1998 International Computer Symposium
Workshop on Software Engineering and Database Systems
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

VISUAL REQUIREMENT REPRESENTATION

Chorng-Shiuh Koong, Wu-Chi Chen, Chung-Chien Hwang, and Deng-Jyi Chen[1]
Institute of Computer Science and Information Engineering
National Chiao Tung University, Hsin-Chu, Taiwan, R.O.C
E-Mail: {djchen,csko,wjchen,cchwang}@csie.nctu.edu.tw Fax:(03)5724176
Shih-Kun Huang
Institute of Information Science, Academia Sinica,Taipei, Taiwan
E-Mail: skhuang@iis.sinica.edu.tw

ABSTRACT

Multimedia technology has played an important role in modern computing due to its friendly user interaction as well as its naturally fitting on real world modeling. Reusable software components are the basic building blocks for software construction based on the software reuse practice.We extended reusable software components to incorporate with multimedia. In other words, the reusable software components include not only code and documents, but also voice narration, animation sequences and message mechanisms. We called such software components as *Multimedia Reusable Components (MRCs)*. Based on these developed MRCs, a novel software requirement representation paradigm is introduced. With this novel representation paradigm for requirement representation, one can view the software requirement representation as sequences of animation instead of reading voluminous software requirements. Such a novel software requirement representation paradigm will provide users a visual effect and to have an earlier feedback from users. Also, it will provide an easy and natural form of the communication between designers and users. In this paper we propose a visual software construction paradigm based on multimedia reusable components and implement a visual requirement authoring tool. Also, an assessment of the proposed approach is discussed.

Keyword: multimedia, requirement reuse, visual requirement

1. Introduction

The system engineering and analysis phases in the software life cycle largely focus on eliciting the requirements from users [4]. Gathering requirements is an extremely difficult task. In performing this task, four requirement elicitation problems must be addressed: communication, irrelevancy, incorrectness, and inconsistency. Users and engineers generally have no consensus formal language and the same domain knowledge while establishing the requirements for software development, subsequently leading to the above problems. As widely recognized, ineffective communication frequently incurs misunderstanding and irrelevant or incorrect requirements. Also, different users offering varied perspectives lead to inconsistent requirements. Furthermore, voluminous textual requirement documentation arising from requirements gathering process is typically difficult to comprehend owing to that text is not a natural form for users to comprehend the requirements of a software system. To alleviate some of these problems, a novel software construction paradigm is developed.

Multimedia technology plays a prominent role in modern application software owing to its friendly user interaction as well as its natural fitting on real world modeling. Multimedia data include components such as text, image, animation, video, and voice, all of which can be manipulated by a computer. Herein, we incorporate reusable software components with multimedia. Restated, in addition to containing a code and documents, reusable software components also include voice narration and possible animation sequences. We called such software components as Multimedia Reusable Components (MRC) [20]. Based on these developed MRCs, we propose an novel software requirement representation paradigm. This novel representation paradigm for requirement specification allows a user to view the software requirement specification as sequences of animation instead of reading or studying voluminous software requirements. Such an novel software requirement representation paradigm provides the user with a visualized effect, allowing him/her to obtain an earlier feedback from users. This paradigm also facilitates a natural form of communication between designers and users.

In this paper, we concentrate mainly on requirement acquisition, particularly the communication ability, requirement representation, presentation, organization, and requirement reuse based on multimedia reusable components. A visual requirement representation model is also proposed, along with an authoring tool based on that model implemented as well.

2. Visual modeling

2.1 Visual modeling

James Rumbaugh stated "Modeling captures essential parts of the system" [3]. Modeling technique is extensively employed during software requirement analysis and design. A visual modeling uses standard multimedia notations to model a system's requirements and software programs. Visual modeling can capture a system process from the user's perspective. Such an approach is a natural

[1] All correspondences should be sent to Prof. D. J. Chen, Computer Science and Information Enginggring Dept., National Chiao Tung University, Hsin-Chu, Taiwan.

communication means and can be used to capture system objects and logic from users. Visual modeling can represent levels of system abstraction, thereby capable of managing the system complexity and defining the software architecture. In a visual modeling environment, multimedia notations are repeatedly used, thus encouraging software reuse.

2.2 Equivalence of requirement representation model

In the high level representation abstraction, a requirement can be represented by different kinds of representation models, including a textual model or visual model. Regardless of what representation models are used, the input requirement is transformed into requirement specification (or called determined requirements), as depicted in Fig. 1. Assume that we define

R: the meaning of a requirement
S: requirement specification
T: textual model
M: visual model

Figure 1. Representation model

The representation model can be a textual model or a visual one. Then, we have

$R(T) \equiv R(M)$, for the same requirement R.

Above equation displays the meaning of a requirement, as represented by a circumstance in which the text model is ***conceptual equivalent*** to the meaning of a requirement represented by visual model. Restated, the different representation models represent the same requirement scenario. In the requirement construction phase, requirements can be represented by a textual model or by a visual one. Regardless of which representation model was used, the implication of the requirement must be the same. Please take notice that the *conceptual equivalent* is defined based on the high level abstraction. Visual modeling may not be used to represent a detail description well.

Textual model and visual one have the same power to represent a requirement meaning. Experiment involving the using of visual requirement representation vs. textual requirement representation is performed in [20]. Experimental results indicate that visual representation is effective for users' understanding and communication. Based on the experimental results, we propose a visual paradigm for software construction.

3. Visual requirements representation model

3.1 Using multimedia reusable components

MRCs are encapsulated with object-oriented paradigm and designed in a standardized format. An MRC, which consists of multimedia data, operations, and message mechanism, can be viewed as an alive (active) object while used in a presentation system. Combining several MRCs allows us to produce a multimedia film. Assume that each MRC represents a requirement segment in a multimedia form. Then, each scene subsequently produced represents a requirement scenarios. Meanwhile, the whole film represents the complete requirement under consideration.

3.2 Scenario

Scenario, occasionally referred to as *use case*, is a relevant strategy towards understanding the interface between the environment and the system. A software system and its environment may be extremely complex and vary. Behaviors and subsystems relationship of a system is difficult to clarify by text description. Under this circumstance, system requirements are barely elicited from users, possibly leading to a misunderstanding between users and developers. Scenario is an evolving description of situations in the environment. The use of scenario can easily elicit and specify software behavior, and can clarify the interrelation between functional and non-functional requirements. A repeating scenario can be reused as a requirement pattern. Based on the visual representation capabilities, we propose visual scenario requirements by using multimedia and visual effect to represent a scenario. In this manner, a visual software requirement is executed with a scenario of multimedia presentation.

3.3 The proposed model

The visual requirement representation model, as shown in Figure 2, includes six parts.

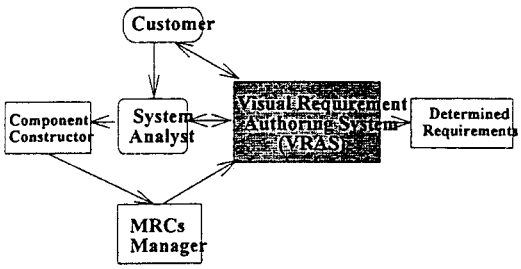


Figure 2. Visual requirements representation model

Visual requirements authoring system (VRAS) can be considered as a multimedia authoring tool, consisting of a script language, graphic user interface (GUI) for interconnecting MRCs, and a playing system. It is used to make a visual requirement and presentation for a customer and system analyst.

Customer: A user who provides the need for a system analyst and can also join the making of requirement film by using the VRAS.

System Analyst: The system analyst can use VRAS to produce a customer's requirement. An analyst must also determine what MRC is necessary for the customer's requirement.

Component constructor: While the VRAS can not identify the desired MRC from the MRCs manager, a new MRC should be designed and add to the manager system. The component constructor helps the designer perform this task. The constructor includes a multimedia editor and a relatively simple programming environment to encapsulate multimedia data and related code components. The constructed MRC can then be added to the manager system. Multimedia artist and software engineer are involved in MRC construction phase.

MRCs' manager: A database management system manages MRCs and provides interface for MRCs adding, deleting, and retrieving. The manager organizes and stores MRCs based on component standardization information and all related files.

Determined requirements generator: Once the visual requirement is produced, the generator can transform the requirement into a formal form. This is referred to as determined requirements. Based on the determined requirements, software design and coding phase can be performed. Owing to limited scope of this research, the generator issue is not addressed herein.

3.4 Term definitions of visual-base requirement representation

Four basic authoring elements are available to create a visual requirement representation: scenes, actors, relationships, and window-based operations.

Actors: An actor, i.e. an object that can send or receive messages, represents a requirement segment and is a minimal reusable requirement element. Based on our representation model, an MRC is considered as an actor.

Scenes: A scene is an actor's container, where an actor can be used to represent a session of requirement. A scene can therefore be used to accumulate several requirement scenarios. A requirement scenario consists of one or more MRCs. A scene also defines a requirement framework. In a scenario, an MRC can be substituted with other MRC. The ability to abstract the scenario pattern allows to reuse the scene as requirement framework.

Relationships: It describes the intra-actor relationship and scene-actor relationship. The object relationships denote the presentation sequence between actors, e.g. sequential or parallel presentation. The space relationships describe MRC's size, position, movement path, depth, and rotation while the MRC is placed in a scene. The time relationships describe the timing conditions during presentation. For instance, the presentation speed, loops, or delays. MRCs are organized by relationship features to perform a requirement scenario.

Requirement project: A requirement project organizes scenes to present a system requirement. The project can be viewed as a multimedia program and can interact with actors in each scene.

Using these four elements allows the user to author his or her visual requirement scenario based on MRCs in a visual manner.

3.5 Script language

A component connection language is deemed necessary to connect MRCs, define presentation behavior, and represent MRCs relationships. Such a language is defined and implemented in a visual form such that users can easily use the proposed language for creating a visual requirement. The visual requirement which is created in visual form is stored by a script language.

While actors can only execute simple tasks, a script controls the MRC on a very high level. The script fulfills a task similar to the process manager in a multitasking operating system. In addition, the script creates and deletes actors and triggers specific events. The script and the actors execute concurrently.

A presentation script has two components: declarations and actions. The declaration part deals with objects where they are stored. The action part concerns itself with the way these objects are presented. An action can be either primitive or composed. A primitive action is defined as a single media presentation. The language does not specify how primitive actions are executed since such actions are executed in MRC.

The script language is used to describe the manner in which actions are executed, such as in parallel or in sequence and their duration. The relationships that are dealt in our script language include:

1). Actor space relationship: presentation path, icon size, position, depth, and rotate
2). Actor time relationship: sequential, parallel-or, parallel-and, delay,
3). Presentation properties: speed, loop, appear, hide, start, stop,
4). Actor creation and deletion,
5). Multimedia hyperlink between scenes: jump, conditional branch, and
6). Concrete: define a scenario pattern.

Table 1 summarizes the presentation actions and script language features.

4. Visual requirements authoring

4.1 Representation structure

Requirement scenarios are complex for a large software system. A structural organization for such a complex requirement scenarios should be provided. In our visual requirement representation model, at least one scene represents the visual requirements. A scene includes actors and relationships to represent requirement scenarios. In addition, organizing the requirement scenarios involves structuring the scenes.

Table 1. Presentation actions and script

Presentation Requirement	Actions	Requirements Presentation Meaning	Script language features
Scene background and display effect	Select display effect Select background Save background Draw on background	Beautify the look of a scene	Background (ID, file name) Effect (ID)
Actor arrangement	Arrange actors' position and animation movement on visual editor. Set actors' attributes	Basic requirement components	ActorName(ID, file name) Position (x, y) Size (width, height) Speed (frames) Depth (level) Rotate (degree) Path ((x1, y1), (x2, y2),...)
Sequential presentation	While an anchor actor starts, a list of actors will follow its presentation one by one. A finished message returned when the last actor in the list finish its presentation.	Sequential scenario Represent serial message flow, control flow, or processes	Sequential (anchor, actor list)
Parallel-or presentation	While an anchor actor starts, a list of actors also starts their presentation at the same time. A finished message returned when any one of actors in the list finishes its presentation.	Parallel scenario Concurrent processes Control transfer occurs at any of the current processes terminated.	ParallelOR (anchor, actor list)
Parallel-and presentation	While an anchor actor starts, a list of actors also starts at the same time. A finished message returned when all actors in the list all finish their presentation.	Parallel scenario Concurrent processes Control transfer occurs at all of the current processes terminated.	ParallelAND (anchor, actor list)
Delay presentation	Delay a period of time while receiving the start presentation message. It is used with sequential and parallel presentation.	Represent a real-time constraint.	Delay (actorID, second)
Loop presentation	Repeatedly present the actor several times.	A process is continuous running or repeats several times.	Loop (actorID, times)
Appear/Hide	Show an actor on a scene. Hide an actor from a scene.	Represent creation or deletion of an object while process running.	Appear (actorID) Hide (actorID)
Scene branch	Close current scene and open another scene.	Scenario connection. Link to subsystem. To view a scenario in detail.	LinkScene (sceneID)
Concrete	Define a scenario pattern.	To represent a scenario (framework) for which actors in the scenario can be substituted.	Concrete Scene Begin ... Scene End
Start/Stop	Start/Stop presenting a scene	Start/Stop visual requirements scenarios presentation.	Start Start (sceneID) Stop

Scenes can generally be structured as list, tree, and graph, as illustrated in Fig. 3. Hyper-link is used to connect among scenes. These three basic scene structures can be arbitrarily combined to represent a complex requirement scenario. A specific scene organization can be abstracted to a reusable requirement representation pattern or representation framework.

4.2 Visual requirements authoring

A visual requirement is created by visual operations and described by the script language internally.

How is a visual requirement film produced? Figure 4 illustrates the visual requirements authoring process. We initially create a project for the system. Next, a scene or several scenes can be created at a time for the project. If a reusable scene pattern (application framework) is located, then we can reuse the scene. Otherwise an empty requirement framework is created. For every scene, actors (MRCs) are selected from MRC database and placed into the scene. For all selected actors in the scene, we define their actions and describe relationships to perform a scenario. Next, a visual requirement scenario is produced. The scenario can be previewed or executed to observe whether or not the scenario adheres to the user's requirement or not. Finally, a decision is made to go around the authoring process or stop authoring.

4.3 Visual requirements reuse

The basic requirement reusable component is MRCs. A scenario is a requirement segment. A specific scenario pattern can be abstracted as a reusable requirement scenario pattern because it provides a relatively easy means of replacing or substituting its MRCs to produce another requirement scenario. A scene represents one or more scenarios and can treat the scene as frameworks and can be reused. A project describes an application framework. In addition, an application framework represents a specific

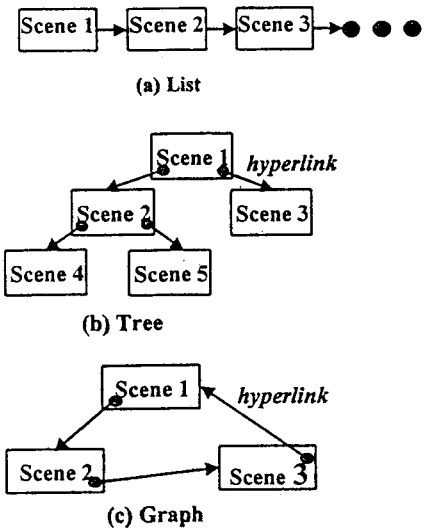


Figure 3. Representation structure

scenario for an application, entities in the framework can be substituted by other related entities. Moreover, reuse at the requirement level can significantly enhance the software development.

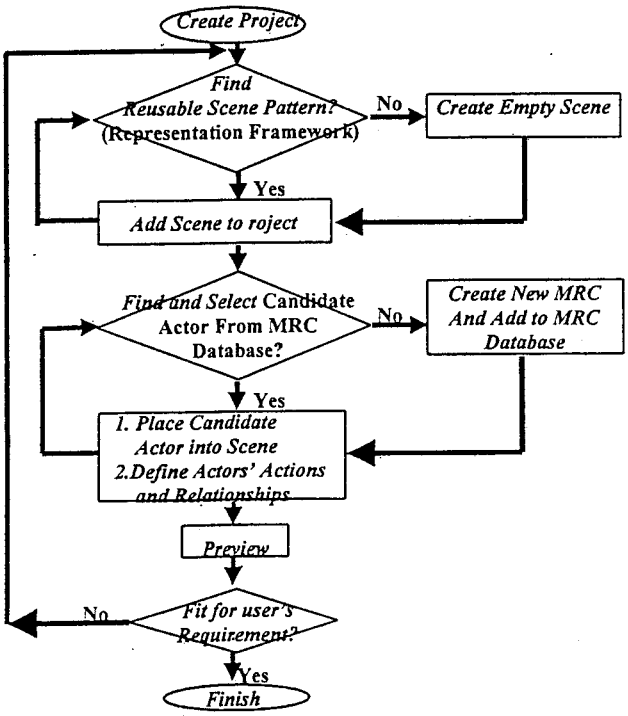


Figure 4. Visual requirements authoring process

5. Implementation

5.1 System structure

The major implementation issue in the visual requirement model is the implementation of a Visual Requirement Authoring Tool. The authoring tool allows users to pick up various MRCs, stored in the MRCs Manager System , subsequently turning them into a film (visual representation) that can be played to customers.

The MRCs must be designed in a standard format. In addition, the MRC designers must generally seek assistance from artists on drawing some meaningful and simple motion pictures to accurately depict the basic meaning of an event (a requirement scenario). The authoring tool must provide various functions that allow analysts to change an MRC's attributes, to make an animation sequence for an event in an MRC, as well as assemble several MRCs together as a scenario-based requirement. These scenario-based requirements are then combined as a feature presentation (film) and be played, by a playback system, to users for evaluating whether if the requirement satisfies his or her need. Figure 5 depicts the system architecture .

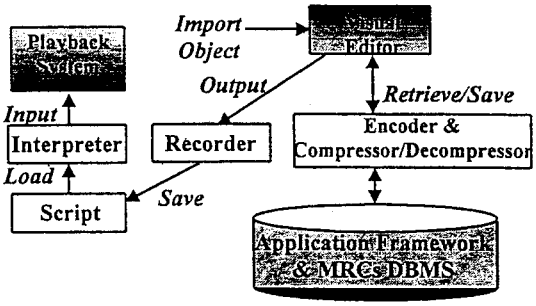


Figure 5. System structure

5.2 Example - A part of the banking transaction system scenario

Herein, a visual requirement authoring tool is implemented and used to create a visual requirement. Thus, the difference between textual representation and visual representation can be compared and evaluated.

Textual Representation of the Partial Requirement of the Banking Transaction

"Customer asks an electronic cash from the bank where the customer deposit his money. The customer's bank should confirm the customer's requirements and update its account balance. After verifying the correctness, the customer's bank transfers the customer identification information and the correct amount of cash to the electronic mint. The electronic mint confirms the identification information from the bank and issues an equivalent amount of electronic cash to the customer. The electronic mint also updates its account data. All the messages are exchanged through Internet and should be encoded for security reason."

After reading the description, the textual representation of the scenario is listed below.

1). Customer sends a "Request_For_E-Cash" message to the bank.
2). Bank requests the customer's personal data for authorization.
3). Customer sends his account information to the bank.
4). Bank update the customer's account information.
5). Bank sends "Request_For_E-Cash" message to Electronic Mint.
6). Electronic Mint updates the Bank's account information.
7). Electronic Mint sends "E-Cash" to the customer.
Figure 6 depicts the event diagram of the textual scenario.

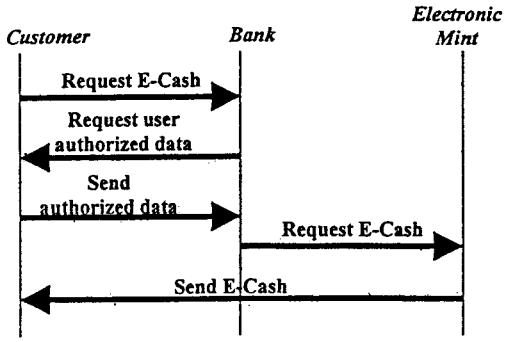


Figure 6. Event diagram of obtaining scenario

Visual representation of the partial requirement of banking transaction

For the same requirement scenario stated in the previous segment is represented using the visual requirement representation. Figure7 depicts the system behavior and the event diagram the same requirement scenario in animation form. Notably, the circled number in Fig. 7 denote the presentation order. All objects are presented in animation with multimedia effect.

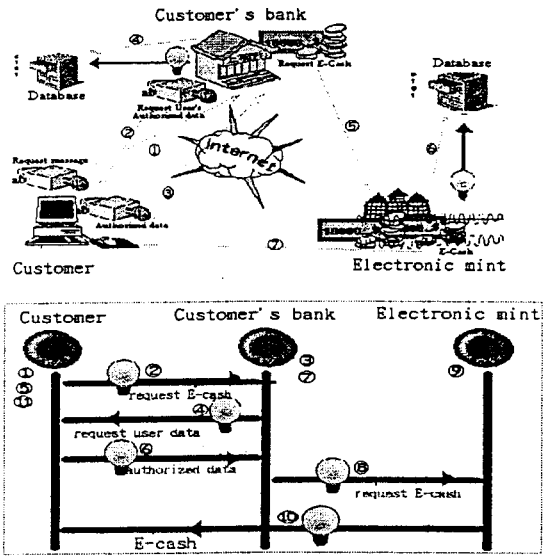


Figure 7. Visualized event diagram of the requirement scenario

5.3 Evaluation of the proposed visual requirement tool

To evaluate the proposed VRAT, an experiment is performed for comparing the tool with other authoring tools. This experiment attempts the following:
1. To understand the difficulties involved in creating the visual requirements representation.
2. To evaluate the merits and limitations of the visual requirements authoring tool.

Experimental design

Twenty four graduate students who participated in a graduate course in Object-oriented Computing are divided into two groups randomly, group A and group B, with twelve persons per group.They were asked to represent a same problem(Electronic Cash System requirements analysis). We collected the time usage in both approaches and performed a questionnaire after they have done the requirement representation. Processes of the experiment include four phases:

Phase 1: All subjects were requested to represent the system by using the textual model and the time usage was recorded.
Phase 2: Both group A used VRAT and group B selected their familiar multimedia authoring tool was asked to create requirement representation. Time usage was recorded.
Phase 3: Perform the questionnaire.
Phase 4: Accumulate data and analysis.

Variables

The experiment manipulates four independent variables:
1). time usage;
2). degree of problem expression capability;
3). easiness of communication between users and developers;
The degree is divided into five ranks: 0%, 25%, 50%, 75% and 100%.

Data and analysis

The time ratio is defined as

$$\text{Time Ratio} \equiv \frac{\text{Time-usage in textual representation}}{\text{Time-usage in visual representation}}$$

Figure 9 obviously indicates that using the proposed VRAT spend less time for creating the same requirement. However, the time deemed necessary to create a meaningful MRC for matching requirement under consideration is not taken into account. Thus, this finding only indicates that with sufficient MRCs in database, the visual requirement authoring becomes a relatively easy task.

Representation difficulties

Figure 10 indicate that the proposed authoring tool has relatively less representation difficulty.

Problem expression capability

Figure 11 indicates that the visual requirement representation has a higher ability than textual representation.

The easiness of communication between users and developers

Figure 12 indicates that visual representation and the proposed VRAT are more acceptable.

Early feedback

Figure 13 indicates that the score for the visual group is higher than textual group.

From the experiment and the rational judgment, the following results are obtained:

1). Visual requirement representation provides an earlier feedback from the users.
2). Visual requirement representation is more natural in corresponding to the communication behavior between users and developers.
3). Visual requirement representation has more freedom in expressing what the user wants (user's requirement) owing to the assistance from audio and video effects.
4). Requirement authoring tool with the multimedia features helps the users to develop visual requirement and presentation.
5). The design of visual representation tool such as VRAT is encouraged.

This experiment provides some quantitative data which do not require repeating again to verify the results obtained from this experiment.

6. Conclusion

Requirement representation can be visualized. The proposed MRCs makes the visual requirement representation possible. By using visual requirements, the requirements can be viewed as an animation sequence instead of reading voluminous requirement documents. Such an novel software requirement representation paradigm provides a more natural means of communication between user and developer, provides earlier feedback from users, more freedom in expressing user's requirement. In addition, the requirement authoring tool with the multimedia features helps users to develop visual requirement and presentation.

References

[1] C. L. Li, An Object-Based Icon Programming Methodology, Master Thesis of N.C.T.U Taiwan, June 1992

[2] M. Lenz, H. A. Schmid and P. W. Wolf, "Software Reuse through Building Blocks", IEEE Software, July 1987, pp. 34-42

[3] James Rumbaugh, et al, Object-Oriented Modeling and Design, Prentice-Hall, 1991

[4] Roger S. Pressman, Software Engineering-A practitioner's approach, 3rd edition, McGraw-Hill, 1992

[5] Peter Freeman, "A perspective on Reusability", The Computer Society of the IEEE, 1987, pp2-8.

[6] Grady Booch, Object-Oriented Design with Applications, Benjamin/Commings Publishing Company, Inc., 1991.

[7] Mitchell D. Lubbars, "Wide-Spectrum Support for software Reusability", Proceedings of the Workshop on Software Reusability and Maintainability, October 1987.

[8] P. Coad and E. Yourdon, Object-oriented Analysis, Prentice-Hall, 1990.

[9] Joseph A. Goguen, "Formality and Informality in Requirements Engineering", Proceedings of the 2nd International Conference on Requirements Engineering, April 15-18, 1996, Colorado Springs, Colorado, pp.102-108

[10] Marina Jirotka, Christian Heath and Paul Luff, "Ethnography by Video for Requirements Capture", Proceedings of the 2nd International Symposium on Requirements Engineering, April 27-29, 1995, York, England, pp.190-191

[11] David P. Wood, Michael G. Christel, and Scott M. Stevens, "A Multimedia Approach to Requirements Capteru and Modeling", IEEE Proceedings: The first International Conference on Requirements Engineering, April 18-22, 1994, Colorado Springs, Colorado, pp.53-56

[12] Atsushi OHNISHI, "A Visual Software Requirements Definition Method", IEEE Proceedings: The first International Conference on Requirements Engineering, April 18-22, 1994, Colorado Springs, Colorado, pp.194-201

[13] Kenji Takahashi, Colin Potts, Vinay Kumar, Kenji Ota, and Jeffrey D. Smith, "Hypermedia Support for Collaboration in Requirements Analysis", Proceedings of the 2nd International Conference on Requirements Engineering, April 15-18, 1996, Colorado Springs, Colorado, pp.31-40

[14] N.A.M. Maien, P. Mistry, and A. G. Sutcliffe, "How People Categorise Requirements for Reuse: a Natural Approach", Proceedings of the 2nd International Symposium on Requirements Engineering, April 27-29, 1995, York, England, pp.148-155

[15] W. Lam, J. A. McDermid, and A. J. Vickers, "Ten Steps Towards Systematic Requirements Reuse", Proceedings of the 3rd International Symposium on Requirements Engineering, July 6-10, 1997, Annapolis, Maryland, USA, pp.6-15

[16] Philippe Massonet and Axel van Lamsweerde, "Analogical Reuse of Requirements Frameworks", Proceedings of the 3rd International Symposium on Requirements Engineering, July 6-10, 1997,

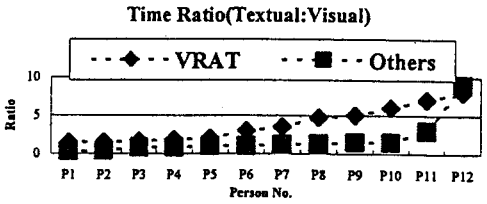
Annapolis, Maryland, USA, pp.26-37

[17] Breen David E., Getto Phillip H., Apocada Anthony A., Schmidt Daniel G., Sarachan Brion D., "The Clockworks: An Object-Oriented Computer Animation System", Proceedings of Eurographics, pp. 275-282, 1987

[18] Barry Keepence, Mike Mannion, Stephen Smith, "SMARTRe Requirements: Writing reusable requirements", Proceedings of the 2nd International Symposium on Requirements Engineering, April 27-29, 1995, York, England, pp.27-35

[19] Chorng-Shiuh Koong, "The Design and Implementation of a Script Language and Playback System for Electronic Dtory Book, Master Thesis of N.C.T.U. Taiwan, 1995.

[20] W.C. Chen, "A Visual and Reuse-based Paradigm for Software Construction," a Ph.D. dissertation, Computer Science and Information Engineering Dept., National Chiao Tung University, Taiwan. June 1998.



VRAT:average=940
Other: average=1761

Time usage Figure 9. Time ratio and time usage

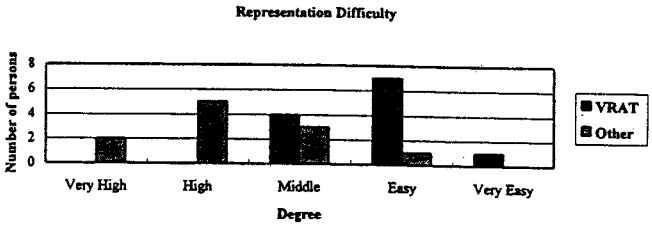


Figure 10. Representation difficulties

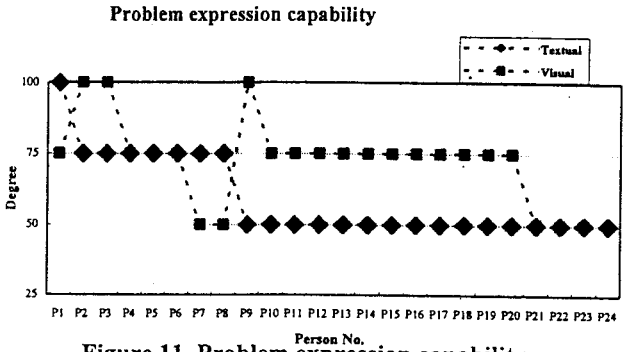


Figure 11. Problem expression capability

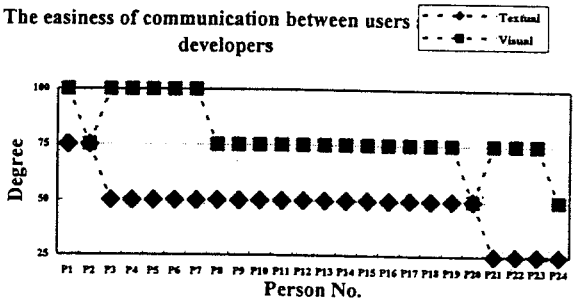


Figure 12. The easiness of communication between users and developers

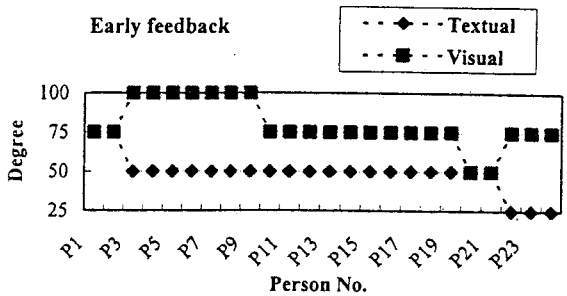


Figure 13. Early feedback