Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

Visual Based Software Construction: Visual Requirement Authoring tool and Visual Program Generator[1]

Deng-Jyi Chen[2], Ming-Jyh Tsai, and Jia-chen Dai
Institute of Computer Science and Information Engineering
National Chiao-Tung University, Hsin-chu, TAIWAN 300
Email: djchen@csie.nctu.edu.tw
David TK Chen[3]
Computer Science and Information Department
FORDHAM University, NY, USA

Abstract

Most software development errors are caused by incorrect or ambiguous requirement specifications gathered during the requirement elicitation and analysis phase. For the past decades, both academic researchers and software engineers have been seeking better software requirement analysis and representation approaches for overcoming the problem mentioned above. Another equally important issue in software engineering area is how to reduce the cost of developing and maintaining the application software.

In this paper, we propose a Visual Based Software Construction Approach that uses the visual based GUI requirement scenario, created by a Visual Requirement Authoring Tool, to depict the requirement of the application software, and then generates the target application software according to the visual requirement scenario obtained in the visual requirement authoring phase. Specifically, the proposed software construction approach supports a Visual Requirement Authoring tool that allows requirement facilitators to produce GUI based requirement scenario and specifications. It also supports a Visual Program Generator that allows programmer to generate the target application system as specified in the visual requirement representation. The target application software can be generated based on the function binding features provided in the Visual Program Generator to bind each GUI component with the associated application function.

The proposed approach has been applied to construct a commercial PC camera's application software. Part of this application example will be illustrated using the proposed approach to demonstrate the feasibility and applicability of the proposed approach. The application software development cost and maintenance effort can be reduced while applying the proposed software construction approach.

Keyword: Visual requirement, Requirement Scenario, Program generator, Multimedia, Software Construction Methodology.

1This project is supported in part by the NSC, in part by Media Tech cooperation, and in part by the Bestwise International Computing Co., Taiwan.

2 All correspondence should be sent to Professor D. J. Chen at Computer Science and Information Engineering Department of National Chiao Tung University, Hsin-Chu, Taiwan.

1. Introduction

An important phase in the software life cycle focuses on eliciting the requirements from users [5]. Users and system designers generally have no common terminology or domain knowledge while establishing the requirements for the software development. As widely recognized, requirement representation form can largely affect the communication between software end users and designers. Traditional requirement representation form (textual based narration) usually leads to misunderstandings, irrelevant, or ambiguous requirements. Furthermore, voluminous textual requirement documentation arising from requirements gathering process is typically difficult to comprehend. It has been shown in [1] that Visual Requirement Representation (VRR) based on the multimedia technology has many advantages over the traditional Text-based Requirement Representation (TRR). This includes the easiness of communication between users and developers, early feedback, better expression ability, and so on. A more detailed treatment is in [1] and [15].

Multimedia technology has played an important role in modern computing because it offers more natural and user-friendly interactions with an automated system. This is particularly true for systems utilizing graphical, icon or window-based input and output. Therefore, we often utilize multimedia technology to display software GUI requirements by static graphical appearance as shown in Figure 1-1 as attached.

However, aside from the static graphic appearance, the software requirement specifications have dynamic behaviors. For example, clicking on the "Take a Snapshot" function button initializes the "Take a Snapshot" dialogue box; then the next step is to choose image size and adjust brightness and contrast. Finally, take a picture and save it. The series of actions described as above is called dynamic behavior. Obviously, this dynamic behavior cannot be displayed using the above mentioned static graphic appearance. Consequently, to assist the dynamic behavior of software requirement specifications, we often used explanatory text and flow charts. However, clear explanations of the details of each dynamic behavior results in excessive explanatory text and flow charting; it just makes it harder for the user to understand the requirement specification.

After the user's requirement specifications is produced, in the conventional way, the programmer implements the requirement specifications using an application software development system such as Borland C++ Builder or Microsoft Visual C++ to write both the corresponding functional program and GUI program for this application system. The implementation of the user's requirement specifications here is to implement the GUI lay out specification. The way makes the programmer's work heavier and longer during software development because the programmer not only has to write the functional program but also has to implement the user's requirement specifications (the GUI program). If there are any changes of software requirement specifications afterwards, the programmer has to re-implement it and rewrite the program, which increases the maintenance cost.

In this paper, we propose a Visual Based Software Construction Approach that uses the visual based GUI requirement scenario, created by a Visual Requirement Authoring Tool, to depict the requirement of the application software, and then generates the target application software according to the visual requirement scenario obtained in the visual requirement authoring phase. Specifically, the proposed software construction approach supports a Visual Requirement Authoring tool that allows requirement facilitators to produce GUI based requirement scenario and specifications. It also supports a Visual Program Generator that allows programmer to generate the target application system as specified in the visual requirement representation. The target application software can be generated based on the features provided in the Visual Program Generator that provides the binding capability between a GUI component and the associated application function.

With the proposed approach, the programmer can concentrate on application function programming and does not need to write code for GUI requirement specification. In the meantime, the relevant functional program is independent which is much easier during maintenance. Therefore, it *not only shortens the time during software development but also reduces the cost of maintenance.*

2. Related Work

There are little specific tools available for creating visual software requirements. Rational Rose, Microsoft Visio, Microsoft PowerPoint, and Macromedia Flash are some related tools that can be used as editing tools for the visual software requirement creation. We studied and compared these different tools (see table 2-1 as attached) for creating Visual software requirements based on the identified features for creating visual representation. In any comparison studies, two basic issues must be addressed which are 1) what to compare and 2) how to compare. In what to compare, we propose some common features for creating multimedia presentation for the visual requirement representation. These common features include:

(1) Providing different types of Multimedia components (or icons) gallery,

(2) Adjusting position and size of component by dragging and dropping,

(3) Supporting connection and switching of different scene,

(4) Authoring of interactive relationship and interactive action without writing any textual script programming,

(5) Integrating with the program generator.

In how to compare, we simply marked it as N, for no such feature or non-applicable, and Y, for supporting such feature.

When the correct software requirement specifications are obtained, an application development system such as Borland C++ Builder [2] can be chosen to write the programs to produce an executable application system that meets the GUI requirement specifications. Two of the application software development systems, Borland C++ Builder and Microsoft C++, are application system development programming languages that primarily use C++; this enables the user to create Windows application programs, DLLs, and so forth that can be executed on the Windows system. The third application system development tool, Microsoft Visual Basic, uses the BASIC programming language.

Those three application software development systems support many UI controls and UI controls property which helps the designer to easily layout the user's requirement specifications. Then, the designer simply double clicks the UI button to write the program related to the UI.

The disadvantages of generating an application system with any of the above mentioned application software development systems are listed below.

(1) Programs must be written to switch between user interfaces.

(2) User interface does not have multimedia components; if you want it, you must write a program.

(3) The programmer not only writes the program but also implements GUI, which makes the programmer's work heavier.

(4) If the user's GUI requirement specification is changed, the programmer would have to re-implement it.

3. Visual Based Software Construction Framework

In this section, we present the framework of the proposed Visual Based Software Construction Approach. See Figure 3-1 as attached. The framework has two major subsystems: the Visual Requirement Authoring System which is used to create visual requirement representation and the Visual Program Generator which is used to generate the target application software according to the visual requirement generated by the visual requirement authoring system.

The requirement facilitator uses the existing MRC to produce the Visual Requirements Representation via Visual Requirement Authoring System. If there is no existing MRC, the art designer uses the Component Constructor to produce a new MRC and then adds it to the MRC manager for further use in the Visual Requirement Authoring System. When the editing process is completed, a visual requirement representation is produced. We then take the produced Visual Requirement Representation as an input for the visual program generator which uses the function binding system to bind the software component and the user self-defined functional program according the GUI visual requirement Representation, and utilize an application software development system to generate the target application system.

The advantages of visual requirement over traditional textual requirement have been discussed in details in [1]. The visual requirement authoring system can be used for the purpose of representing visual GUI requirement specification and this task can be done alone by the requirement facilitator without GUI programmer's interaction. If the user's GUI requirement specification is changed, then the GUI requirement facilitator or GUI designer just uses the authoring system to create a new GUI without brothering programmer to re-write GUI code. Thus it reduces the GUI construction and modification effort.

The visual program generator, on the other hand, provides a binding system for system integrator or project leader to integrate the target application software by assembling existing MRC and the user-defined functional modules according to the GUI requirement layout (part of the visual requirement scenario). Thus, it reduces program construction and maintenance effort.

4. The Visual Requirement Authoring System and Visual program Generator

In this section, we present the system architecture of the proposed Visual based Software Construction

Framework. Specifically, the Visual Authoring System and Visual Program Generator will be elaborated in section 4.2 and 4.3 respectively.

4.1 System architecture

The system architecture of the Visual Software Construction Framework is depicted in Figure 4-1 as attached.

As shown in Figure 4-1, the system architecture includes Visual Requirement Authoring System (left hand side of Figure 4-1) and Program Generator (right hand side of Figure 4-1). The system operation procedure is depicted in Figure 4-2 as attached.

The system operation procedures are summarized below:

Step 1) the requirement facilitator selects scene background for GUI layout. In this step, the requirement facilitator can decide how many scenes for the complete visual requirement representation.

Step 2) actor setting in each scene is conducted by adding new actor, deleting or copying actors by using scene authoring system. The requirement facilitator can choose appropriate GUI button (MRC or actor) from the MRC database for the actor setting).

Step 3) scenario setting (time and space interaction among chosen actors) for actors in the scene is conducted by using scenario (or interactive) authoring system to carry out the simulation of the dynamic behavior interaction. The requirement facilitator can preview the authoring result (GUI visual requirement) using the player (the previewing system). If the result does not meet the need of end users, or needed to be changed for any reasons, then requirement facilitator simply repeats the authoring process from Step 1) to Step 3) until the visual requirement specification is satisfied.

Step 4) binding of each GUI button with the associated application function (binding setting) is performed. The project leader or system integrator will choose the appropriate application function (usually in *.dll form) from the application function library and perform the binding with the associated GUI button. If the desired application function is not available from the existing application library then programmer will need to define and implement the application function and will be compiled and integrated before the binding process can be performed.

Step 5) the GUI source code is compiled to generate the target application system by using the Borland C++ Builder programming system.

4.2 Visual Requirement Authoring System

The Visual Requirement Authoring System's (VRAS) major goal is to let requirement facilitators easily lay out GUI requirement specification for the target application system without writing any textual-based programs and to receive earlier feedback from end users. As shown in Figure 4-1, the VRAS contains three subsystems: scene authoring system, scenario authoring system, and preview system. These three subsystems are described below.

1.) Scene Authoring System The main goals are authoring the scenes, achieving interaction between scenes, and defining interaction among actors.

2.) Interaction Authoring System:
It is also known as scenario setting system which is responsible for the interactive definition, time and space relationship, and dynamic behavior among actors.

3.) Preview System:
It is used to preview the visual requirement scenario created by the authoring system. It interpreters the time and space relationship and scenario definition among actors according to the script program generated by the authoring system.

Figure 4-3 as attached illustrates the visual requirements authoring process. We initially create a project for the system. Next, one or more scenes can be created for the project. We can reuse a scene if a reusable scene pattern is found. Otherwise an empty requirement framework is created. For every scene, actors (or MRC) are selected from MRC database and placed on the scene. This work is completed by the scene authoring system. For all selected actors in the scene, we use the scenario authoring system to define their actions and describe relationships to perform a scenario. A visual requirement scenario is then produced. We can use the preview system to preview it, and examine whether this scenario meets the user requirements.

4.3 Program Generator

The program generator's major goal is to reduce the cost of the application system development and maintenance. The program generator uses the confirmed visual requirement representation as input then uses the function binding system to bind each GUI component with the corresponding application software function, external executable file, and user's defined program. The final step is through the GUI program generation system to generate a target application system.

The program generator has four subsystems: function binding system, user interface source code generator, play system, and the compile environment. A detailed description is in the following subsections.

4.3.1 Function Binding System The major goal of the function binding system is to be able to *directly* embed the software component into the visual requirement representation (or user's requirement specifications). The software component is designed by the programmer, after which, through the embedding of the function binding system into the scene and formulation of the interactive relationship, the interactive description information is written into the script files

4.3.2 UI Source Code Generator The UI Source code Generator needs to use the information of the user-defined program existed in the function binding information. Consequently, after using the function binding system, it is necessary to obtain the script files, so that the UI Source code Generator could be operated.

4.3.3 The UI Interpreting System Like the preview system in the visual authoring system, the UI interpreting system is used to present the generated GUI visual requirement specification after the binding process.

From the Figure 4-1, we can see that the UI Interpreting system includes the following three subsystems.

1.) **UI Source Code Generation** This part is generated from UI source code generator.
2.) **Software Component Management**

System A software component management system allows the software component to assist the components in dispatching messages when it is loaded into the system. This achieves the goal of smooth communication between components.

3.) **UI Container** The UI container goal is to play the already authored Scene, allowing multimedia playing, and to interact with the user during the play.

4.3.4 Compile Environment The defined programs by the user can be added to the UI interpreting system. After this, we still need a compilation environment to compile these components, allowing us to obtain the final application system execution files. We take Borland C++ Builder to compile these components.

5. Application Example - Create a Digital Camera Application System

The application software we constructed for a commercial PC camera is based on Visual Based Software Construction Approach. This commercial product is distributed by Bestwise International Computing Inc.

6. Conclusion

In this research, we first show that user requirements can be visualized. We utilized a visual requirement authoring system to convert the user's requirement from a textual documentation to a visual requirement representation. Such a visual requirement representation has the following benefits:

1) Replace voluminous textual documentations.
2) Provide a more natural means of communication between user and system designer.
3) Elicit an early feedback from user, more expressive in describing user's demands.
4) Lay out requirement representation without writing textual program.

Second, we take directly the confirmed visual requirement presentation as input for the program generator to perform the function binding and to generate the target application system as specified. This approach not only can shorten the time during software development but also can reduce the cost of maintenance. The proposed visual based software construction approaches had been applied to produce a commercial product in PC camera's video capturing application service and illustrate its feasibility and applicability.

7. Reference

[1]Deng-Jyi Chen, Wu-Chi Chen, Krishna M. Kavi, "Visual requirement representation", the Journal of System and Software 61 (2002), pp129-143.

[2]Borland C++ Builder, Available: http://www.Borland.com.

[3] Microsoft Visual C++ and BASIC, Available: http://www.Microsoft.com.

[4]Macromedia Flash, Available: http://www.Macromedia.com.

[5]Ian Sommerville, Software Engineering, AW, pearson.

[6] Jyi-Sheng Tyan, "The Design and Implementation of a Script Language and Playback for Scenario-Based Electronic Book," Master Thesis of NCTU Taiwan, 1999.

[7] Chwan-Hung Wang, "Visual-Based User Interface Requirement representation," Master Thesis of NCTU, Taiwan, 2002.

[8] Chorng-Shiuh Koong, "The Design and Implementation of a Script Language and Playback System for Electronic Story Book," Master Thesis of NCTU, Taiwan, 1995.

[9] Shih-Fang Chuang, "The Design and Implementation of a Visual Language for Scenario Based Electronic Book," Master Thesis of NCTU, Taiwan, 1999.

[10] Deng-Jyi Chen and S. K. Huang, "Interface of Reusable Software Components", the Journal of Object-Oriented Programming, Vol. 5, No. 8, pp 42-53, January 1993.

[11] Ohnishi A. "A Visual Software Requirements Definition Method", IEEE Proceedings: The first International Conference on Requirements Engineering, April 18-22, 1994, Colorado Springs, Colorado, pp31-40

[12] Ohnishi, A.; Tokuda, N. "Visual Software Requirements Definition E", computer software and application conference, 1997.

[13]Ohnishi, A. "Audio-visual Software Requirements Specification", Multimedia software engineering, 1998.

[14] Jia-Chen Dai, "Visual Based User Interface Generator", Master Thesis of NCTU, Taiwan, 2002.

[15] Wu-Chi Chen, "A Visual and Reuse-based Paradigm for Software Construction," Ph.D. Dissertation, NCTU, Taiwan, 1998.

[16] Chorng-Shiuh Koong, "A component-based Visual Scenario Construction Environment for Non-Programming Users to create Interactive Electronic Books," Ph.D. Dissertation, NCTU, Taiwan, 2000.

Attachment [Figure]

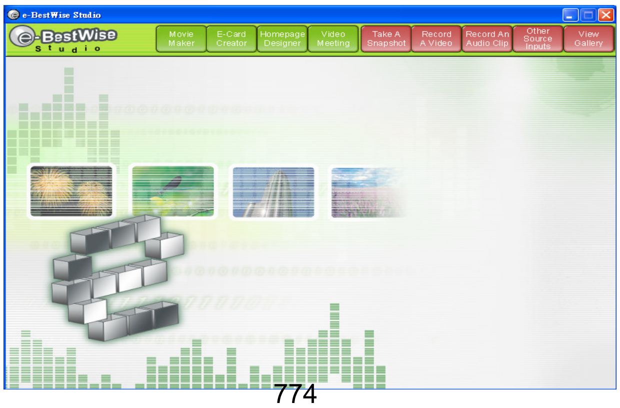

Figure 1-1 Static graphic appearance of application system

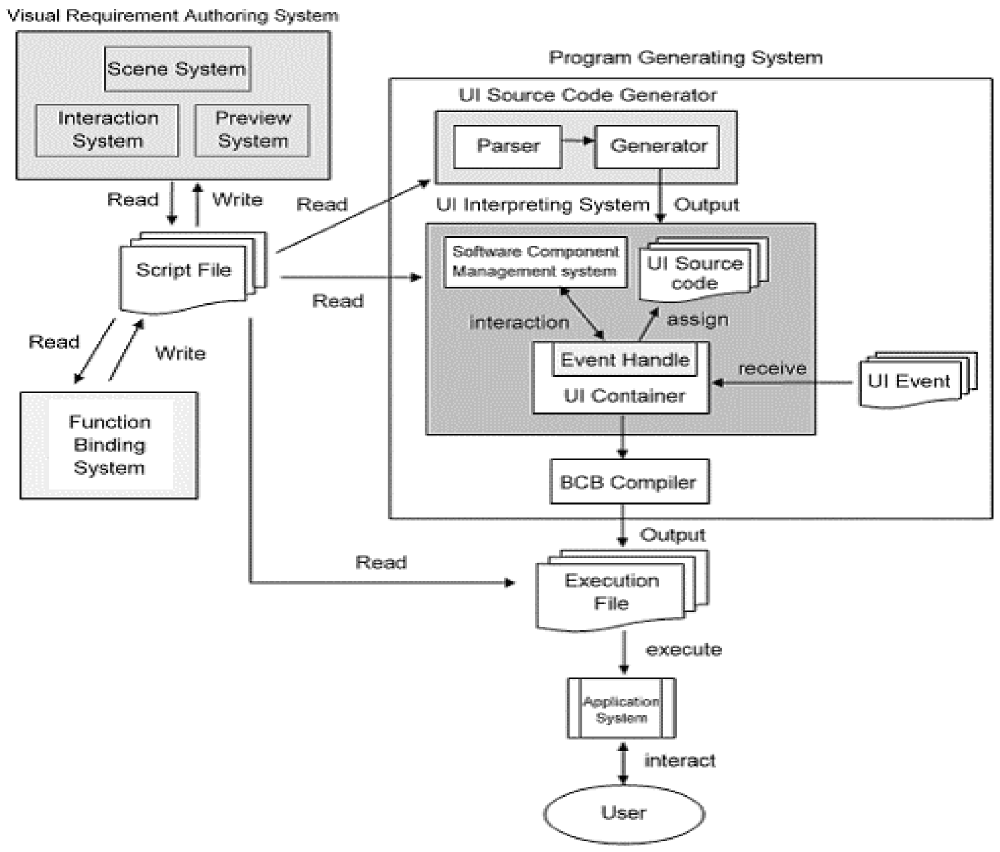

Figure 4-1 System Architecture of Visual based Software Construction.

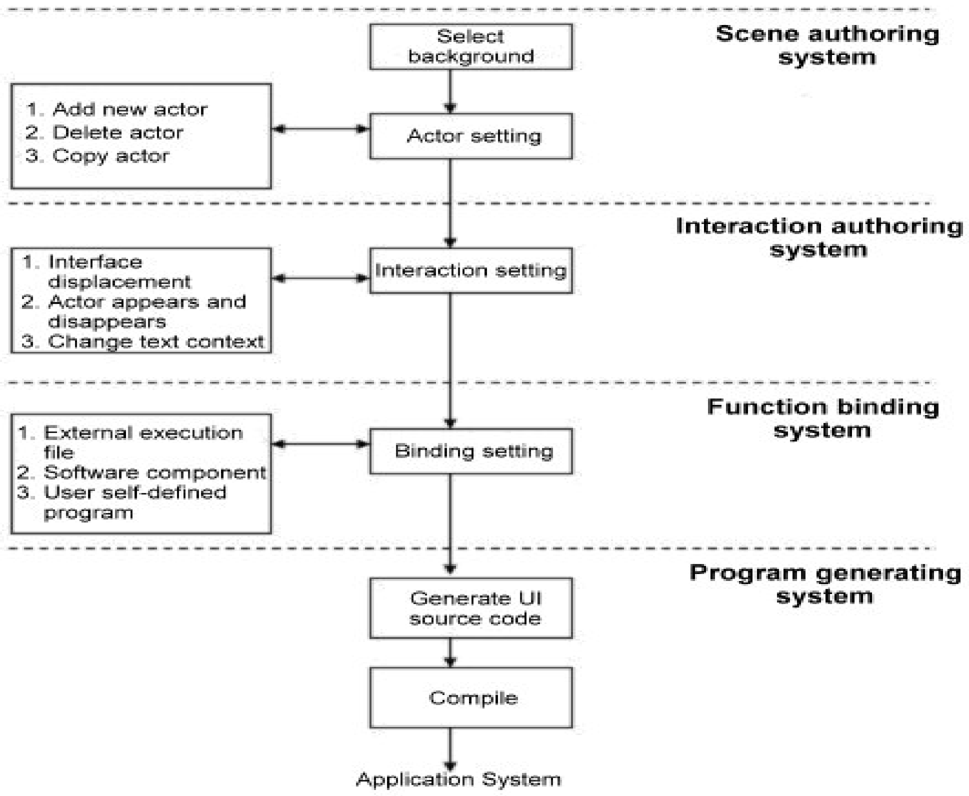

Figure 4-2 System Operation Procedures

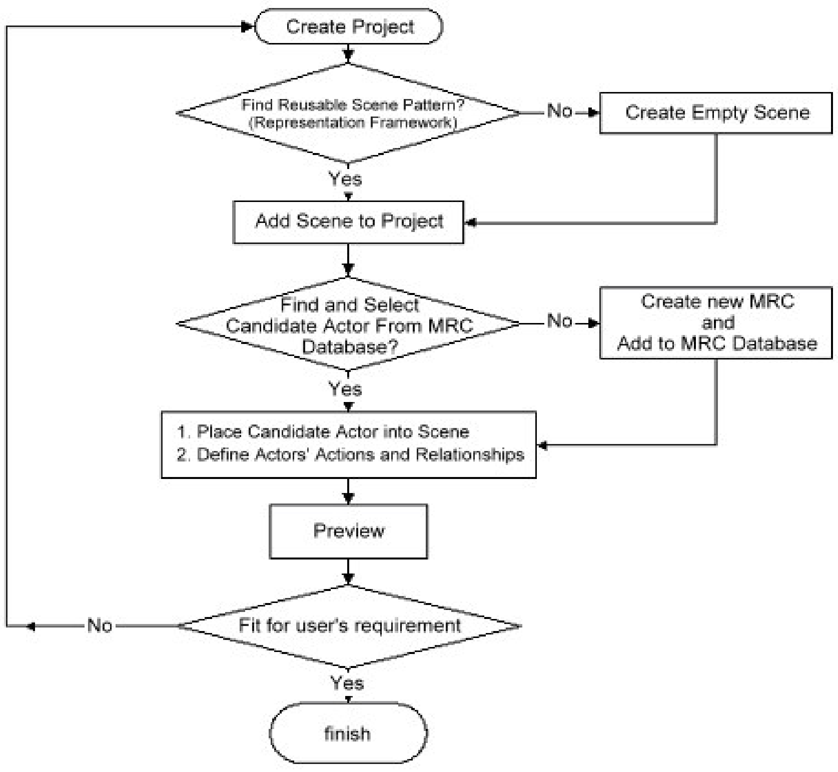
Create Project
Find Reusable Scene Pattern? (Representation Framework)
No
Create Empty Scene
Yes
Add Scene to Project
Find and Select Candidate Actor From MRC Database?
No
Create new MRC and Add to MRC Database
Yes
1. Place Candidate Actor into Scene
2. Define Actors' Actions and Relationships
Preview
No
Fit for user's requirement
Yes
finish

Figure 4-3 Visual Requirement Authoring process [1].

Attachment [Table]

Table 2-1 Comparison editing tools

Software tool / *What to compare*	Rational Rose	Microsoft Visio[3]	Microsoft PowerPoint [3]	Macromedia Flash [4]
Providing different types of Multimedia components (or icons) gallery	N	N	Y	Y
Adjusting position and size of component by dragging and dropping	Y	Y	Y	Y
Supporting connection and switch of scene	N	N	Y	Y
Authoring of interactive relationship and interactive action without writing any textual script programming	N	N	N	N
Integrating with the program generator	N	N	N	N