Hakam W. Alomari

Assistant Professor

Dept. of Computer Science & Software Engineering
Miami University
Oxford, OH USA 45056
Office: 201B Benton Hall

Office: (513)-529-0356
Cell: (786)-656-6139
Email: alomarhw@miamioh.edu
URL: www.users.miamioh.edu/alomarhw/

1. Education

Ph.D.	Computer Science (Software Engineering)	Kent State University	2012
M.Sc.	Computer Science (Data Mining)	Jordan Univ. of Science & Tech.	2006
B.Sc.	Computer Science	Yarmouk University	2004

2. Dissertation

"Supporting Software Engineering via Lightweight Forward Static Slicing".

https://etd.ohiolink.edu/ap:10:0::NO:10:P10_ACCESSION_NUM:kent1341996135

3. Thesis

"ACUTE: A Spatial Data Clustering using Topological Relations Effectively".

Available from JUST Main Library, Collection: thesis, Call No: QA76.9.D343 O42 2006

4. Professional Experience

- **Assistant Professor**. Department of Computer Science & Software Engineering, Miami University, Oxford, Ohio USA. August 22, 2016 – present.
- **Visiting Assistant Professor**. Department of Computer Science & Software Engineering, Miami University, Oxford, Ohio USA. August 17, 2015 – August 21, 2016.
- **Assistant Professor**. Faculty of Information Technology, Jerash University, Jerash, Jordan. October, 2012 – May, 2015.
- **Part-time Assistant Professor**. Department of Software Engineering, Jordan University of Science & Technology, Irbid, Jordan. Fall semester 2012.
- **Graduate Teaching Assistantship**. Department of Computer Science, Kent State University, Kent, Ohio USA. 2011 – 2012.
- **Research Assistantship**. Department of Computer Science, Kent State University, Kent, Ohio USA. Supported by Professor Jonathan I. Maletic. 2009 – 2012.
- **Part-time Lecturer**. Department of Computer Science, Jordan University of Science and Technology, Irbid, Jordan. Spring 2007, Spring 2008.
- **Part-time Lecturer**. Department of Computer Information Systems, Yarmouk University, Irbid, Jordan. Spring 2007.
- **Part-time Lecturer**. Department of Computer Science, Al-Balqa Applied University, Huson, Jordan. 2006 – 2007.
- **Research Member and Software Developer**. Software Development Laboratory <SDML>[1], Kent State University, Kent, Ohio USA. 2009 – 2012.

[1] http://www.sdml.info/people.html

5. Professional Affiliations

- Institute of Electrical and Electronics Engineers (**IEEE-membership**) since 2015.

6. Research

6.1. Research Interests

Software engineering; software maintenance and evolution; program analysis, program comprehension; program slicing; reverse engineering; software measurements; software visualization to support maintenance and evolution.

6.2. Software Developed

1. *MosaiCode-Beta version*: an independent visualization front end for software analysis. The initial version was developed collaboratively with D. Mosora under supervision of Professor Jonathan I. Maletic.
2. *srcSlice*: an efficient and highly scalable slicing tool that was shown to work on a variety of C/C++ programs and language features. Available for download at www.srcML.org and open source under GPL.
3. *vizSlice*: is a web-based application[2] that implements a pipeline of tools including (1) srcML, an XML format for source code, (2) srcSlice, (3) a newly created vizSlice parser used to extract information to enable visualization of slicing, and (4) D3 visualization.

7. Publications

7.1. Conference Proceedings

1. Alomari, H. W., Jennings, R. A., Virote de Souza, P., Stephan, M., Gannod, G. C., "vizSlice: Visualizing Large Scale Software Slices". In the Proceeding of the 4th **IEEE** Working Conference on Software Visualization (VISSOFT'16) Tool Demonstrations Track, NC, Raleigh USA, Oct 3-4, 5 pages.
2. Newman, C., Sage, T., Collard, M.L., Alomari, H. W., Maletic, J.I., "srcSlice: A Tool for Efficient Static Forward Slicing", In the Proceeding of the 38th **ACM/IEEE** International Conference on Software Engineering (ICSE'16) Tool Demonstrations Track, Austin, Texas USA, May 14-22, Pp. 621-624. (**32% acceptance in Tools Track**).
3. Alomari, H. W., Amer F. Al-Badarneh, "A Topological-Based Spatial Data Clustering". In the Proceeding of **SPIE** 9845, Optical Pattern Recognition XXVII, 98450S; doi:10.1117/12.2229413, Baltimore, Maryland USA, April 17- 21, 2016.
4. Alomari, H. W., "A Slicing-Based Effort Estimation Approach for Open-Source Software Projects". Presented in the International Conference on Advances in Business Management & Information Technology (ICABMIT '15), New York, USA, June 5th, 2015.
5. Alomari, H. W., Collard, M. L., Maletic, J. I., "A Slice-Based Estimation Approach for Maintenance Effort". In the Proceedings of the **IEEE** 30th International Conference on Software Maintenance and Evolution (ICSME '14). Victoria, British Columbia, Canada, September 28 – October 3, 2014. Pp. 81 – 90. (**19% acceptance rate out of 210 full papers**).
6. Alomari, H. W., Collard, M. L., Maletic, J. I., "A Very Efficient and Scalable Forward Static Slicing Approach". In the Proceedings of the **IEEE** 19th International Working Conference on Reverse Engineering (WCRE '12), Kingston, Ontario, Canada, October 15 – 18, 2012. Pp. 425 – 434. (**25% acceptance rate out of 138 full papers**).

7.2. Journal Articles

[2] http://vizslice.csi.miamioh.edu

7. Alomari, H. W., "A Slicing-Based Effort Estimation Approach for Open-Source Software Projects" International Journal of Advanced Computational Engineering and Networking (IJACEN), DOI: IJACEN-IRAJ-DOI-2678, Vol. **3**, No. **8**, Pp. 1 – 7. August 2015. (**Impact Factor: 2.25**).
8. Alomari, H. W., Collard, M. L., Maletic, J. I., Alhindawi, N., Meqdadi, O., "srcSlice: Very Efficient and Scalable Forward Static Slicing". **Wiley** Journal of Software: Evolution and Process (JSME), DOI: 10.1002/smr.1651, Vol. **26**, No. **11**, Pp. 931 - 961. November 2014. (**Special issue on the best papers from WCRE 2012**).

7.3. Under Review, conferences & Journals (submitted)

9. Alomari, H. W., Walia, G., Zaback, K., Kiper, J., "Using WReSTT Web-Based Testing Tool in Computer Science and Software Engineering Courses". Submitted to the SIGCSE 2017, March 8th – 11th, Seattle, Washington, USA.

7.4. In Progress

10. Alomari, H. W., Stephan, M., Gannod, G. C., "A Slicing-Based Approach to Measure Semantic Change between Software Versions".
11. Alomari, H. W., Gannod, G. C., Maletic, J. I., "Is it Time to Change our Views of Program Slicing? A Comprehensive and Comparative Review of all Related Work".
12. Sayagh, M., Adams, B., Alomari, H. W., "A Very Efficient and Scalable Backward Static Slicing Approach".

8. Funding and Support

1. srcSlice is supported in part by a grant: CI-ADDO-EN: Collaborative Research: Enhancing the srcML Infrastructure: A Mixed-Language Exploration, Analysis, and Manipulation Framework to Support Software Evolution, J. Maletic (PI), M. Collard (University of Akron) $800,000 Total, $600,877 Kent State University, $199,123 U of Akron.
    a. http://www.nsf.gov/awardsearch/showAward?AWD_ID=1305217. srcML.org.
    b. It is for srcML but we talked about releasing tools such as srcSlice. And we have a current release of srcSlice under that project.
2. Graduate Research Assistantship. Under supervision of Professor Jonathan I. Maletic, Kent State University, Kent, Ohio USA. Sep 2009 – May 2010. This project is supported in part by a grant from National Science Foundation grants for visualizing large-scale software change characteristics.
3. Graduate Research Assistantship. Under supervision of Professor Jonathan I. Maletic, Kent State University, Kent, Ohio USA. Sep 2010 – May 2011. This project is supported in part by a grant from ABB Inc. for slicing large-scale software systems.
4. Graduate Research Assistantship. Under supervision of Professor Jonathan I. Maletic, Kent State University. Ohio USA. Summer 2011 & Summer 2012.
5. Graduate Teaching Assistantship. Department of Computer Science, Kent State University. Ohio USA. 2011-2012 Academic Year.

9. Departmental, University, and Professional Service

9.1. International Conferences & Workshops

1. Technical Committee Member at the 7th International Conference on Computing, Communication and Networking Technologies (7thICCCNT). July 6 – 8, **2016**. Dallas, USA.
2. LESSEP 2016 – First Workshop on Using Learning and Engagement Strategies in Software Engineering and Programming Courses. Florida International University. Florida, Miami, June 10th & 11th, **2016**.
3. ICSE 2016 – The 38th **ACM/IEEE** International Conference on Software Engineering. Austin, TX, May 14 - 22, **2016**. (Tools Track).
4. ICABMIT 2015 – International Conference on Advances in Business Management & Information Technology, New York, June 5th **2015**.
5. WISTPC-15-1 – Workshop on Integrating Software Testing into Programming Courses. Florida International

University. Florida, Miami June 12th – 13th, **2015**.

6. ICSME 2014 – The 30th **IEEE** International Conference on Software Maintenance and Evolution. Victoria, BC, Canada, Sep 28th – Oct 3rd, **2014**.

9.2. Additional Reviewer

IEEE/ACM International Conference on Automated Software Engineering (**ASE**)	2011
IEEE International Conference on Program Comprehension (**ICPC**)	2009
IEEE International Conference on Software Maintenance (**ICSM**)	2009, 2010
IEEE Working Conference on Reverse Engineering (**WCRE**)	2011
IEEE Working Conference on Mining Software Repositories (**MSR**)	2009
ACM Symposium on Software Visualization (**SOFTVIS**)	2010

9.3. Departmental and University Services

2012 – 2015 **Jerash University**

- Faculty of Information Technology Representative
- Faculty of Information Technology Curriculum Committee
- Chair for the Local Conferences and Workshops Committee. Developed content, organize, and description for the local conferences as part of a Computer Science major concentration within our graduate curriculum.
- Chair of Joint Committee (Graduate Studies and Curriculum Committees) to establish a new graduate program for 2014/2015. The committee developed and proposed a graduate curriculum. This committee assessed the current state of the department's undergraduate curriculum and how it's fit with the proposed graduate program. The proposed curriculum focuses on the support of all (and future) departmental research initiatives and facilitates the production of high quality, knowledgeable, graduate students.
- Chair of Undergraduate Committee to establish a new undergraduate program in Software Engineering. This committee responsible about developing a program curriculum, courses description, and the committee shall interpret and implement undergraduate program as approved by the faculty.
- Undergraduate – Graduation Project committee.
- Chair for the Quality Office in Jerash University.

2007/2008, 2012/2013 **Jordan University of Science and Technology**

- Exams Committee

2011/2012 **Kent State University**

- Computer Science Graduate-studies Committee – Graduate Student Representative (elected).

9.4. Course and Curriculum Development

MIAMI UNIVERSITY

Course Development & Revision

CSE 278 Computer Architecture – Spring 2016.

CSE Software Construction – Spring 2016.

CSE 381 Operating Systems Course – Fall 2015. An inclusive revision of this course was made. A detailed syllabus developed so this course better fits into the undergraduate catalog. This course focused on providing a detailed coverage of the fundamentals of operating systems and effectively uses their features through their primary C/C++ APIs. The emphasis of the course is on using OS APIs to develop software involving multiple processes and threads. In addition, the course also covers core job-control and scripting languages (such as: shell scripts) native to an OS.

CSE 274 Data Abstractions and Data Structures – Fall 2015. The content of this course was revised, standardized, and aligned with its prerequisites. New textbooks were examined and a detailed syllabus developed so this course better fits into the new undergraduate catalog. The course on data structures is one

of the critical courses in computer science. Every computer science curriculum in the world includes a course on data structures. Data structures are that important; it is usually the first course that ties together the theoretical mathematics that underpins computer science with the practical aspects of writing programs in a systematic way. The better the student understand this material, the better she will do later.

JERASH UNIVERSITY

New Course Development & Teaching

Program Comprehension Course – Developed content and description for the course as part of a Computer Science major concentration within our undergraduate curriculum. This course offered as special topics during summer semester 2013.

Course Development & Revision

Software Engineering Course – An inclusive revision of this course was made including a new catalog description. New textbook was selected and examined under my guidance that better supports our students and program, and a detailed syllabus developed so this course better fits into the new undergraduate catalog. This course focused on OO design and programming, OO software development process. Additional topics include process improvement, agile development, software metrics, effort estimation, design patterns, and frameworks. This course is a required course in the undergraduate curriculum and is chosen to be required in the proposed graduate curriculum.

Data Structures and File Processing – The content of this core course was revised, standardized, and aligned with its prerequisites. The course focused on advanced computer programming design, and development with a primary focus on data structures and abstraction using an object oriented programming language. A course laboratory was created and developed all materials, topics, and assignments for labs. Revised lecture materials to reflect lab component. http://www.cs.kent.edu/~halomari/1001220.htm

JORDAN UNIVERSITY OF SCIENCE & TECHNOLOGY

Course Development & Revision

SE 321 Software Requirements Engineering – A complete revision of this course was made including a new catalog description. New textbook was examined and a detailed syllabus developed so this course better fits into the new undergraduate catalog. The course now is focused on concepts, methods, and tools for the creation of large-scale software systems. In addition, methods, tools, notations, and validation techniques to analyze, specify, prototype, and maintain software requirements. Students participate in a group project on software requirements, specification, and object-oriented software design. Students learn to develop their projects using role-playing method and tools. This aligns with the new ACM/IEEE recommendations. http://www.cs.kent.edu/~halomari/SE321.htm.

SE 471 Client/Server Programming (PHP) – Developed content and description for the course as part of a Software Engineering major concentration within the undergraduate curriculum. The content of this course was revised and standardized to fit with its prerequisites and the follow on courses. Changes to this course were made in conjunction with the Curriculum Committee and others who teach it regularly. The course now is focused on server side technologies such as PHP. Students learn to design and implement real-world web applications. http://www.cs.kent.edu/~halomari/SE471.htm

KENT STATE UNIVERSITY

Course Development & Teaching

CS23001 Laboratory – laboratory instructor under supervision of Professor Maletic and provides a help in developed materials, topics, and assignments for new laboratory component of the course for first offering in Fall 2011.

10. Teaching

10.1. Teaching Interests

Software engineering (introductory and advanced levels), introduction to computer science and computer programming, object-oriented programming, operating systems, data abstractions and data structures.

10.2. Teaching Philosophy

The goal is to instill, in the student, a strong fundamental background of the subject, as well as problem solving strategies and abstract thinking skills. This is achieved by conveying basic knowledge of the subject and incorporating problem solving scenarios during lectures, homework, exams, and in-class assignments. Instead of asking students to regurgitate textbook material, application of learned knowledge to given problem situations is preferred. This requires students to integrate newly gained knowledge of the subject and apply it to a real problem. This also supports independent and abstract reasoning, since the student is challenged to critically analyze the problem, use their knowledge to solve the problem, and build on that knowledge for the future.

Teaching involves selecting and presenting material in a manner that is most readily understood by the student. It is up to the instructor to ensure that the material is presented in an organized and logical way. First, an important aspect of imparting knowledge during lectures involves engaging students in the process. The lines of communication between the instructor and students always need to remain open during lectures, as well as offline during office hours, and emails, which is conducive to the whole student learning experience. It is important that the instructor incorporate relevant personal experience as a past student and instructor in computer science as well as experiences from industry. Second, it is important for students to engage in collaborative learning experiences with fellow students, when possible, such as working together in teams or jointly contributing to a class project. This builds strong interpersonal skills (both written and verbal) as well as experience with teamwork that is reflective of real world software projects. Third, the instructor must demonstrate enthusiasm and interest for the subject, as students pick up on this very quickly. Students feel more confident and motivated if they believe the instructor is not only knowledgeable but also passionate about teaching the subject material. Finally, students are evaluated fairly based on the requirements set forth during the course and any concerns related to the course are respectfully and promptly addressed.

The teaching philosophy is adjusted and evolved based on the subject material being taught as well as the level of the students involved. For instance, the teaching strategy and topic focus is different between undergraduate and graduate level courses. Repetition and clarification using example scenarios are two techniques that are consistently used during lectures and more importantly if the material is harder to grasp.

In conclusion, each and every student must take an active role and ownership over their learning. This requires time, focus, effort and attention of the student. Learning can be self-directed but should also be facilitated by the course curriculum and through my support and encouragement. My purpose is to create a classroom environment where students are free to develop professionally and personally. The objective of any course is to provide student with a foundation of knowledge and skills that is readily transferable to her subsequent coursework at the university.

I work with students on a case-by-case basis but must ensure consistency and fairness to all students.

10.3. Teaching Experience (Courses taught)

10.3.1. Title/Subject | Terms/Dates | Institution

CSE 212 – Software Engineering for User Interface and User Experience Design	Fall 2016	Miami University
CSE 201 – Software Engineering	Summer 2016	Miami University
CSE 211 - Software Construction	Spring 2016	Miami University
CSE 278 - Computer Architecture	Spring 2016	Miami University
CSE 381 - Operating Systems	Fall 15	Miami University
CSE 274 - Data Abstractions & Data Structures	Fall 15	Miami University
My Average teaching evaluation for all courses taught at Miami University	**Fall 15, Spring 16, and Summer 16**	**3.05 Out of 4**
Software Engineering	Spring 15 and Fall 13	Jerash University
Data Structures and File Processing	Spring 15, 14, 13 and Fall 14, 13, 12	Jerash University

Spatial Topics in C.S. (Cloud Computing)	Summer 14	Jerash University
An Introduction to Computer & Internet	Fall 13	Jerash University
Project Management	Spring 14	Jerash University
Fundamental of Programming	Spring 14	Jerash University
Principles of E-Commerce	Spring 15, and Fall 13	Jerash University
Program Comprehension	Summer 13	Jerash University
Programming Languages	Spring 13	Jerash University
Management Information Systems	Fall 12	Jerash University
My Average teaching evaluation for all courses taught at Jerash University	**Oct. 2012 – May. 2015**	**4.33 Out of 5**
Computer Skills	Spring 14, 13, 08, and Spring 07	Jerash University, Yarmouk University, Jordan University of Science & Technology
System Requirements Engineering	Fall 12	Jordan University of Science & Technology
Client/Server programming (PHP)	Fall 12	Jordan University of Science & Technology
Selected Programming Language	Spring 08, 07	Jordan University of Science & Technology
Microprocessor Systems	Spring 06	Al-Balqa Applied University
Publishing on the Internet (HTML)	Spring 06	Al-Balqa Applied University
An Introduction to System Programming	Spring 06	Al-Balqa Applied University
Data Structures & Abstraction (Lab instructor)	Spring 12 and Fall 11	Kent State University

Research Statement

Hakam W. Alomari
Department of Computer Science & Software Engineering
Miami University
Oxford, OH USA 45056
alomarhw@miamioh.edu

Research Interests: Software engineering; software maintenance and evolution; program analysis, program understanding/comprehension; software slicing; reverse engineering; software measurements; software visualization to support maintenance and evolution.

1. Introduction

The research program focuses on developing and constructing methods for lightweight static program analysis. The objective is to develop new analysis methods that are highly scalable for applications on very large software systems (>MLOC). Clearly, there will be a trade-off of reduced accuracy. Much of my work deals with understanding the difference in accuracy and how this impacts the conclusions that can be drawn from the results. Specifically, my dissertation work dealt with the construction of a lightweight forward static slicing approach. I developed a tool, srcSlice [3] [1], [2], [3], and compared its results to a commercial slicing tool (CodeSurfer [4]). srcSlice produced a very reasonable result in comparison. When compared to a full PDG (Program Dependency Graph) approaches, srcSlice had up to four orders of magnitude increase in speed for large systems. The tool is built on top of the srcML [5] [4], [5] infrastructure that is developed at Kent State University by Professor Jonathan I. Maletic. Both tools are available to download from www.srcML.org. Instructions for installing srcML and srcSlice along with documentation are also available at this site.

A very fast and scalable, yet slightly less accurate, slicing approach is extremely useful for a number of reasons. Developers will have a very low cost and practical means to estimate the impact of a change within minutes versus hours/days. This is very important for planning the implementation of new features and understanding how a change is related to other parts of the system. It will also provide an inexpensive test to determine if a full, more expensive, analysis of the system is warranted. A fast slicing approach will open up new avenues of research in metrics and mining of histories based on slicing. That is, slicing can now be conducted on very large systems and on entire version histories in very practical time frames. A number of experiments and empirical investigations previously too costly to undertake may now be performed.

The following sections present the research topics the author is involved in.

2. A lightweight Forward Static Slicing

Program slicing is a widely used method for understanding and detecting the impact of changes to software. The idea is fairly simple; given a variable and the location of that variable in a program, find what other parts of the program are affected by this variable. Unfortunately, there are not any open-source slicing tools available that are scalable to large systems. While some free versions of commercial tools are available for academic use they are limited on the size of code base that can be analyzed. For example, the free version of CodeSurfer will not slice programs over 200KLOC.

The concept of program slicing was originally identified by Weiser [6], [7] as a debugging aid. He defined the slice as an executable program that preserved the behavior of the original program. Weiser's algorithm traces the data and control dependences by solving data-flow equations for determining the direct and indirect relevant variables and statements. Since that time, a number of different slicing techniques and tools have been proposed and implemented. These techniques are broadly distinguished according to the type of slices: static versus dynamic, closure versus executable, inter-procedural versus intra-procedural, forward versus backward, conditioned, amorphous, union, and quasi-static slicing.

Program slicing is typically based on the notion of a Program Dependence Graph (PDG) or one of its variants, e.g., a System Dependence Graph (SDG). Unfortunately, building the PDG/SDG is quite costly in terms of computational time and space. As such, slicing approaches generally do not scale well and while there are some (costly) workarounds, generating slices for a very large system can often take days of computing time. Additionally, many tools are strictly limited to an upper bound on the size of the program they can slice.

---

[3] srcSlice (sõrs slice), n. srcSlice is a very efficient and highly scalable slicing tool that was shown to work on a variety of C/C++ programs and language features. srcSlice was supported in part by a grant from the ABB Inc., http://www.abb.com/

[4] CodeSurfer is a produce of GammaTech Inc. www.gammatech.com.

[5] srcML (sõrs em el), n. srcML is an infrastructure for the analysis and manipulation of source code. In addition, it is a lightweight, highly scalable tool to convert source code into srcML. srcML is supported in part by a grant from the National Science Foundation (CNS 13-05292/05217).

The tool developed by the author, srcSlice, addresses this limitation by eliminating the time and effort needed to build the entire PDG. In short it combines a text-based approach, similar to Cordy's [8], with a lightweight static analysis infrastructure that only computes dependence information as needed (*aka* on-the-fly) while computing the slice for each variable in the program. The slicing process is performed using the srcML format for source code. The srcML format provides direct access to abstract syntactic information to support static analysis.

This work is detailed in my paper at the IEEE International Working Conference on Reverse Engineering (WCRE 2012) [1]. This paper was selected and invited to a special issue of the best papers at WCRE'12 of the Wiley Journal of Software: Evolution and Process (JSME) [2]. One of the reviewers stated that this work was an important new direction in the field of program slicing and the paper deserves very serious attention and is likely to have a far-reaching and hugely important impact on the program slicing research agenda and its applications.

A new implementation of the srcSlice tool is released as a SAX parser. The particular implementation of SAX parser is a C++ wrapper around libxml2's SAX interface called srcSAX; it was made specifically to support building tools that use srcML. Since SAX parsers store no data about previously seen tags, they are very memory and operation efficient. To take full advantage of this feature of SAX, we have worked to store as little data as possible and minimize repetition. As a result, srcSlice is very fast. On the largest system we present here (see TABLE I), the Linux Kernel, srcSlice clocks in at about 239K identifiers per minute. On Blender, srcSlice ran at 240K identifiers per minute. The Linux Kernel had about 7 times the number of identifiers as Blender, so not quite a 1:1 ratio between number of variables and size increase in code. The ratio between the speed of execution for srcSlice on Linux and Blender is negligible; it scales extremely well. A research objective is to investigate more ways to increase srcSlice's speed.

This work has been published in the 38th IEEE/ACM International Conference on Software Engineering (ICSE 2016) [3].

TABLE I. RESULTS OF SRCSLICE APPLIED TO MULTIPLE SYSTEMS WITH THE SIZE OF THE SYSTEM, THE NUMBER OF VARIABLES FOUND, AND THE EXECUTION TIME FOR THE SRCSLICE TOOL.

System	Size (LOC)	Variables	Execution Time
Linux Kernel-4.06	~13,000,000	~1,918,000	7 min
Blender-2.68	~1,300,000	~265,000	70 sec
Inkscape-0-.91	~410,000	~74,000	18 sec

3. Slicing Profile Changes Over History

This area of research focuses on leveraging the slicing approach to examine how the slicing profile of a system changes over its history. This could not be previously done due to the time constraints necessary to slice large systems (i.e., days of computing time). We start by examining how the Linux kernel has changed in the context of slicing and we generated the slice for every variable across every version of the Linux kernel (~1000 version with a total lines of code equal to ~ 4.5 billion LOC). This will allow for understanding the impact of changes from one version to the next. Additionally, a slice-based software metrics over versions history that related to the maintenance effort is to be introduced. (e.g., *sliceSize*, *hashSize*, *sCoverage*, etc.). These metrics are extracted directly from the source code without any other metadata needed.

This work has the potential for a number of interesting and publishable results.

4. A Slice-Based Estimation Approach for Maintenance Effort

This work addresses the following program maintenance concerns: build an estimation approach for maintenance effort using a slice based metrics, identify the type of maintenance being performed during the life time of the system, and finally characterize the system's evolution using Lehman's laws of software evolution. More specifically, slice-based software measures and a corresponding process that can represent maintenance effort in open-source systems are being developed. This work analyzed almost all the versions of Linux kernel, and constructed, validated, and compared three indirect maintenance-effort models.

This work is detailed in my paper at the IEEE International Conference on Software Maintenance and Evolution (ICSME 2014) [9]. Additionally, this work is extended and investigated towards open-source software projects in my journal article at the International Journal of Advanced Computational Engineering and Networking (IJACEN 2015) [10].

5. A Slicing-Based Approach to Measure Semantic Change between Software Versions

The srcSlice is used as a basis for an approach to automatically capture the semantic differences between different versions of systems. Whereas previous approaches to semantic differencing have tended to have too much of a syntactic character, this work uses dependence analysis to find (measure) not merely changed elements, but also those elements of the code that are affected by the changes. As such, the proposed approach could be thought of as a combination of impact analysis and change analysis; combining the two to understand the impact of changes as a quantity of semantic difference. Each version is represented as a collection of program slices on different levels of granularity: variable, method and file. The changes between versions are represented as the set of changed slices.

This work is submitted as a full paper to the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015) [11]. The reviewers strongly accepted the novelty of the proposed approach as a semantic differencing, however the paper is rejected due to the lack of strong evaluation and not considering threats to

validity. This paper is ready for submission now; I'll submit this paper again to ASE 2017 by May 12th, in Illinois, USA.

6. Visualizing Program Slices

This area of research seeks to understand the results of slicing very large-scale software systems. That is, understanding how a slice-based change is related to other parts of the system. The ultimate goal is to provide a suitable interface for program slicing that allows slicing at the variable, function, or file level, and provide fast visual feedback on slice structure. The investigation includes a method that support visual mining of system trends and artifact's hotspots in a software repository. The described method detects the system hotspots as system artifacts with high activity over flexible time intervals and indicated by different measures to augment the visualization method. A new visualization approach is developed at the Department of Computer Science & Software Engineering, Miami University. This project is applied over the Linux, and currently other systems are under investigation.

This work has been published in the IEEE Working Conference on Software Visualization (VISSOFT'16) [15].

7. Miscellaneous Investigations

A new article is submitted to the Transaction of Software Engineering Journal (TSE) [12] under the title: *Is it Time to Change our Views of Program Slicing? A Comprehensive and Comparative Review of all Related Work*. In this article, a new direction in the field of program slicing, based on our previous lightweight slicing approach, is presented and compared against the literature. We feel the impact of this lightweight slicing tool on how developers and researchers leverage program slicing to address various software engineering tasks could be much broader than heavyweight approaches, and have a far-reaching and hugely important impact on the program slicing research agenda and its applications.

Other current project: an approach that automatically discovers different cluster shapes that are hard to discover by traditional clustering methods (e.g., non-spherical shapes) is presented and submitted [13]. This allows discover useful knowledge by dividing the datasets into sub clusters; in which each one have similar objects. The approach does not compute the distance between objects but instead the similarity information between objects is computed as needed while using the topological relations as a new similarity measure. An efficient tool was developed to support the approach and is applied to a multiple synthetic and real datasets [14]. The results are evaluated and compared against different clustering methods using different comparison measures such as accuracy, number of parameters, time complexity, and visually inspection. The tool performs better than error-prone distance clustering methods in both the time complexity and the accuracy of the results.

8. References

[1] Alomari, H.W., Collard, M.L. and Maletic, J.I., "A Very Efficient and Scalable Forward Static Slicing Approach," Proc. IEEE International Working Conference on Reverse Engineering (WCRE'12), 2012, pp. 425-434.

[2] Alomari, H.W., Collard, M.L., Maletic, J.I., Alhindawi, N. and Meqdadi, O., "srcSlice: Very efficient and scalable forward static slicing," Journal of Software: Evolution and Process, vol. 26, no. 11, 2014, pp. 931-961; DOI 10.1002/smr.1651.

[3] Newman, C., Sage, T., Collard, M.L., Alomari, H. W., Maletic, J.I., "srcSlice: A Tool for Efficient Static Forward Slicing", In the Proceeding of the 38th ACM/IEEE International Conference on Software Engineering (ICSE'16) Tool Demonstrations Track, Austin, Texas USA, May 14-22, Pp. 621-624.

[4] Collard, M.L., Maletic, J.I. and Robinson, B.P., "A Lightweight Transformational Approach to Support Large Scale Adaptive Changes," Proc. Proceedings of the 2010 IEEE International Conference on Software Maintenance (ICSM), IEEE Computer Society, 2010, pp. 1-10.

[5] Collard, M.L., Decker, M. and Maletic, J.I., "Lightweight Transformation and Fact Extraction with the srcML Toolkit," Proc. 11th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM'11), 2011, pp. 10 pages.

[6] Weiser, M., "Program Slices: Formal, Psychological, and Practical Investigations of an Automatic Program Abstraction Method," Ph.D. Dissertation, University of Michigan, Ann Arbor, MI, USA, 1979.

[7] Weiser, M., "Program Slicing," Proc. Proceedings of the International Conference on Software Engineering (ICSE '81), 1981, pp. 439-449.

[8] Cordy, J.R., Eliot, N.L. and Robertson, M.G., "TuringTool: A User Interface to Aid in the Software Maintenance Task," IEEE Transactions on Software Engineering (TSE '90), vol. 16, no. 3, 1990, pp. 294-301.

[9] Alomari, H.W., Collard, M.L. and Maletic, J.I., "A Slice-Based Estimation Approach for Maintenance Effort," Proc. IEEE International Conference on Software Maintenance and Evolution (ICSME'14), 2014, pp. 81-90.

[10] Alomari, H.W., "A Slicing-Based Effort Estimation Approach for Open-Source Software Projects," International Journal of Advanced Computational Engineering and Networking (IJACEN), vol. 3, no. 8, 2015, pp. 1-7; DOI IJACEN-IRAJ-DOI-2678.

[11] Alomari, H.W., "A Slicing-Based Approach to Measure Semantic Change between Software Versions," (In Progress).

[12] Alomari, H.W., "Is it Time to Change our Views of Program Slicing? A Comprehensive and Comparative Review of Related Work, Submitted:"IEEE Transaction on Software Engineering.

[13] Alomari, H.W. and Al-Badarneh, A.F., "A Topological-Based Spatial Data Clustering, Submitted:"Proc. SPIE Defense, Security and Sensing, 2016.

[14] Alomari, H.W. and Al-Badarneh, A.F., "ACUTE: A Spatial Data Clustering using Topological Relations, Submitted:" International Journal of Geographical Information Science.

[15] Alomari, H. W., Jennings, R. A., Virote de Souza, P., Stephan, M., Gannod, G. C., "vizSlice: Visualizing Large Scale Software Slices". In the Proceeding of the 4th IEEE Working Conference on Software Visualization (VISSOFT'16) Tool Demonstrations Track, NC, Raleigh USA, Oct 3-4, 5 pages.

9. Submitted Papers While in Miami University

Alomari, H. W., Jennings, R. A., Virote de Souza, P., Stephan, M., Gannod, G. C., "vizSlice: Visualizing Large Scale Software Slices". In the Proceeding of the 4th **IEEE** Working Conference on Software Visualization (VISSOFT'16) Tool Demonstrations Track, NC, Raleigh USA, Oct 3-4, 5 pages

Newman, C.D., Sage, T., Collard, M.L., **Alomari**, H.W. and Maletic, J.I., "srcSlice: A Tool for Efficient Static Forward Slicing" In the Proceeding of the 38th **ACM/IEEE** International Conference on Software Engineering (ICSE16).

Alomari, H. W., Amer F. Al-Badarneh, "A Topological-Based Spatial Data Clustering". In the Proceeding of SPIE 9845, Optical Pattern Recognition XXVII, 98450S; doi:10.1117/12.2229413, Baltimore, Maryland USA, April 17- 21, 2016.

Alomari, H.W., "A Slicing-Based Approach to Measure Semantic Change between Software Versions". (Ready for submission to the **ASE** in April, 2016).

Alomari, H.W., "Is it Time to Change our Views of Program Slicing? A Comprehensive and Comparative Review of Related Work" IEEE Transaction on Software Engineering (**TSE**).

Alomari, H. W., Walia, G., Zaback, K., Kiper, J., "Using WReSTT Web-Based Testing Tool in Computer Science and Software Engineering Courses". Submitted to the SIGCSE 2017, March 8th – 11th, Seattle, Washington, USA.