Proceedings of the 14th Colloquium for Information Systems Security Education
Baltimore Marriott Inner Harbor
Baltimore, Maryland June 7 - 9, 2010

Using the Workflow Technology in Secure Software Engineering Education

India Waddell, Nadia Jones, Crystal Steed, Xiaohong Yuan, and Yaohang Li, *North Carolina A&T State University*

Abstract – Security has become an increasingly important topic in software engineering. In this paper, an approach of using the workflow technology in teaching secure software engineering courses is presented. This approach can free students from low-level tools manipulation and command line interactions so that students can focus on learning the important secure software principles. Four case studies using the workflow technology, including using a local static analysis tool for code review, using a remote tool for code analysis, integrating local and remote tools, and implementing a web service fuzzer for penetration tests, are presented. Our educational practice has shown that the benefits of using the workflow technology in teaching secure software engineering classes have been well received by the students.

Index terms – workflow technology, secure software engineering

I. Introduction

In recent years, the use of distributed software applications in industry and academy has been increasing significantly, enabled by enhancing Internet bandwidth and availability, development of web technologies, and emergence of Grid computing [1] and cloud computing [2]. More and more software engineering practice is moving toward the Service-Oriented Architecture (SOA) [3] based applications, where the focus of software development is shifting from program writing to service composition. The premise of SOA is to erase application boundaries and technology difference to allow interoperability among potentially distributed software services. On one hand, SOA adds a new dimension to software engineering practice by adapting and integrating a variety of distributed services and tools [4]. On the other hand, engineering of software based on SOA faces a lot of new challenges, where security is probably the most critical one [5] – the traditional security techniques built into individual software packages are not good enough for interoperable services.

Corresponding Author: Yaohang Li (yaohang@ncat.edu)
Department of Computer Science
North Carolina A&T State University
1601 East Market St.,
Greensboro, NC 27411, USA

Consequently, the security requirement of distributed software poses new challenges in secure software engineering education, where the traditional techniques for enhancing software security is not sufficient to handle distributed software engineering practice in SOA paradigm. Unfortunately, teaching modern secure software engineering based on SOA is a technically difficult endeavor. A secure software engineering course typically requires a series of analysis and testing services and tools [6]. Due to the fact that most of these services and tools used in secure software engineering education are developed by different groups, each of them has extremely heterogeneous platform preferences, installation/configuration instructions, user interfaces, and usage parameters. Lacking of a common user interface, students and instructors may end up distracting and even struggling in low-level and complicated software installation, system setup, service configuration, and data manipulation while losing concentration in learning the important secure software engineering principles.

In this paper, we describe a new approach of using the workflow technology [7] in teaching secure software engineering. The workflow technology provides unified, interactive graphical interfaces for students and enables secure software engineering cases to be built without the need for low level programming or command-line interactions. Moreover, the workflow technology enables seamless integration of distributed and local services/tools to support secure software engineering practice. Four case studies using the Kepler scientific workflow system [8] are presented to show how we use our approach to author and enact workflows for secure software engineering scenarios. These case studies include using a local static analysis tool for code review, using a remote tool for code analysis, integrating local and remote tools, and implementing a web service fuzzer for penetration test. Student feedbacks on using the workflow technology in teaching secure software engineering class are also discussed.

The rest of the paper is organized as follows. Section 2 describes the workflow technology in general and its applications in education. Sections 3 and 4 illustrate several secure software engineering case studies using workflow and discuss the effectiveness of the workflow approach in our secure software engineering class, respectively. Finally, section 5 summarizes our conclusions and future directions.

ISBN 1-933510-99-4 ©2010 CISSE

II. Workflow Technology and its Applications in Education

Originally, workflow is an administrative concept from the field of managing business operation, referring to a business process that delivers services from one participant agent to another. In 1996, getting its definition from the Workflow Management Coalition, a workflow is described as [9]:

"The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."

Workflows introduce automation that enforces data validation and verification within business operations, overcomes constraints in time and space, maintains consistency in the business system, and significantly eliminates possible human errors. While workflow contributes importantly to many types of businesses, the concept extends beyond conventional business process management and is now applied more broadly in scientific computing [10], bioinformatics [11], sensor networks [12], engineering [13], image processing [14], e-commerce [15], and many other areas.

A fundamental element in a workflow is a task. A task can basically be defined by three parameters: input description, transformation (actor), and output description. The input description provides the information required to complete the task. The transformation is composed of the algorithms carried out in the task. The output is the information produced in this task which can be provided as input to the downstream tasks. Typically, a workflow is composed of various tasks performing operations of accessing a service or executing a specific function. Closely related tasks can also be organized as a sub-workflow which can be reused in composition of other workflows. In general, the structure of a workflow can be represented as a DAG (Direct Acyclic Graph) where tasks are connected by arcs. The relationship between tasks can be sequential, parallel, or selective [16].

There are a number of systems to facilitate the composition of workflows. In addition to Kepler used in this paper, the others include Triana [17], Taverna [18], Karajan [19], Gridflow [20], ScyFlow [21], and GridNexus [22]. Typically, a workflow system provides GUI layout for users to drag and drop and connect services or logic components to build a workflow. Given input and output parameters, a correctly composed workflow can be compiled to a format described by a workflow description language, such as the Web Services Flow Language (WSFL) [23], the Grid Service Flow Language (GSFL) [24], JXPL [25], or other XML-based languages. The workflow can then be executed. There are numerous examples demonstrating workflow techniques in important biology, chemistry, business, and mathematics applications. The key advantage of workflow is its capability of extricating people from many complicated and low-level system configurations and data manipulations and thereby allowing them to focus on the application development.

In addition to workflow's popularity in applications of scientific computing and e-business, educators have begun to investigate the feasibility of using workflow technology to support education practice. Van der Veen et al. [26] has found that using workflow in projects on business applications has provided added value to students. Santoro et al. [27] integrated the workflow concept to support collaborative project-based learning. Hiekata et al. [28] used a semantic web-based workflow framework to support design engineering education, which helped shorten the students' learning duration. Wilkinson and Ferner [29] used the GridNexus workflow editor to teach Grid Computing classes across universities in North Carolina. The workflow technology has also been adopted into e-learning tools to enhance undergraduate bioinformatics teaching and learning in Singapore [30]. In this paper, we propose to apply the workflow technology to secure software engineering education.

III. Secure Software Engineering Case Studies with Workflow

A. Using a Local Tool for Code Review

Code review is one of the most important phases in secure software development life cycle. Many security bugs can be discovered, identified, and eliminated in this phase. In the code review phase in secure software engineering class, students learn to examine various programs using a variety of static analysis tools. We first present a simple example of using Rough Auditing Tool for Security (RATS) [31] to perform code review using the Kepler workflow system, as shown in Figure 1. In this example, the RATS software package is installed on the local computer and the "running RATS" actor is used to execute the RATS program. A string accumulator is used to specify the location of the RATS command, options, and the target test file. The execution results will be displayed in text format as well as a visualization analysis program. Figure 2 shows an extension of the simple RATS workflow in Figure 1 for reviewing multiple test programs and visualizing the summarized test results.

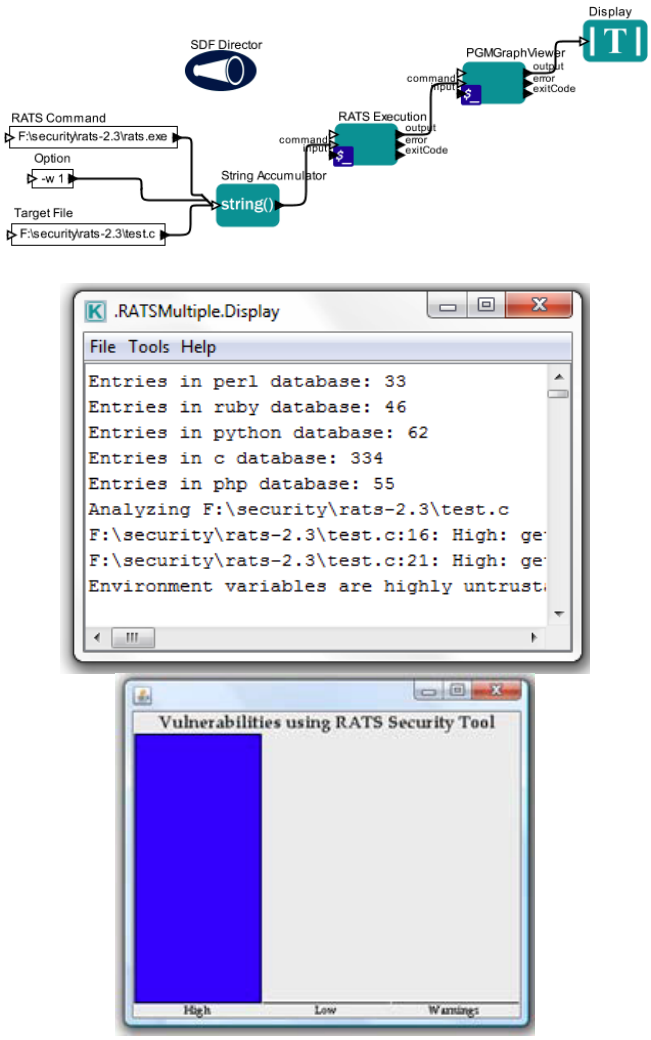

Figure 1: Kepler Workflow of Using RATS for Code Review

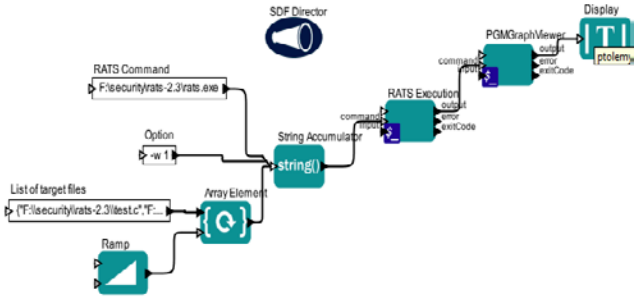

Figure 2: RATS Workflow for Reviewing Multiple Programs

B. Using Software Tools on a Remote Server

In secure software engineering education, we want to expose our students to various software analysis tools. However, these tools have heterogeneous platform, installation, and configuration requirements. One of the difficulties is that our students have to spend a lot of time in installing and setting up various software packages, which leads to loosing focus in learning the secure software engineering concepts. One practical solution is to install the software packages in a public server and allow students to connect to the server to use the software tools. Kepler provides a straightforward way to build a workflow to execute software tools on a remote server. Figure 3 shows a Kepler workflow example of using the Flawfinder package [32] installed on a remote Linux server for code review. The SSH file copier and SSH execution actor are used to upload test files to the remote server and to remotely execute the Flawfinder commands, respectively. Instead of typing various commands such as logging on to the server, uploading files, and executing commands, students manipulate the workflow and its parameters to perform code review practice.

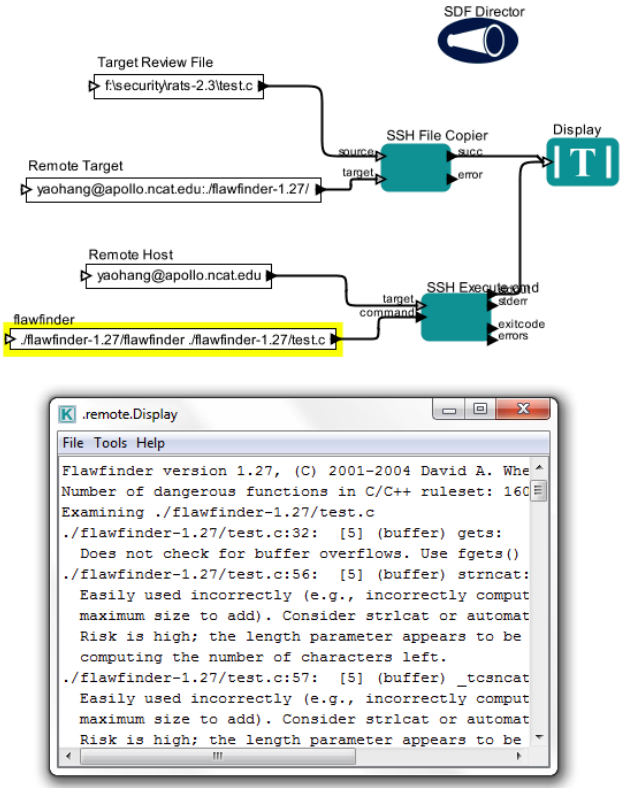

Figure 3: Workflow of using Flawfinder Installed on a Remote Computer for Code Review

C. Integrating Remote and Local Tools

One of the key strengths of the workflow technology is its capability of integrating and orchestrating remote and local services or tools. In our secure software engineering class [6], students use various tools to analyze the security vulnerabilities of code samples and compare their results.

Consider a typical student laboratory assignment of using RATS and Flawfinder to analyze vulnerability of a C++ program. Because RATS can be executed in Windows and Flawfinder can only be installed on UNIX-based platforms, using command lines to complete this laboratory assignment is rather cumbersome. A student has to install and configure these two software packages on different operating systems, run code review separately using these tools, and then collects result data for comparison.

Figure 4(c) shows the Kepler workflow with seamless integration of locally installed RATS and remotely installed Flawfinder. Modules of Flawfinder and RATS are provided as composite actors, which contains definitions of subworkflows of executing Flawfinder and RATS shown in Figures 4(a) and 4(b), respectively. Implementation details and local variables are hidden in the implementation of the composite actors. These encapsulated composite actors can be provided to the students and allow them to concentrate on building workflows and comparing analysis results. This example can be extended to incorporate and orchestrate more software tools into the workflow to implement more complicated tasks.

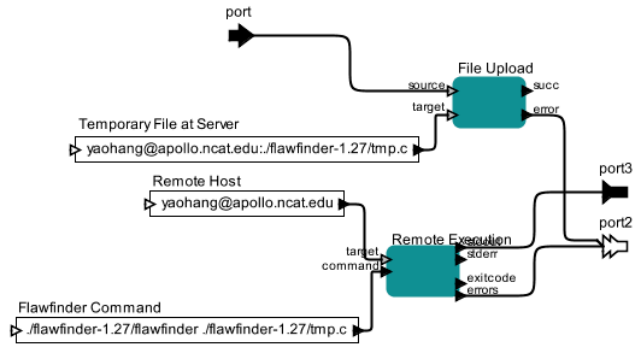

Figure 4(a): Composite Actor of Flawfinder Subworkflow

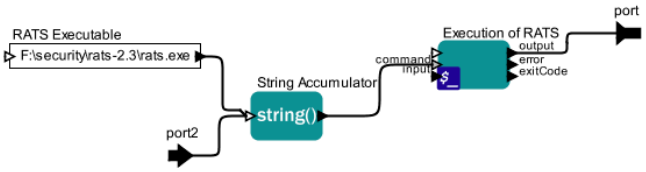

Figure 4(b): Composite Actor of RATS Subworkflow

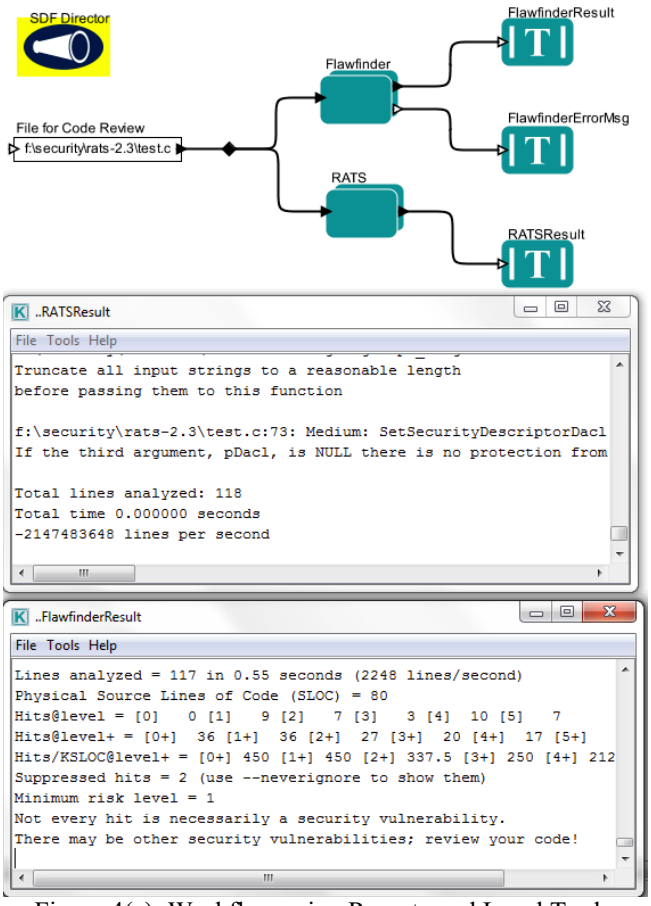

Figure 4(c): Workflow using Remote and Local Tools

D. Penetration Testing of Web Service

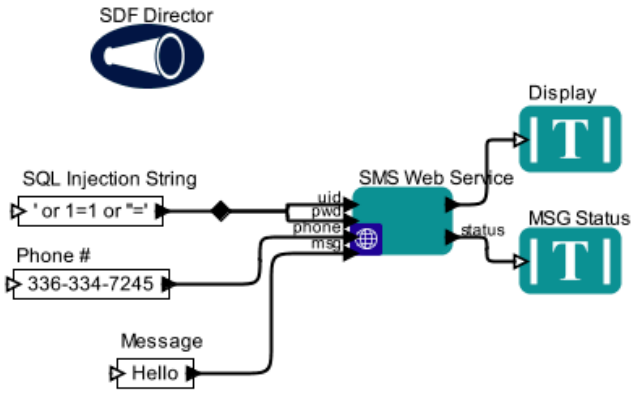

Figure 5: Workflow of Performing SQL Injection Test on a Web Service

Analyzing vulnerability of Web applications is an important topic in modern secure software engineering curriculum [33]. The workflow technology can be a useful tool to build penetration test cases for analyzing vulnerability of various Web applications and Web services. Figure 5 shows a Kepler workflow example of performing an SQL injection

attack test on a target Web service. Strings with incorrectly filtered escape characters are delivered to the username and password fields of a target web service to test whether it is vulnerable to the SQL injection attack or not. After all, workflow technology provides an intuitive way to build and illustrate penetration test cases for web applications.

The workflow technology is also an easy-to-use tool to incorporate various attack patterns [35] to Web-based applications. WSFuzzer [34] is a powerful fuzzing penetration testing tool used against HTTP SOAP based web services. In our educational practice, many students have difficulty in installing WSFuzzer due to incompatibility of the PyXML package required in WSFuzzer. In our secure software engineering class, we use Kepler workflow to implement a simple Web Service fuzzer to illustrate the mechanism of WSFuzzer, which is shown in Figure 6. Attack patterns recorded in the attack data file, which is used for WSFuzzer, are extracted to attack a target web service. After executing the workflow, students can then examine the outputs to uncover security vulnerability in a certain attack. Students can also easily manipulate the workflow and the attack data to incorporate other more sophisticated attack patterns.

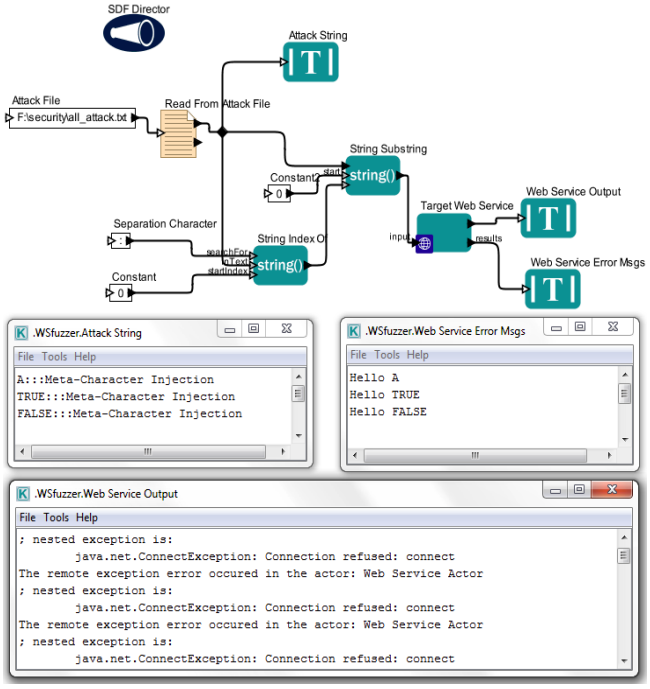

Figure 6: Workflow Implementation of a Simple Web Service Fuzzer

IV. Effectiveness Evaluation of Workflow Technology in Secure Software Engineering Class

The workflow technology was introduced to our secure software engineering course in the Spring 2010 semester. The students first finished an assignment in which they were asked to install RATS and FlawFinder on their computers, and use the tools to scan security vulnerabilities of three sample programs. The students were asked to compare the results from RATs and from FlawFinder. As a result, a number of students had great difficulty in installing and configuring these tools to run on their computers.

The students were then introduced to the four case studies using the Kepler scientific workflow system for code analysis and security testing. Afterwards, the students were asked to complete a survey on using workflow technology for teaching secure software engineering. Eleven students participated in the survey. The results of the survey are summarized as below.

The students were asked to compare the advantages and drawbacks of using workflow method and using command line methods. The students mentioned the following advantages of using workflow over using command line methods:

- Workflow makes it much easier to execute code analysis tools. The students do not need to worry about how to configure and install these tools;
- Workflow allows concurrent execution of code analysis and penetration tools;
- Workflow provides a unified GUI interface, and is more interactive;
- Workflow does not require programming. It is easy to understand and use;
- Workflow can use remote computer servers easily;
- Workflow can be used on many different systems; and
- With workflow, many testing scenarios can be created conveniently.

As to disadvantage of workflow, some are concerned that there are lots of actors in workflow which increases the learning curve. However, from the above comments, the students obviously conceived the benefits of using workflow technology for executing software security tools compared with using command line method.

The students were also asked to rate the degree of their agreement with the following statements: (1) The workflow technology is very useful for learning secure software engineering; (2) You enjoyed learning this module; (3) You are interested in learning more about the workflow for secure software engineering. The students can give a rating

from 0 to 5, while 0 represents strongly disagree, and 5 indicates strongly agree. The average ratings of the three statements are 4.5, 4.6 and 4.3 respectively. This shows on average they agree or strongly agree with the above statements. Overall, student feedback is very positive on the effectiveness of using workflow to study secure software engineering.

V. Conclusions and Future Directions

This paper describes a new approach of teaching secure software engineering using the workflow technology. The workflow technology provides unified, interactive graphical interfaces for students and enables secure software engineering cases to be built without the need for low level programming or command-line interactions. It also enables seamless integration of distributed and local services to support secure software engineering practice. Four case studies using the Kepler scientific workflow system are presented to demonstrate how to author and enact workflows for secure software engineering scenarios. These case studies include using a local static analysis tool for code review, using a remote tool for code analysis, integrating local and remote tools, and implementing a web service fuzzer for penetration tests. The workflow technology was introduced to a secure software engineering class, and the student feedback on the benefits of the workflow technology in teaching secure software engineering is very positive. The students enjoyed learning this course module and are very interested in learning more about workflow technology for secure software engineering.

Our future direction will focus on using the workflow technology to implement more sophisticated case studies for our secure software engineering curriculum. For example, we plan to implement workflows using at least some of the attack patterns described in [35] to illustrate the mechanisms of various attacks. We will also continue evaluating the effectiveness of this method in teaching secure software engineering classes.

VI. Acknowledgements

This work is partially supported by NSF under grants DUE-0737208, DUE-0737304, and DUE-0737355, and by Department of Education under grant P120A090049.

VII. References

[1] I. Foster, C. Kesselman, S. Tueske, "The Anatomy of the Grid," International Journal of Supercomputer Applications, 15(3), 2001.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," Future Generation Computer Systems, **25**: 599-616, 2009.

[3] T. Erl, "Service-Oriented Architecture: Concepts, Technology, and Design," Prentice Hall, 2005.

[4] Z. Stojanovic, A. Dahanayake, "Service-oriented software system engineering: challenges and practices," IGI Publishing, 2005.

[5] J. Epstein, S. Matsumoto, G. McGraw, "Software Security and SOA: Danger, Will Robinson!," IEEE Security and Privacy, **4**(1): 80-83, 2006.

[6] A. Frazier, S. Hudson, Y. Li, X. Yuan, "Developing Software System Security Modules," Proceedings of 12th Colloquium for Information Systems Security Education, Dallas (ISSE08), 2008.

[7] D. Georgakopoulos, M. Hornick, A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Database, **3**: 119-153, 1995.

[8] A. Ilkay, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, S. Mock, "Kepler: An Extensible System for Design and Execution of Scientific Workflows," Proceedings of the 16th Conference on Scientific and Statistical Database Management (SSDBM 2004). Santorini Island, 2004.

[9] R. Allen, "Workflow: an Introduction", Workflow Handbook, 2001, Workflow Management Coalition.

[10] M. P. Singh, M. A. Vouk, "Scientific Workflows: Scientific Computing Meets Transactional Workflows," Proceedings of NSF Workshop on Workflow and Process Automation, 1996.

[11] M. Addis, J. Ferris, M. Greenwood, P. Li, D. Marvin, T. Oinn, A. Wipat, "Experiences with e-Science Workflow Specification and Enactment in Bioinformatics," Proceedings of the UK e-Science All Hands Meeting, 2003.

[12] Y. Li, A. C. Esterline, C. Baber, K. Fuller, M. Burns, T. Hansen, T. LeFebvre, M. Schultz, M. Govett, P. Hamer, A. Mysore, "A Sensor Information Framework for Integrating and Orchestrating Distributed Sensor Services," Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, 2008.

[13] D. Vucina, Z. Lozina, I. Pehnec, "Reverse Engineering with Shape Optimization using Workflow-based Computation and Distributed Computing," Proceedings of the World Congress on Engineering, 2009.

[14] A. Radu, D. V. Gorgan, "Diagrammatic Description of Satellite Image Processing Workflow," Proceedings of International Symposium on Symbolic and Numeric Algorithm for Scientific Computing, 2007.

[15] Y. Li, Q. Cai, Y. Li, "Toward a Dynamic E-Commerce Automation with XML and Workflow Techniques on the Grid," Proceedings of IEEE SoutheastCon, 2004.

[16] J. Yu, R. Buyya, "A Taxonomy of Scientific Workflow Systems for Grid Computing," SIGMOD Record, 34(3): 44-49, 2005.

[17] I. Taylor, M. Shields, I. Wang, "Resource Management of Triana P2P Services," Grid Resource Management, Kluwer, Netherlands, 2003

[18] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver and K. Glover, M.R. Pocock, A. Wipat, P. Li, "Taverna: a tool for the composition and enactment of bioinformatics workflows," Bioinformatics, 20(17):3045-3054, Oxford University Press, London, UK, 2004.

[19] G. von Laszewski, M. Hategan, "Java CoG Kit Workflow Concepts," Journal of Grid Computing, 2006.

[20] J. Cao, S. A. Jarvis, S. Saini, G. R. Nudd. GridFlow:Workflow Management for Grid Computing. In Proceedings of 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan, 2003.

[21] K. M. McCann, M. Yarrow, A. DeVivo, P. Mehrotra, "ScyFlow: an environment for the visual specification and execution of scientific workflows," Concurrency and Computation: Practice & Experience, 18(10): 1155-1167, 2006.

[22] J. L. Brown, C. S. Ferner, T. C. Hudson, A. E. Stapleton, R. J. Vetter, T. Carland, A. Martin, J. Martin, A. Rawls, W. J. Shipman, M. Wood, "GridNexus: A Grid Services Scientific Workflow System," International Journal of Computer and Information Science, 6(2): 72-82, 2005.

[23] F. Leymann, "Web Services Flow Language (WSFL1.0)", IBM, 2001.

[24] S. Krishnan, P. Wagstrom, G. von Laszewski, "GSFL: A Workflow Framework for Grid Services," In Preprint ANL/MCS-P980-0802, Argonne National Laboratory, 2002.

[25] C. S. Hunt, C. S. Ferner, J. L. Brown, "JXPL: an XML-based scripting language for workflow execution in grid environment," Proceedings of IEEE SoutheastCon., 2005.

[26] J. van der Veen, V. Jones, B. Collis, "Using Workflow for Projects in Higher Education," Computer Science Education, **10**(3): 283-301, 2000.

[27] F. M. Santoro, M. R. S. Borges, N. Santos, "Using Workflow Concepts to Support Collaborative Project-Based Learning," Proceedings of World Conference in Educational Multimedia, Hypermedia, and Telecommunications, 2003.

[28] K. Hiekata, H. Yamato, P. Rojanakamolsan, W. Oishi, "A Framework for Design Engineering Education with Workflow-based e-Learning System," Journal of Software, 2(4): 88-95, 2007.

[29] B. Wilkinson, C. Ferner, "Towards a Top-down Approach to Teaching an Undergraduate Grid Computing Course," Proceedings of SIGCSE, 2008.

[30] S. J. Lim, A. M. Khan, M. D. Silva, K. S. Lim, Y. Hu, C. H. Tan, T. W. Tan, "The Implementation of e-learning Tools to Enhance Undergraduate Bioinformatics Teaching and Learning of Singapore," BMC Bioinformatics, 10: S12, 2009.

[31] Fortify Software RATS (Rough Auditing Tool for Security) site, http://www.fortifysoftware.com/securityresources/rats.jsp

[32] Flawfinder site, http://www.dwheeler.com/flawfinder/.

[33] J. Walden, "Integrating Web Application Security into the IT Curriculum," Proceedings of the 9th ACM SIGITE conference on Information Technology Education, 2008.

[34] WSFuzzer site. http://www.owasp.org/index.php/Category:OWASP_WSFuzzer_Project.

[35] G. Hoglund, G. McGraw, "Exploiting Software: How to break the code", Ed: Addison-Wesley.