, flows are depicted which explains how the program flows are programmed on visual programming platform and how they are helpful to understand program logic.

The flow is used for encoding and decoding a text. Encoding visual element (Base64 encoding) is used to encode "INPUT TEXT" which results to generate a base64 text output "SU5QVVQgVEVYVA==". Result of Encoding element is passed to Decoding element (Base64 decoding) which generates a decoded output "INPUT TEXT".

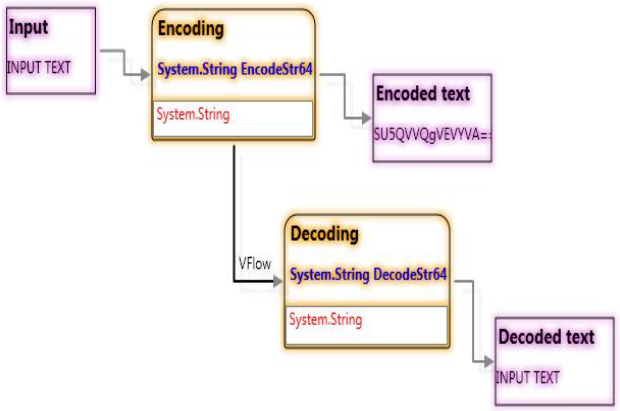

Fig. 3. Encoding & Decoding example

IV. Test Cases

VISER provides a higher level of abstraction where user can add own components from multiple platform binaries to make it common for user specific applications. Main components are listed below:

- Image processing, cryptographic, scientific computations, etc.
- Date Time components.
- Data input and output.

A. Image Processing Samples

VISER provides a separate visual element image I/O which can display input and output images. Processed images can be saved into various standard formats (BMP, JPEG, TIFF, GIF, PNG, ICO, WMF, and EMF). Image processing samples: Fig. 4 and Fig. 5.

B. Distributed Execution: Web Services

Visual Sequencer provides direct support for integration of web services. VISER supports web services in WSDL format. An example is depicted in Fig. 6.

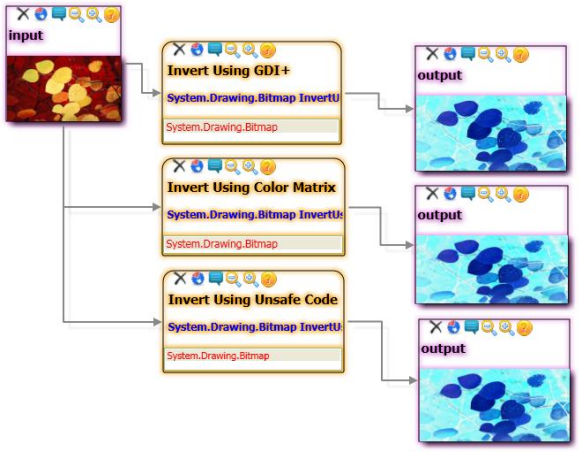

Fig. 4. ImageInvert: image processing – inverting the image using various algorithms (Multi flow)

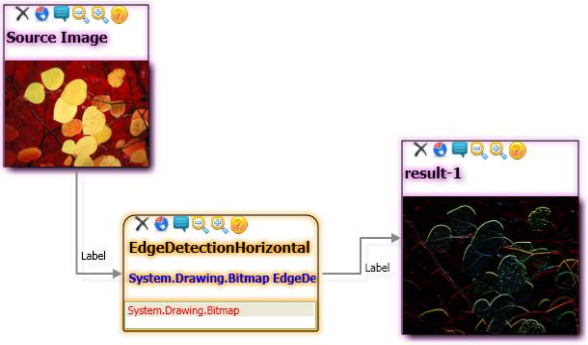

Fig. 5. Horizontal edge detection

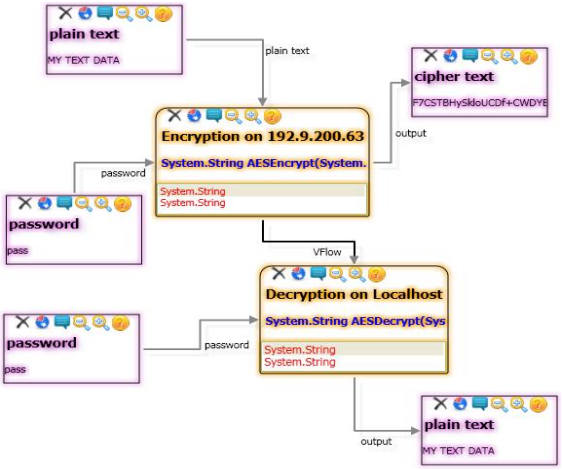

Fig. 6. A visual program with web services

V. Test and Measurement

VISER is not limited to visual programming or data flow programming but one can have the following features:

- Measure the performance of algorithms.
- Comparative study to analyze which is the best algorithm.
- Measure the performance on cross platforms (.NET framework, JVM, etc).
- Measure the performance across the network systems (heterogeneous systems).

VISER embeds a direct support of measuring the performance after the execution of program flow. We tested the flow "ImageInvert" of image processing for the purpose of knowing the best algorithm to invert the bitmap image as depicted in Fig. 4. ImageInvert uses three algorithms:

- InvertUsingGDI+
- InvertUsingColorMatrix
- InvertUsingUnsafeCode (Using pointers).

After execution of ImageInvert project, InvertUsingUnsafeCode takes much lesser time than other algorithms as shown in performance chart Fig. 7. This is because the operations in pointers are faster than others.

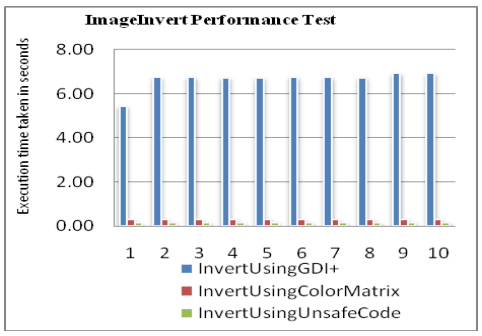

Fig. 7. Performance test for invert algorithms (Fig. 4)

VI. Exception Handling

Successful and unsuccessful activities will be monitored in two levels. Firstly, exceptions will be thrown by individual frameworks with stack trace information and secondly, in VISER framework itself. Exceptions are logged every time when they are thrown and can be checked by right clicking on unsuccessful or red color marked activities as depicted in Fig. 8.

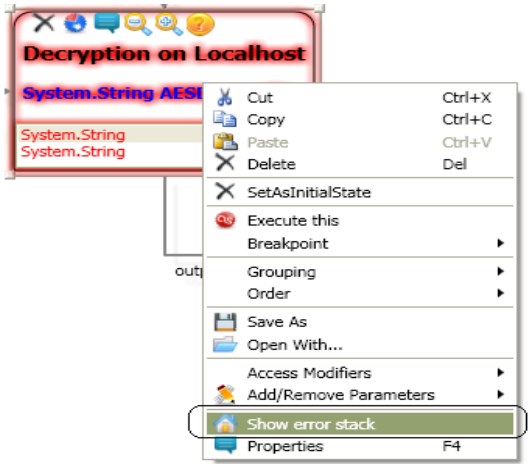

Fig. 8. Exception handling

VII. VISER Performance

VISER was tested on HP Workstation xw4200, Intel PIV (HT) 3.40GHz, 1 GB of RAM, Windows XP Service Pack 3. We underwent some more tests including single and multi-flow programs.

Flow program ImageInvert as depicted in Fig. 4, tested on individual frameworks and VISER framework, performance chart is shown in Fig. 9. We concluded that VISER takes slightly more processing time because of parsing of data flow programs and some explicit data validation processing.

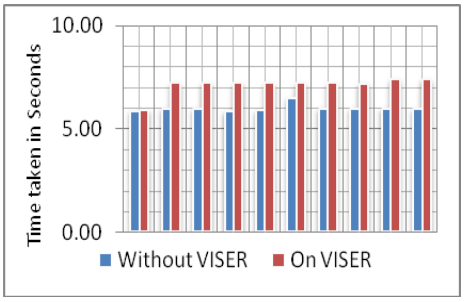

Fig. 9. ImageInvert performance test on VISER (Red color) and without VISER (blue color)

VIII. Conclusion

In this paper, our approach enriches programming environment by integrating multiple platforms into an Integrated Visual Programming Environment (IVPE) called Visual Sequencer - VISER where building a program is not a big task for those who don't know the syntax of the program. "Anyone can do programming" is the main goal of VPE. Using VPE in our approach also enriches RAD (Rapid Application Development) system for developing programs. VISER is very helpful in testing and measuring algorithms in various dimensions including cross platforms and across network systems.

Application usages of VISER:

- Design and development of visual flow graphs depicting applications.
- Test and Measurement of algorithms.
- Analysis and simulation of the programming problems.

IX. Future Work

We have implemented a platform where programs are written graphically using multiplatform components. Some of future activities are integrating a secure execution environment which can detect malicious activities performed during program execution, distributed execution of flow processes, enriching visual programming environment which will be useful to "anyone can do programming" concept.

References

[1] N. C. Shu, *Visual Programming*, Van Nostrand Reinhold, New York, 1988, ch. 1, pp. 9.

[2] M. Burnett, "Visual programming," *Encyclopedia of Electrical and Electronics Eng., J. G. Webster*, ed. New York: John Wiley & Sons, 1999.

[3] M. Burnett, "Software engineering for visual programming languages," *Handbook of Software* Engineering *and Knowledge Engineering*, vol. 2, World Scientific Publishing Company, pp. 77-92, June 2001.

[4] VisiQuest: A Problem Solving Environment for Scientific Computing and Visualization. [Online]. Available: http://www.aertia.com/docs/accusoft/VisiQuest_A problem solving environment for scientific computing and visualization.pdf
[5] Synopsis Visual Programming Tool. [Online]. Available: http://www.codemorphis.com/
[6] Ariane Data flow visual programming environment. [Online]. Available: https://clouard.users.greyc.fr/Ariane/.
[7] M. Young, D. Argiro, and S. Kubica, "Cantata: visual programming environment for the Khoros system," *ACM SIGGRAPH Computer Graphics*, vol. 29, no. 2, pp. 22-24, May 1995.
[8] Microsoft Windows Presentation Foundation. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms754130.aspx

Hari Om Prakash is presently working in the Systems Engineering Group, ADRIN, Department of Space, as Scientist/Engineer "SC" and involved in developing applications in the field of network security and secure programming language design and implementation field. He can be contacted at hariom@adrin.res.in.

R. Phani Bhushan is presently working as Section Head, Data Security Section under the Systems Engineering Group, ADRIN, Department of Space. He is leading a group involved in development activities in both host based and network based security. He can be contacted at phani@adrin.res.in.

Geeta Varadan is presently Director, ADRIN, Department of Space. She can be contacted at director@adrin.res.in.

S. Venkataraman is presently Group Director, ADRIN, Department of Space. He is heading System Engineering Group at ADRIN. He can be contacted at kalka@adrin.res.in.