An Eclipse Based Environment to Define and Execute Processes with Application to the Reverse Engineering[1]

Andrea De Lucia[+], Fausto Fasano[+], Michele Risi[+], and Giuseppe Scanniello[*]

[+] Dipartimento di Matematica e Informatica, Università di Salerno, Via Ponte Don Melillo, 84084, Fisciano (SA), ITALY
[*] Dipartimento di Matematica e Informatica, Università della Basilicata, Viale Dell'Ateneo, Macchia Romana, 85100, Potenza, ITALY
{adelucia, ffasano, mrisi}@unisa.it, giuseppe.scanniello@unibas.it

Abstract. In this paper we present an Eclipse plug-in for defining and executing processes to reverse engineering and comprehend traditional and web based information systems. Processes are defined in terms of UML activity diagrams, where predefined or newly developed software components can be associated to each activity. Components implemented using traditional programming languages and software environments for data analysis can be reused.

Keywords: Program comprehension, reverse engineering, legacy systems, visual environment.

1. Introduction

Maintenance is the largest and most expensive activity in the software engineering field [17]. In fact, during maintenance a software system will be continuously changed and enhanced for several reasons. Changes are needed to meet new user requirements, to adapt software to interact with external entities, including people, organizations, and artificial systems, to correct faults, to improve performances, etc...

Technical and managerial problems contribute to the costs of software maintenance, which increases when documentation lacks. Supporting tools to reverse engineer existing or legacy systems could be used to support software engineers in the maintenance phase. These tools should generally support the comprehension of existing software systems, abstracting higher level models from lower level representations of a software system.

Different software tools are available to reverse engineer and comprehend existing software systems, both freeware and commercial. Generally, all-in-one reverse engineering and comprehension tools often lack. Hence, in case a specific tool is needed a high effort is required to design and develop it from scratch. To ease the production of reverse engineering tools several software environments tools have been proposed in the literature[4][5][12]. Generally, these environments are based on domain specific or scripting language to prototype and/or generate reverse

[1] This work has been supported by the project METAMORPHOS under grant PRIN-2006-2006098097.

engineering tools. However, even using these environments, the development of reverse engineering tools might require a sensible effort and specific user competences.

Many reverse engineering components are available to be combined within reverse engineering processes. Examples of such components are source code analyzers, clustering tools, graph visualizers, and so on. In this case, most of the effort consists of integrating these components into the required reverse engineering process. In such a scenario, Workflow Management technologies [26] could represent a viable solution to support modeling, integration, and orchestration of processes composed of different and heterogeneous software components. However, Workflow Management Systems (WfMSs) have not been used to integrate different tools and components during the definition and execution of reverse engineering and comprehension processes.

This paper presents an Eclipse plug-in, to define and execute processes aimed at comprehending both traditional and web based information systems. The plug-in presents a visual environment, where the software engineer defines processes expressed in terms of UML activity diagrams. Predefined or newly developed components can be associated to each activity of the process. Software components implemented using traditional programming languages and software environments for data analysis (e.g., MATLAB or R) can be also reused. Once the process has been fully defined the software engineer executes it. To assess the plug-in we defined a process to group static and dynamic web pages that are similar at the structural level. The process has been then executed on two dynamic web sites.

The remainder of the paper is organized as follows. In Section 2 we present related work, while the plug-in is discussed in Section 3. The case study is discussed in Section 4, while final remarks and future work conclude the paper.

2. Related work

Workflow Management Systems [26] have been developed by researchers to provide support to the modeling, improvement, and automation of business and industrial engineering processes [6][13]. In case they are used to support software processes [14][15] WfMSs are also named as PSEE (Process-centered Software Engineering Environment). Most of the WfMSs are client-server systems, with centralized enactment facilities, although they do not exploit the web as basic infrastructure to ease the accessibility by remote users. Recent research on workflow management is focusing on the use of web technologies and/or specialized middleware to support distributed processes across organizations [2][6][13][15].

A number of Process Definition Languages have been proposed in the literature, based on several formalisms such as event-condition-action mechanisms [2], graph rewriting mechanism [14], Petri Nets [3], etc.. Several authors propose to adopt UML for representing business processes [10][19], as it has been conceived for the communication among people and it has a notation that can be easily understood and used. Although the usefulness of WfMSs based on UML has been widely recognized in the software engineering field [10][14][19], they have not been used to integrate different tools during the definition and execution of reverse engineering and comprehension processes.

Regarding systems to develop code analyzers tools, Canfora *et al.* [4] propose a system to produce analyzers to derive structural information at different abstraction levels, ranging from fine grained (e.g., control and data flow analysis and dependence) up to architectural models (e.g., structure charts, abstract data types, and object classes). Analyzers are automatically generated from a high-level specification expressed in a domain-oriented language and can be also embedded into C programs using foreign interfaces. The system and the underlying approach have been also assessed through a case study for software remodularization. The achieved results are presented and discussed in [5].

Ducasse *et al.* [12] describe the Moose reengineering environment. It is focused on a language independent meta-model and offers services like grouping, querying, navigation, and advanced tool integration mechanism. Differently from our proposal it does not provide a visual environment where processes to reverse engineer or comprehend legacy information systems can be defined and executed. Wong in [25] describes the Rigi system. This system provides several features and supports the comprehension and the re-documentation of existing software systems implemented in C, C++, and COBOL. A visual environment is also provided to help software engineers. The Rigi system is adaptable to different languages only to analyze software systems from the structural point of view. The Software Refinery toolkit [20] uses an object-oriented database, called REFINE, to store a fine grained program model consisting of an attributed abstract syntax tree. This toolkit offers languages based on the procedural, declarative, and object-oriented paradigms to implement analyzers and to produce parsers and user interfaces, respectively. Our approach is different from the previous approaches, as we concentrate on the definition and execution of reverse engineering processes and the integration and orchestration of the different components and analyzers required during the comprehension process.

It is worth noting that reverse engineering approaches are generally focused on structural information. Nevertheless, the domain knowledge of the developers is generally embedded in the code comments. Kuhn *et al.* in [16] describe an approach to group software artifacts using Latent Semantic Indexing [8], a widely known and adopted information retrieval technique. The approach is language independent and tries to group source code containing similar terms in the comments. The authors consider different levels of abstraction to understand the semantics of the code (i.e., methods and classes). The tool implementing the approach is named Hapax and is built on top of the Moose reengineering environment [12]. Case studies are used to assess the approach and the tool support.

To group software artifacts, clustering based approaches have been largely adopted in the past [1][7][9][21][23][24]. For example, Wiggerts [24] introduces clustering algorithms widely employed to group entities into software modules. This work has been extended by Anquetil and Lethbridge in [1]. In particular, the authors present a comparative study of different hierarchical clustering algorithms and analyze their properties with regard to software remodularization. In [22] the Arch tool is discussed. It is a graphical and textual structure chart editor for maintaining large software systems that can be used to group related procedures into modules using a clustering algorithm. Clustering algorithms have been also used to comprehend legacy web applications. For example, Di Lucca *et al.* [9] propose a clustering method to decompose a web application into groups of functionally related components. Ricca

and Tonella [21] propose a semi automatic approach to identify clusters of duplicated or similar pages to be generalized into a dynamic page. Similarly, the same authors propose a semiautomatic approach to identify and align static HTML pages whose structure is the same and whose content is in different languages [23]. An approach based on a competitive clustering algorithm is proposed in [7] to identify similar pages at the structural level. Page structures are encoded into strings and then the Levenshtein algorithm is used to compute the distances between pairs of pages.

3. The Eclipse Plug-in

The proposed plug-in enables to visually define and execute processes in terms of UML activity diagrams to reverse engineer or comprehend existing traditional and web based information systems. A snapshot of the plug-in is shown in Fig. 1. In particular, this figure shows a process within the graphical editor integrated within the proposed plug-in. We used the Eclipse Graphical Editing Framework[2] (GEF) to implement this graphical editor.

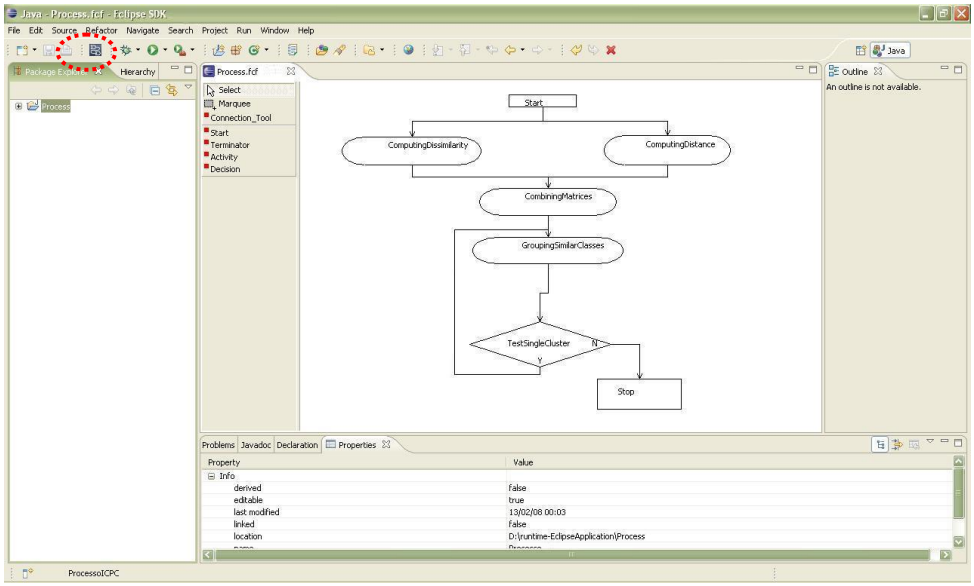

Fig. 1. The proposed plug-in

A software engineer first arranges the activities (or the phases) of a process and then associates a software component to each of them. These components can be implemented using both traditional programming languages and software environments for data analysis. Indeed, we posed great emphasis on the reuse of

2 http://www.eclipse.org/gef/

software component implemented using the R[3] and MATLAB[4]. This was due to the fact that these environments are widely used and a huge number of existing components are freely available for reuse.

To integrate software components implemented in R, the framework provides an automatic support using Rserve[3]. In particular, a Java class is used to remotely invoke procedures written in R, which in turn are encapsulated by employing the Adapter pattern. MATLAB routines are integrated using the JMatLink engine[5]. Similarly to the R procedures the Adapter pattern is used to wrap MATLAB routine.

In order to associate existing software components to the activities of the defined process the plug-in provides an easy to use mechanism. The framework provides predefined components implemented in R, MTLAB, and Java, while are ready to be reused. For example, Fig. 2 shows how an existing component implemented in R is associated to an activity (i.e., *CombiningMatrices*) of the process shown in Fig. 1. Information on the process in terms of activities and their relationships are stored in an XML file. This file also contains information concerning the software components associated to the activities of the process. An excerpt of the XML file of the process shown in Fig. 1 is depicted in Fig. 3. In particular, Fig. 3 shows how the activity CombiningMatrices and the *lsa_procedure.r* software component have been associated. The procedure of *lsa_procedure.r* that has to be invoked (i.e., *mergeMatrixFromFile*) when the process is executed is shown as well.

The execution of an R existing software component within the defined process is enabled using a wrapper that has been implemented once for all. An excerpt of the wrapper code is shown in Fig. 4. The wrapper uses the class *RManager* to enable the communication with existing components. The information specified during the process definition (see Fig. 1 and Fig. 2) is employed as well. For example, the values of the parameters allow the framework to get the location of the software components, to specify the library to use, and to invoke the procedures to be employed in the process execution. Software components implemented in MATLAB are managed in a similar way through a middleware component implemented once for all. Consequently, the method to associate existing MATLAB or R components to the activities of a given process is nearly the same. The framework also enables the reuse of existing component implemented in Java. In such a case wrappers are not required.

The framework can be easily extended to manage existing software components implemented using different technologies. This is due to the flexible architecture we have designed. However, in case software components implemented using different technologies have to be managed, a suitable software wrapper has to be implemented.

In case a software engineer has to define a new activity, the framework proposes an editor where Java source code can be written. To specify conditions (i.e., diamond within a UML activity diagram) the same Java editor is used, which also provides features to highlight and indent the code. The code of both activities and conditions is compiled within the editor and then executed when the software engineer runs the process.

---

3 http://www.r-project.org/

4 http://www.mathworks.com/

5 Available under GPL license from http://www.held-mueller.de/JMatLink/

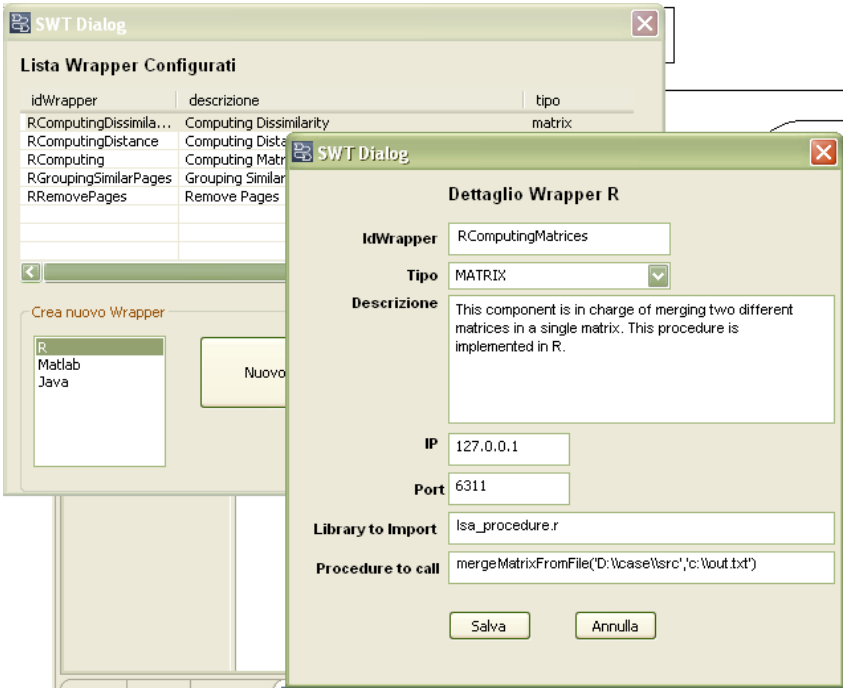

Fig. 2. Associating an existing software component to an activity

```
<matrix  id="RComputingMatrices" class-name="cddTools.MatrixGeneratorR" class-path="">
        <parametri-generali>
                <p-g name="IP" value="127.0.0.1" />
                <p-g name="PORT" value="6311" />
                <p-g name="PROC" value="mergeMatrixFromFile('D:\\case\\src','c:\\out.txt')" />
                <p-g name="LIB" value="source('lsa_procedure.r')" />
        </parametri-generali>
        <parametri-posizionali></parametri-posizionali>
</matrix>
```

Fig. 3. An excerpt of the XML file of the process

The plug-in also provides a higher level of reusability. Indeed, a software engineer can reuse predefined processes and then associate to each activity a newly or an existing software component. This level of reusability was introduced as we observed that some reverse engineering approaches proposed in the literature [7][9][21][23][24] share nearly the same process.

```
public Object esegui() throws Exception {
        String result = "0";
        try {
          rUtility.RManager rMan = new RManager(this.getParametroGenerale("IP"),
                  Integer.parseInt(this.getParametroGenerale("PORT")),
                  this.getParametroGenerale("LIB"));
          rMan.init();
          rMan.start();
          result = rMan.procedure(this.getParametroGenerale("PROC").replace("'", "\'"));
          rMan.stop(false);
        } catch (RException e) {
                  logger.error("esegui()", e);
        }
        return result;
}
```

Fig. 4. The wrapper code

Once the process has been fully defined the software engineer executes it within the framework (see dashed ellipse in Fig. 1). In case the early activities of a process require a lot of computation time and they have been previously executed, the software engineer can decide to skip them. This is possible due to the fact that closer activities communicate by storing the produced artifacts on the prototype file system.

4. Case study

The plug-in has been assessed on different processes to reverse engineer and comprehend existing traditional and web based systems. However, in this paper we describe a process conceived to group similar static and dynamic web pages at the structural level. Fig. 5 shows this process within the graphical editor of the proposed plug-in.

The *Page Transformation* phase produces suitable representations of the pages to be considered in the identification of pages that are similar at the structural level. The structures of the web pages are encoded into strings and then provided as input to the *Computing Distance Matrix* phase, which is in charge of computing distances between pairs of pages.

Distances between pairs of pages are computed using the Levenshtein algorithm [18] on the strings encoding the web pages. Page distances are used in the *Computing Distance Matrix* phase to compute the distance matrix, which is then provided as input to the *Clustering Web Pages* phase. This phase uses one of the widespread partitional clustering algorithms to group observations from the statistical recognition perspective, i.e., WTA (Winner Takes All) [11]. Further detail on the process can be found in [7].

The approach has been evaluated on two web applications developed using JSP technology. In particular, we considered the web application of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 02) and the web application named SRA (Student Route Analysis). SEKE 02 was used to support the organizers and the academic community for paper submission, refereeing, and

conference registration. On the other hand, the SRA web application was devised to provide the students in Politics Science at University of Salerno with statistics and information on their academic carriers. We have executed the defined process on these applications. To identify possible differences in using the plug-in, we have compared the achieved results with the outcomes presented in [7]. As expected, we did not identify any difference in results.

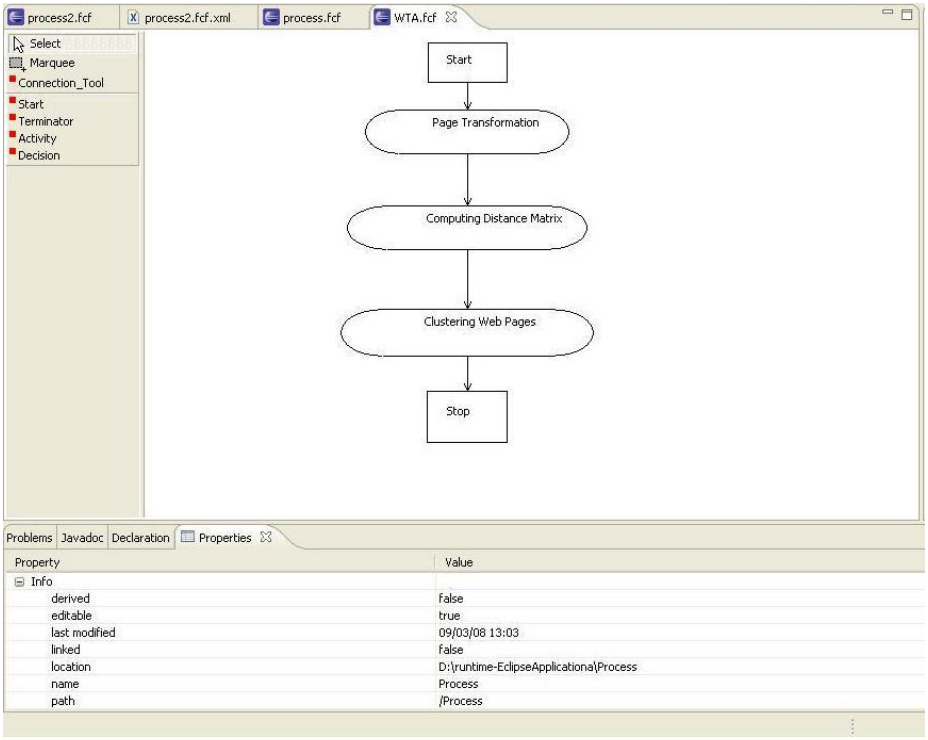

Fig. 5. A process to group similar pages

5. Conclusions and future work

In this paper we have presented an Eclipse plug-in to define and execute processes to reverse engineer and comprehend existing web based and traditional software systems. It integrates a visual environment implemented using GEF, where processes expressed in terms of UML activity diagrams, are specified. Software engineers complete the definition of the process associating newly developed or predefined software components (i.e., implemented in R, MATLAB, and Java). The association between existing components and activities is specifiable in an easy way using a form based user interface. However, due to the designed architecture, the framework can be

easily extended to manage existing software components implemented using different technologies. The framework also supports reuse and customization of previously defined processes. The framework has been assessed on processes to comprehend existing web applications [7] and remodularize legacy information systems, respectively.

We are currently working to further assess the proposed environment. Future work will be also devoted to further experiment the environment on different processes. We also plan to empirically validate the usefulness of the plug-in. To this end, students, academic researchers, and professional programmers will be selected as subjects. In particular, we plan to perform controlled experiments and actual industrial case studies with professional programmers. Usability studies will be also performed.

References

[1] Anquetil N. and Lethbridge T.C.:Experiments with Clustering as a Software Remodularization Method, Proc. of 6th Working Conference on Reverse Engineering, IEEE CS Press (1999) 235-255.

[2] Aversano, L., De Lucia, A., Gaeta, M., Ritrovato, P., Stefanucci, S., and Villani, M.L.: Managing Coordination and Cooperation in Distributed Software Processes: the GENESIS Environment, In Software Process Improvement and Practice, vol. 9, no. 4, Wiley, (2004) 239-263.

[3] Bandinelli, S., Di Nitto, E., and Fuggetta A.,:Supporting cooperation in the SPADE-1 environment. IEEE Trans. on Software Engineering, vol. 22, no. 12, (1996) 841-865.

[4] Canfora, G.; Cimitile, A.; De Carlini, U.; De Lucia, A.: An extensible system for source code analysis, IEEE Trans. on Software Engineering, vol. 24, no. 9, (1998) 721-740.

[5] Canfora G., De Lucia A., and Di Lucca G.A.:A System for Generating Reverse Engineering Tools: A Case Study of Software Modularisation, In Automated Software Engineering, vol. 6, no. 3, (1999) 233-263.

[6] Cugola, G., Di Nitto, E., and Fuggetta, A.:The JEDI event-based infrastructure and its application to the development of the OPSS WfMS. In IEEE Transactions on Software Engineering, vol. 27, no. 9, (2001) 827-850.

[7] De Lucia A., Scanniello G., Tortora G.: Identifying Similar Pages in Web Applications using a Competitive Clustering Algorithm. In. Journal on Software Maintenance and Evolution, vol. 19, no. 5, John Wiley & Sons, (2007) 281-296.

[8] Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., and Harshman R.:Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science, no. 41, (1990) 391-407.

[9] Di Lucca G. A., Fasolino A. R., De Carlini U., Pace F., and Tramontana P.: Comprehending web applications by a clustering based approach. Proc. of the 10th International Workshop on Program Comprehension, IEEE CS Press (2002) 261-270.

[10] Di Nitto, E., Lavazza, L., Schiavoni, M., Tracanella, E., and Trombetta, M.:Deriving executable process description from UML. In Proceedings of 24th International Conference on Software Engineering, Orlando, Florida, ACM Press, (2002) 155-165.

[11] Duda R. O., Hart P. E., and Stork D. G.: Pattern Classification. Wiley-Interscience Publication, JOHN WILEY & SONS, Inc. New York, 576-581.

[12] Ducasse S., Girba T., Lanza M., and Demeyer S.:Moose: a Collaborative and Extensible Reengineering Environment. Tools for Software Maintenance and Reengineering, RCOST / Software Technology Series, Franco Angeli, (2005) 55-71.

[13] Eder J., and Panagos E.: Towards Distributed Workflow Process Management. AT&T Research Labs (1999).
[14] Heimann, P., Joeris, G., Krapp, C. A., and Westfechtel, B.:DYNAMITE: dynamic task nets for software process management. In Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, (1996) 331-341.
[15] Kaiser, G.E., Dossick, S. E., Jiang, W., Yang, J. J., and Xi Ye S.:WWW-based Collaboration Environments with Distributed Tool Services. World Wide Web Journal, vol. 1 no. 1, (1998) 3-25.
[16] Kuhn A., Ducasse S., and Girba, T.:Enriching reverse engineering with semantic clustering, Proc. of 12th Working Conference on Reverse Engineering, IEEE CS Press, (2005) 10-20.
[17] Lehman MM and Ramil JF, Software Evolution and Software Evolution Processes, Annals of Software Engineering, In Software Process-based Software Engineering, vol. 14, (2002) 275 – 309.
[18] Levenshtein V. L.:Binary codes capable of correcting deletions, insertions, and reversals. In Cybernetics and Control Theory, vol. 10, (1966) 707-710.
[19] Marshall, C.:Enterprise Modellingwith UML: Designing Successful Software trough Business Analysis. Addison-Wesley, (2000).
[20] Reasoning Systems, REFINE User's Guide, Reasoning Systems, Palo Alto, CA, 1989.
[21] Ricca F. and Tonella P.:Using Clustering to Support the Migration from Static to Dynamic Web Pages. Proc. of International Workshop on Program Comprehension, Portland, Oregon, USA, (2003) 207-216.
[22] Schwanke R. W.:An Intelligent Tool for Re-engineering Software Modularity. Proc. of the 13th International Conference on Software Engineering, (1991) 83-92.
[23] Tonella P., Ricca F., Pianta E., and Girardi C.: Restructuring Multilingual Web Sites. In Proc. of International Conference on Software Maintenance, Montreal, Canada, IEEE CS Press (2002) 290-299.
[24] Wiggerts T.A.:Using Clustering Algorithms in Legacy Systems Remodularization, Proceedings of 4th Working Conference on Reverse Engineering, IEEE CS Press (1997) 33 - 43.
[25] Wong. K.: Rigi blurb, available at http://rigi.cs.uvic.ca/downloads/papers/pdf/blurb-rigi.pdf, (1996).
[26] Workflow Management Coalition.: Workflow Management Coalition Interface 1: Process Definition Interchange Process Model. Doc. no. WFMC-TC-1016-P (1999).