ISSN No. 0976-5697

Volume 1, No. 4, Nov-Dec 2010

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Re-Engineering for GUI Conversion from Markup Language to Programming Language

Dr. E. Kirubakaran
Sr. Deputy General Manager, OutSourcing,
B.H.E.L., Trichy-14, India
e_kiru@yahoo.com

S Manimekalai*
Research Scholar, MTWU, Kodaikannal.
Lecturer in Computer Science,
Government Arts College, Trichy-22, India
mega_somu@yahoo.com

Abstract: Knowledge differs from person to person due to their way of thinking. Now-a-days, all the things are developed based on technology. A technology which is easy for a person may seem to be difficult for another person. In software development architecture, there involves a group of persons to develop software, and they work in different phases. Each phase is interlinked with each other to develop the software successfully. The result produced by a phase is used in another phase to develop the software. Also the technology implemented in one phase by a person may differ from other person in other phase.

To develop the process, designing phase is necessary and that design is created by the designer involved in that phase. The design is developed based on their idea and known technology. But it is necessary to convert the design according to the programming language used in the implementation phase. This conversion may leads to time consuming, when the programmer involves in this conversion since it has lot of work from identifying the tool to choosing the appropriate tool from the programming language implemented.

In this paper, the conversion of the tools becomes easy by using the re-engineering technique. The Re-Engineering technique is the process of converting the tools involved in the design from one technology into another automatically using some technique. In other words, alter the design from one concept to another is termed as Re-Engineering. The snapshot about this re-engineering technique is discussed and implemented in this paper.

Keywords: Designing, Implementation, Knowledge, phase, Programming language, Re-Engineering, Snapshot, Software Development Architecture, Technology, Thinking.

I. INTRODUCTION

In Software development architecture, more number of processes occurs such as analysis, design, implementation, testing and maintenance. In this paper, re-engineering technology is explained by converting the design from one technology to other technology.

Of these processes, the designing and implementation phase is more important and both are interconnected with one another. The designing is done by a design team based upon the analysis done by the analyst team. The designing must explains about the processes involved in the software development, technology used in that software, and the work flow of the software. These things are clearly explained in the designing and it is referred to as View Edition.

Basically, the designers are well trained in HTML, since it is easy to design. It is not necessary for the designers to know about the programming language implemented in the development stage. Sometimes, both may be entirely different. (i.e.,) the designing language is different from the implemented language. In that situation, the re-engineering technique helps the programmer to re-design the software by analyzing the tools used by the designers which is in HTML language. The programmer may re-design it in Java language using the techniques such as AWT or in Applet.

In that situation, the HTML design must be entirely converted into either AWT or in Applet. It takes more time to redesign it. To avoid this complexity and to perform the conversion, we use re-engineering technology. This technique is implemented to observe the tool and analyze the control in HTML and then convert it into tool in Java. The designing is must for GUI Programming, since it acts as the interactor between the application and the user. So it is carefully handled using the re-engineering technology.

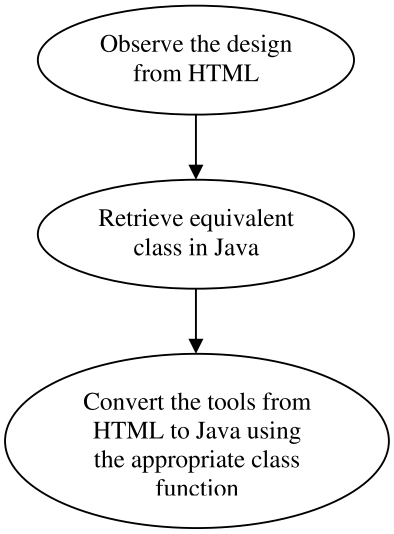

Observe the design from HTML
Retrieve equivalent class in Java
Convert the tools from HTML to Java using the appropriate class function

Figure-1 Conversion of View Edition

Each language has its own set of controls to design and from that design the implementation can be preceded. So, based upon the language and technology to be implemented, the re-engineering redesign the view edition and make it appropriate for further use. In this paper, the design from the HTML is converted into AWT and Applet is explained briefly in this paper by first analyzing the tools in HTML and then retrieves the appropriate classes in Java and then proceeds the conversion.

II. RELATED WORK

To implement the concept of re-engineering technology to convert the design from one technology to another, we analyze the works done by other researchers which are related to Re-engineering. The papers we referred are given below:

The paper [1], shortly describes methods and tools under development to support a model-based reengineering process of user interfaces of legacy applications. This reengineering process enables the use of Human Comptuer Interaction (HCI) patterns and allows an adaption of user interfaces to different contexts of use.

Comparable work of extracting user interface models from existing applications and reengineering was also done by Vanderdonckt et. al. [2,3]. They focussed on web-pages and applications written in the programming language C(++).

Other related work was done by Mainetti et.al. [4]. They are focussing on the redesign of web applications, also including code analysis to derive object models and business logic. To assist the reengineering of character based user interfaces to graphical user interfaces a model based approach was proposed by Tucker and Stirewalt [5]. It focusses on batch-command systems and utilizes an intermediate model comparable to data flow diagrams to determine required user input and application flow. It seems to have been abandoned though.

The main principle to achieve this is to translate HCI patterns into machine-readable attributed components when and where appropriate. We call these components “pattern instance components (PIC)” [6] as they contain an instance of their respective pattern. PICs are used to semiautomatically transform parts of abstract or concrete interfaces to follow a certain pattern. Arnout's [7] research on object-oriented pattern components inspired our proceeding, which is e.g. described in [8].

Taking it as input we are able to derive an AUI of this application in an UIML [9] related dialect. See USGP in [10] for details, we do not want to discuss that matter in this paper. In paper [11], they present a new tool for reengineering telecommunication systems, recovering the current architecture, and extracting state machines reflecting the system behavior. The tool is based on a structure graph of the architecture and allows architectural modifications with according code changes. The modifications are specified as graph transformations using FUJABA enabling the generation of a Java prototype, which is accessible via a GUI based on the Graphical Editor Framework (GEF) plug-in for the Eclipse workbench.

There exist several graph-based reengineering tools. The comparison with other projects following a graph-based approach, such as Rigi [12], Bauhaus [13], and GUPRO [14], shows that most of these tools lack the support provided by a high-level specification language. Hence, graph transformations cannot be specified in a declarative way. These projects also concentrate on reverse engineering and do not support software restructuring. The approach in [15] shows how refactorings for object-oriented software can be defined by using graph rewrite rules using FUJABA and AGG [16] for tool validation. AGG is a general tool environment for algebraic graph transformation following the interpretative approach.

The AGG environment consists of a graphical user interface and an interpreter, which can be used for the specification and prototypical implementation of Java applications with complex graph-structured data. The paper at hand presents a very similar approach but aims at the reengineering of programs written in a different kind of programming language. As they consider the software on a higher architectural level, without going into a detailed analysis of every single statement, the studied re-design transformations are also different. The FUJABA Tool Suite RE [17] is a collection of reengineering tools and plug-ins. It allows the parsing of Java source code and supports different kinds of static and dynamic analyses, such as recognition of design patterns and anti-patterns [18].

The paper [19], presents an approach of a software framework to support software developing engineers by handling procedural software. The framework offers practical assistance for separating the control flow from the numerical functionality. Especially by long-lived and growing software tools, it allows to implement a flexible and clear documentable workflow, based on a configurable state machine. A rudder design tool illustrates the approach. This tool has been continuously developed over years with several changes of the responsible engineer.

Web sites are rarely de-signed and developed to fit such a large variety of contexts of use as each context (e.g., each computing platform, each device) has its own set of constraints. This paper[20] describes a model-based approach for reengineering web pages into a presentation and a dialog model stored with XIML, a model-based user interface specification language. These models are then further exploited to reengineer other user interfaces either for the same context of use (by changing presentation design options) or for different contexts of use (by changing properties of computing platform model). For this purpose, three key elements of the presentation model (i.e. presentation units, logical windows, and abstract interaction objects) and two key elements of the dialog model (i.e., navigational structure and transition) were defined.

To address these demands, ad hoc development is no longer acceptable in terms of the cost and time required for software engineering and maintenance. Forward engineering [21] has been consequently considered as a good candidate for producing quality web sites. For instance, model-based approaches [22, 23, 24] can produce a user interface (UI) for a web site by exploiting knowledge captured in various models, such as the presentation and dialog models.

In the paper [20], they first report on what the assumptions are for adopting a model-based approach for forward engineering. Then, we describe a model-based approach for reverse engineering web pages implemented as an automatic or mixed-initiative process in the VAQUITA software, with an eye to applying forward engineering subsequently. Reengineering methods are then considered to produce new UIs for other contexts of use, thus creating a capability to rapidly produce UIs for different computing platforms, various access devices, etc.

Cross-platform development is not new as several environments provide support for this purpose: Galaxy [25] and Open Interface [26] render the same UI on different platforms with their native look & feel, while SUIT [27] employs a unique UI definition that can be processed on multiple platforms. However, not one of these systems truly adopts a model-based approach, although SUIT's common definition holds some presentation abstractions. CT-UIMS [28] pioneered the platform model by supporting some AIO [29] redistribution for OSF/Motif large screen and small Macintosh screens. AUIDL [30, 31, 32] is probably the first set of abstractions for reverse and re-engineering UIs: from internal hierarchical structures, type and variable declarations, a UI can be recovered in IDL with a presentation model (based on OO paradigm) and a dialog model (based on Milner's process algebra).

In the paper [33], we put forward a methodology for reengineering the architecture of a legacy software system. The proposed approach is not restricted to any specific source and target architectures, or programming language. It consists in (1) achieving a representation of the source code through its categorization and structuring, (2) transforming it into the new intended architecture, and (3) generating the code for the target platform. First, the code is categorized according to its purpose by pre-defined rules and represented as a model that is an instance of a type graph. Then, this representation is transformed into the intended target architectural paradigm using graph transformation techniques. The generation of the target code is not covered in this report but will be studied in the near future. The approach attempts to address problems that are repeatedly encountered in legacy reengineering industry projects.

In the paper[34], they first show how we use Rational Rose to produce UML diagrams of SNiFF+ projects. Afterwards we give a quick introduction to the Information Exchange Model that is used for the intermediate format and its representation in the industry standard CDIF.

The quality of the Software RefineryTM KBSE environment derives significantly from the quality of its embedded Refine programming language. While Refine provides strong support for many modern high-level programming paradigms (functional, logic, object-oriented and metaprogramming), a number of improvements seem appropriate. Suggested improvements are (i) motivated through analysis of Refine's existing capabilities, (ii) embodied in suggested language changes, and (iii) validated by implementation feasibility studies is discussed in paper[35].

III. METHODOLOGY

A. *Proposed Method*

The aim of the proposed method is to convert the View Edition GUI Tools from HTML to Java AWT and Applet.

The summary of the proposed method is as follows: The first step of this conversion is that carefully observe the input objects in HTML. Then extract the equivalent classes in Java for the input objects. Then convert the input objects into appropriate tools available in Java. Thus the tools are converted into appropriate programming language implemented to develop the process.

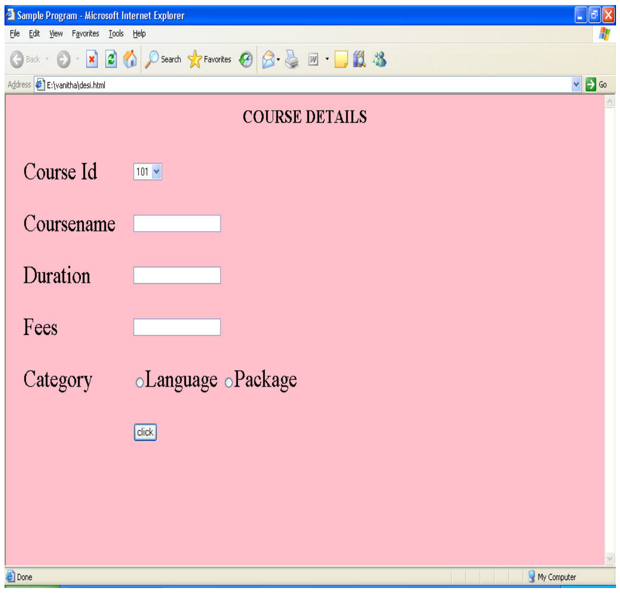
The tools are designed in HTML by using **<input>** tag. These tools are converted into java by using the **awt** package. In that package, there exist many built-in classes for Textbox, Radiobutton, Checkbox, Label and so on.

The first step involves only in conversion which discards the designing such as alignment, color, height, width and so on. The second step of the re-engineering process is to monitor the designs which are designed in CSS (Cascading Style Sheet). Now these are examined and the converted design in first step is now implemented with these styles using the suitable methods in AWT and Applet.

The styles applied in cascading style sheet is done in either internal style sheets or external style sheets using font-color, font-family, text-height, etc., Whereas in java these are available in Font class with parameters such as dialog, style and size. And the colors are applied by using setBackground and setForeground properties.

By using the above two steps, the re-engineering technique is used to convert the design from HTML to Java's AWT and Applet including the styles applied in the HTML design. A mock-up that illustrates the re-engineering technology is given below in which both the HTML design and the Java design are shown.

HTML DESIGN:


Sample Program - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address E:\vanitha\desi.html
Go
Done
My Computer

COURSE DETAILS

Course Id 101

Coursename

Duration

Fees

Category ○Language ○Package

click

Figure-2: Design in HTML

AWT DESIGN:

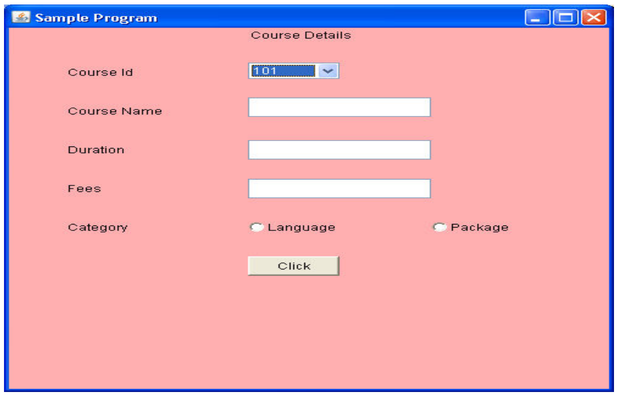


Figure-3: Design in AWT

APPLET DESIGN:

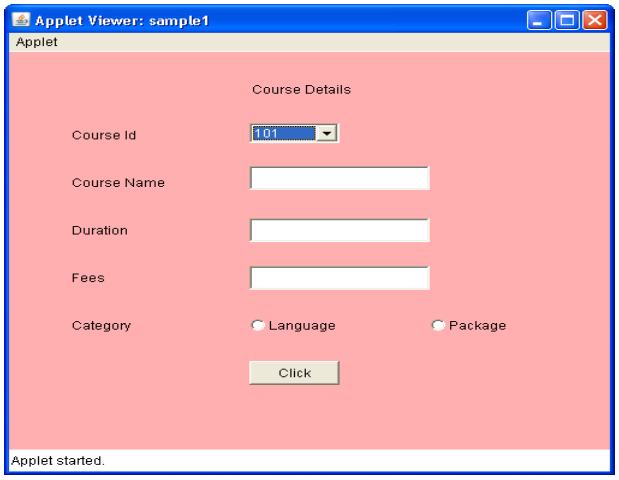


Figure-4: Design in Applet

B. Algorithm

```
Start the Process
Step-1:
Function: read ()
{
Count = Count the number of lines
Open the HTML Output Design
Read the First Line
Identify the Tag, Attribute and its value
Control = Control Name
Eg: <input type=text name=t1>
}
Step-2:
Function: readstyles ()
{
Read the height and width of the controls in HTML
Observer the color of the form
Height = height of the control
Width = width of the control
Color = color of the form

}
```

```
Step-3:
   Function: conversion
   {
   Search for the equivalent class in Java
   Identify the controls
   Javacontrol = Control Name
   If(Control ==Javacontrol)
   {
        Create Object in Java
        Eg: TextField t1=new TextField(20);
        Fix the height and width of the control using
            setBounds(x, y, Width, Height);
        Change the color of the form using
            setBackground(Color.COLOR);
   count = count--;
}
}
Step-4:
Function: Repeat
{
If(Count != 0)
{
   Goto Step-1 to read the next line
}
Else
Show the result in Java
Stop the process.
```

IV. EXPERIMENTAL RESULTS

The technique Re-Engineering is used to re-design the forms from one technology to another which helps the programmer to implement the code according to their platform. The designer just creates the tools in their platform. In this paper, this technique is implemented in software development architecture in which the designer designs the forms in HTML language which is easy for them.

But the programmer who implements the design may need to convert the design according to their programming language. If they want to convert it into GUI Applications such as AWT and Applet, then the re-engineering technique is implemented to automatically convert the design from HTML and generate the output in GUI Applications (i.e) AWT and Applet.

This process is done by reading the design in HTML line by line and then converts it into AWT and Applet. Thus the programmer who wants the design in AWT and Applet may easily get the design in less time using the Re-Engineering technology. Each programming language has its own set of functions to develop the applications. These functions are stored in the memory to convert the design based upon the selected programming language. This conversion is generated by matching the controls and styles applied in one technology (HTML) with the functions in other technology (AWT and Applet).

Thus this paper provides efficient tools to convert the view edition into GUI application using Re-Engineering technique.

V. CONCLUSION

The aim of the paper was to convert the view edition from HTML to Java platform. In Java, the commonly used GUI Tools are AWT and Applet. So conversion from HMTL to AWT and Applet is explained in this paper.

The conversion is made possible by creating and maintaining the design tools for each programming language. Then the conversion process is carried out by analyzing and identifying the appropriate controls from the destination language such as AWT and Applet, which are suited for the control in the Source language such as HMTL.

This conversion is done by implementing the concept of Re-Engineering which reduces the time for the programmer to convert the design according to their language.

In future, the re-engineering technology can be implemented to convert the code from one language to another language is developed to improve the progress of Re-Engineering.

VI. REFERENCES

[1] " Model-based Reengineering of User Interfaces Andreas Wolff, Peter Forbrig", University of Rostock Institute of Computer Science Albert Einstein Str. 21, 18059 Rostock, Germany

[2] J. Vanderdonckt, L. Bouillon, N. Souchon, Flexible Reverse Engineering of Web Pages with VAQUITA, in Proceedings of IEEE 8th Working Conference on Reverse Engineering WCRE'2001 (Stuttgart, 2-5 october 2001, IEEE Press, Los Alamitos, 2001, pp. 241-248.

[3] L. Bouillon, J. Vanderdonckt, J. Eisenstein, Model-Based Approaches to Reengineering Web Pages, in Proceedings of 1st International Workshop on Task Models and Diagrams for user interface design TAMODIA'2002, INFOREC Printing House, Bucharest, 2002, pp. 86-95.

[4] Mainetti, Paiano, Pandurini: User-Centered reverse engineering: Genesis-D project, in proceedings of Web Maintenance and Reengineering 2006, CEUR Workshop Proc. 193

[5] Tucker, K. and Stirewalt, R.: Model based userinterface reengineering, in Proc. 6th WCRE, 1999. http://citeseer.ist.psu.edu/tucker99model.html

[6] Rathsack, Wolff, Forbrig: Using HCI-Patterns with Model-based Generation of Advanced User-Interfaces, Proc. of MDDAUI 2006, Models 2006, Genova, Italy

[7] Arnout, Karine: From Pattern to Components, PhD dissertation, Swiss Institute of Technology, Zurich 2004

[8] Wolff, A.; Forbrig, P.; Dittmar, A.; Reichart, D.: Tool Support for an Evolutionary Design Process using Patterns, Proc. of Workshop on Multi-channel Adaptive Context-sensitive Systems 2006, Glasgow, GB, p. 71-80

[9] UIML, User Interface Markup Language http://www.uiml.org

[10] Müller, Andreas: Spezifikation geräteunabhängiger Benutzerschnittstellen durch Markup-Konzepte, PhD dissertation, University of Rostock, 2003

[11] RePLEX: A Model-Based Reengineering Tool for PLEX Telecommunication Systems Christian Fuss, Christof Mosler, Marcel Pettau [fuss|mosler|pettau]@i3.informatik.rwth-aachen.de http://www.se.rwth-aachen.de Department of Computer Science 3 (Software Engineering) RWTH Aachen University, Germany

[12] H. A.M¨uller, K.Wong, S. R. Tilley. Understanding Software Systems Using Reverse Engineering Technology. In The 62nd Congress of L'Association Canadienne Francaise pour l'Avancement des Sciences ACFAS 1994. Pp. 41–48. Montreal, Canada, May 1994.

[13] R. Koschke. Atomic Architectural Component Recovery for Program Understanding and Evolution. Doctoral thesis, Institute of Computer Science, University of Stuttgart: Stuttgart, Germany, Stuttgart, Germany, 2000. 414 pp.

[14] J. Ebert, B. Kullbach, V. Riediger, A. Winter. GUPRO – Generic Understanding of Programs: An Overview. Electronic Notes in Theoretical Computer Science 72(2), 2002. URL: http://www.elsevier.nl/locate/entcs/volume72.html.

[15] T. Mens, N. Van Eetvelde, S. Demeyer, D. Janssens. Formalizing refactorings with graph transformations. Journal on Software Maintenance and Evolution: Research and Practice, pp. 247–276, 2005.

[16] FUJABA – From UML to Java and Back Again. 1999. http://www.fujaba.de/.

[17] FUJABA Tool Suite RE. 2005. http://wwwcs.uni-paderborn.de/cs/fujaba/ projects/reengineering/.

[18] J. Niere, W. Sch¨afer, J. P. Wadsack, L. Wendehals, J. Welsh. Towards Pattern-Based Design Recovery. In Proc. of the 24th International Conference on Software Engineering (ICSE), Orlando, Florida, USA. Pp. 338–348. ACM Press, May 2002.

[19] Software Development and Reengineering outside of the IT Industry using a Procedural Workflow Framework Wilfried Abels, TU Hamburg-Harburg, Hamburg/Germany, w.abels@tu-harburg.de Lars Greitsch, TU Hamburg-Harburg, Hamburg/Germany, lars.greitsch@tu-harburg.de

[20] Model-Based Approaches to Reengineering Web Pages Laurent Bouillon, Jean Vanderdonckt Université catholique de Louvain Belgian Lab. of Computer-Human Interaction IAG-ISYS, Place des Doyens, 1 B-1348 Louvain-la-Neuve, Belgium +32-10/47.{8349, 8525} {bouillon, vanderdonckt}@isys.ucl.ac.be Jacob Eisenstein University of Southern California Department of Computer Science 941 W. 37th Place Los Angeles, CA 90089-0781 USA +1 213 740 4496 jacob@isi.edu

[21] F. Bodart, A.-M. Hennebert, J.-M. Leheureux, and J. Vanderdonckt, "Computer-Aided Window Identification in TRIDENT", Proc. of the 5th IFIP TC13 Conf. on Human- Computer Interaction Interact'95 (Lillehammer, 25-29 June 1995), Chapman & Hall, London, 1995, pp. 331-336. Accessible at http://www.info.fundp.ac.be/cgi-publi/pub-specpaper?RP-95-021

[22] R.E.K. Stirewalt, "MDL: A Language for Binding User- Interface Models", in [26], pp. 159-184.

[23] P. Szekely, P. Luo, and R. Neches, "Beyond Interface Builders: Model-Based Interface Tools", Proc. of ACM Conf. on Human Aspects in Computing Systems InterCHI'93, ACM Press, New York, 1993, pp. 383-390

[24] J. Vanderdonckt and P. Berquin, "Towards a Very Large Model-based Approach for User Interface Development", Proc. of 1st Int. Workshop on User Interfaces to Data Intensive Systems UIDIS'99, IEEE Computer Society Press, Los Alamitos, 1999, pp. 76-85.

[25] J. Eisenstein and A. Puerta, "Adaptation in Automated User- Interface Design", Proc. of ACM Int. Conf. on Intelligent User Interfaces IUI'2000 (New Orleans, 9-12 January 2000), ACM Press, New York, 2000, pp. 74-81.

[26] M.M. Moore and S. Rugaber, "Domain Analysis for Transformational Reuse", Proc. of 4th Working Conf. on Reverse Engineering WCRE'97 (6-8 October 1997), IEEE Computer Society Press, Los Alamitos, 1997.

[27] "Open Interface™", Neuron Data, 156 University Avenue, Palo Alto, CA 94301, 1992.

[28] F. Lonczewski and S. Schreiber, "The FUSE-System: an Integrated User Interface Design Environment", Proc. of 2nd Int. Workshop on Coputer-Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996), Presses Universitaires de Namur, Namur, 1996, pp. 37-56. Accessible at ftp://hpeick7.informatik.tu-muenchen.de/pub/papers/sis/fuse_ca dui96.ps.gz

[29] M.M. Moore, "Representation Issues for Reengineering Interactive Systems", ACM Computing Surveys, Vol. 28, No. 4, December 1996. Article # 199. Accessible at http://www. acm.org/pubs/articles/journals/surveys/1996-28-4es/a199- moore/a199-moore.html

[30] Generation and Interaction", Proc. of Interact'90, Elsevier Science Pub., Amsterdam, 1990, pp. 651-657.

[31] E. Merlo, J.F. Girard, K. Kontogiannis, P. Panangaden, and R. De Mori, "Reverse Engineering of User Interfaces", Proc. of 1st Working Conference on Reverse Engineering WCRE'93 (Baltimore, 21-23 May 1993), R.C. Waters, E.J. Chikofsky (eds.), IEEE Computer Society Press, Los Alamitos, 1993, pp. 171-179.

[32] E. Merlo, P.-Y. Gagné, and A. Thiboutôt, "Inference of graphical AUIDL specifications for the reverse engineering of user interfaces", Proc. of Int. Conf. on Software Maintenance (19-23 September 1994), IEEE Computer Society Press, Los Alamitos, 1994, pp. 80-88.

[33] Software Reengineering at the Architectural Level: Transformation of Legacy Systems R. Correia1;2, C. Matos1;2, M. El-Ramly1, R. Heckel1, G. Koutsoukos2, L. Andrade2 1Department of Computer Science, University of Leicester, U.K. 2ATX Software, Lisboa, Portugal frmc20,cmm22,mer14,reikog@le.ac.uk fgeorgios.kousoukos,luis.andradeg@atxsoftware.com

[34] SNiFF+ Talks to Rational Rose Interoperability using a Common Exchange Model Sander Tichelaar and Serge Demeyer, Software Composition Group, University of Berne, Switzerland, {tichel,demeyer}@iam.unibe.ch

[35] Refine "Refine" – Towards a More Expressive Reengineering Tool Development Platform Paul A. Bailes Ian Peake Centre for Software Maintenance School of Information Technology and Electrical Engineering The University of Queensland QLD 4072 AUSTRALIA