Using Concept Maps to Introduce Software Security Assurance Cases

Dallas Snider, University of West Florida
John Coffey, University of West Florida
Thomas Reichherzer, University of West Florida
Norman Wilde, University of West Florida
Chris Terry, Comnet Marketing Group
Joe Vandeville, Northrop Grumman
Allison Heinen, Northrop Grumman
Sarah Pramanik, Northrop Grumman

Abstract. To improve the security of software systems, we need to improve the software development processes used to produce them. Software security assurance cases have been proposed as a way of establishing security properties of software at different phases of the software development lifecycle; however, these assurance cases are difficult to write, communicate and introduce into an already burdened software development process. We evaluated a team-based, knowledge engineering approach to introduce software security assurance cases to neophytes through the utilization of concept maps. This approach allowed the study's participants to engage in conversations with security experts about security requirements for their software and with knowledge engineers to construct concept maps demonstrating how their software met the requirements. Our survey results and feedback show great promise for our method to be effective and efficient for disseminating knowledge about software security to new hires and students which in turn would make them cognizant of the security requirements in their organization.

Introduction

Enhancing software security is, in part, about software process improvement. To improve the security of software systems we need to improve the software development processes used to produce them. Software security assurance cases have been proposed as a way of establishing security properties of software at several security touchpoints in the development process [1]. These touchpoints are defined as "lightweight software security best practice activities that are applied to various software artifacts such as requirements and code" [2]. Assurance cases are structured and reviewable artifacts to demonstrate that a system possesses required security properties. However assurance cases are not easy to write, communicate, or introduce into an already burdened software development process.

So how can we introduce software security assurance cases to a development organization or to a new hire in such an organization? In this paper we propose a team-based, knowledge engineering approach that introduces assurance cases through the use of concept maps [3]. A concept map provides a simple visual representation for the capture and communication of technical knowledge about a particular domain.

The knowledge engineering approach we employed combines initial skeleton concept maps that describe known types of software vulnerabilities, concept maps of the software structure derived by parsing code or interface documents from a particular system, an interview process in which a knowledge engineer facilitates a conversation between a security expert and a programmer to develop the concept map-based assurance case for that system, and a review process in which the resulting assurance case is presented to stakeholders.

If successful, this approach could have several advantages. First, the knowledge engineering approach could facilitate the introduction of assurance case touchpoints into the development process. Second, participation in the interviews could improve programmer sensitivity to potential software vulnerabilities and expedite knowledge transfer. Furthermore, the concept maps supporting the assurance cases could be linked to other documentation such as design rationale to provide an integrated view of the software structure. Finally, the visual nature of assurance cases enhanced with concept maps could facilitate communication with diverse project stakeholders.

In this paper we provide a discussion of literature pertaining to software security assurance cases and concept mapping. We then discuss two small-scale case studies that were conducted to gain feedback on the practicality of a knowledge engineering approach to the construction of assurance cases containing concept maps. We summarize results of these studies including the results of questionnaires that address the utility of this approach and conclude with a discussion of lessons learned.

Software Security Assurance Case Development

Security assurance is a multidimensional concern. Evaluation of security requires evaluating targets (systems that may be attacked), processes (used to develop the targets), and remediation (corrective action to mitigate vulnerabilities) [4]. Software security assurance cases fall at the juncture of these three concerns; they are developed for a specific target as part of its software engineering process, and they may identify vulnerabilities for remediation.

An assurance case is a body of evidence that is analogous to a case presented in legal proceedings and is basically an argument assuring that some claim about a system holds. Assurance cases are structured and reviewable artifacts used to document that a system possesses required properties such as security, safety and reliability.

Software security assurance cases are prepared at defined security touchpoints during the software development lifecycle. Throughout this lifecycle, the software security assurance case evolves as it is reviewed by stakeholders with varying areas of expertise. Besides security experts, these stakeholders can include developers, testers, installers, system administrators and customers [5].

A proven software security assurance case is reusable for new versions of an existing system and may be adaptable for a completely new system. The software security assurance case creates a structure for the analysis of changes to a system and helps to ensure that changes do not have adverse effects on security by introducing new vulnerabilities. A properly created

assurance case will take into consideration the personnel, processes and technology involved in the development and delivery of software. Arguments based on personnel quality might be based on an employee's training and experience. Process arguments could be based on established testing procedures, bug tracking, code management and release schedules. Technology arguments could cover properties of hardware, operating systems and integrated development environments [6].

One of the challenges in creating a software security assurance case is the large number and diverse nature of the security requirements. These requirements are often introduced by citing large external checklists and standards such as the U.S. DoD's Application Security and Development Checklist [7] and 8500.1 Information Assurance Implementation [8] along with the U.S. Department of Commerce's SP800-53 Recommended Security Controls for Federal Information Systems and Organizations [9]. Not following these requirements correctly or in a timely manner can cost organizations millions of dollars in extra costs, damages or lost business. In addition to being lengthy, these requirements documents are subject to interpretation, which might require the opinion from someone in an organization's legal department. Requirements documents are sometimes conflicting which leads to the question of which document takes precedence. The result is a lengthy security assurance case document to address this myriad of requirements. These assurance cases are expensive to produce and are also subject to interpretation [10].

Another challenge is that a software security assurance case often requires a combination of scarce knowledge from different experts. A development organization may have security experts with deep knowledge of relevant kinds of security vulnerabilities. It will also have software developers with deep knowledge of a particular system. But these will not normally be the same people. Further exacerbating this problem is the fact that security requirements are constantly changing due to new threats and exposed vulnerabilities. Also contributing to the problem are personnel changes in information security departments and software development teams resulting in scarce knowledge of security and source code walking out the door.

One of the goals of this work is to determine if techniques from the discipline of knowledge engineering could help, at least to "break the ice" when an organization is adopting assurance cases for the first time, or when a new employee is being introduced to security assurance.

Knowledge Capture With Concept Maps

Concept maps provide a simple visual representation of the concepts in a particular domain and of the relationships between them. (See Figure 1 for a concept map pertaining to knowledge management.) Concept mapping has been used extensively to capture domain knowledge for purposes of knowledge preservation, consensus finding, and subsequent construction of formal knowledge representations for automated reasoning. Concept maps are well established in a wide variety of domains and experience levels and they are excellent for communicating structured knowledge [11].

Concept maps can be created by individuals or groups at levels of capability that extend from elementary school children to subject matter experts (SMEs). Concept mapping for knowledge capture

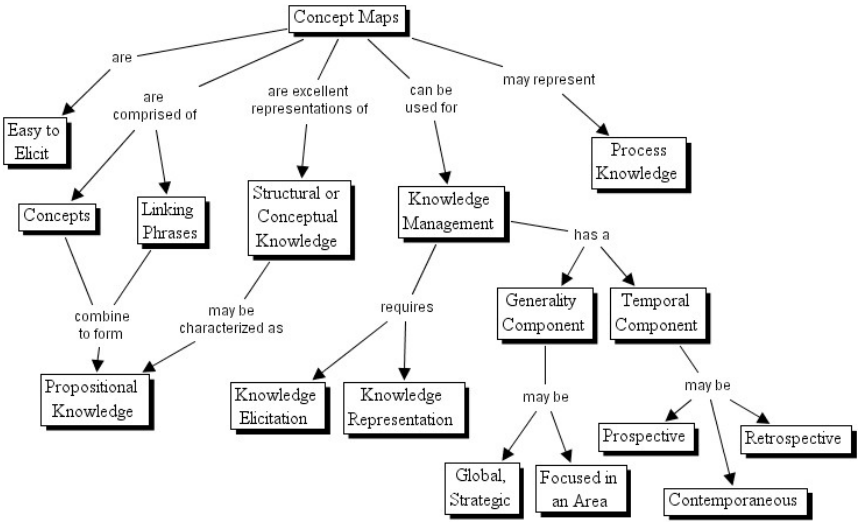


Figure 1. A concept map about concept maps used for knowledge management.

often involves more than one participant: one who is an expert in the knowledge domain, and another who is skilled at concept mapping. The person who is skilled at eliciting and representing knowledge from experts, the knowledge engineer, creates the concept map during an interview process with the SME. Mature, open-source applications for concept map development such as the Institute for Human and Machine Cognition's CmapTools are available for use [12]. CmapTools allows the knowledge engineer to populate hierarchical groups of maps and to provide drill-through links to other electronic resources such as source code and configuration files [13].

For software security assurance cases, there are typically at least two kinds of knowledge that must be brought to bear: expertise about security vulnerabilities and expertise about the specific software system under consideration. Accordingly, we suggest a team of three for the interview process as shown in Figure 2. This approach to the development of an assurance case involves the security expert interviewing the developer with questions pertaining to specific security issues, while the knowledge engineer captures the key ideas in the discussion in concept maps. The interview process captures the specialized knowledge of the security expert and allows this knowledge to be transferred to new hires verbally while creating a document that will capture this specialized knowledge for posterity. The concept maps can be augmented with other resources such as program code or other documentation that lend support to the emerging case such as testing protocols, test results and snippets of source code. Issues that need to be addressed can be represented along with the evidence for issues that have been properly addressed.

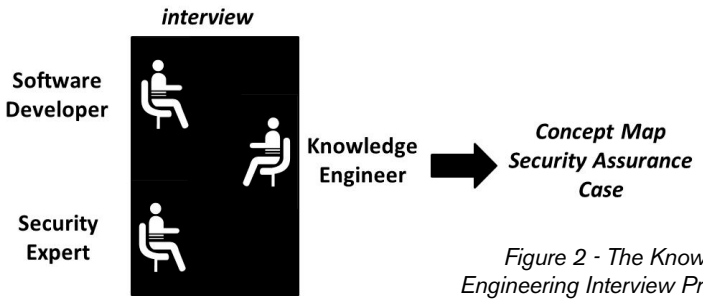


Figure 2 - The Knowledge Engineering Interview Process

Case Studies

In this section, we describe two case studies that were performed to test the potential of the approach to developing software security assurance cases. The two studies were both small-scale exercises to provide a sort of "sanity check" on using concept maps for this purpose. We wanted to see if early-career developers could create and review such cases. Further we wished to gather feedback from participants on how well such cases communicated the basic idea of an assurance case, relevant knowledge pertaining to the case, and any difficulties in the process.

The first study was academic and performed in the context of a project course for Masters of Science students in Software Engineering at the University of West Florida. Participants in the study were mature students, with some professional experience either in programming or system administration. However none had any particular background in software security. All activities were conducted online since the students were distributed geographically. The main theme of the course was the development or maintenance of services to enhance an existing Services Oriented Architecture application. Their development process included steps for:

1. **Preliminary design**
2. **Design Phase Software Security Assurance Case (touchpoint)**
3. **Design Phase Software Security Assurance Case Review, moderated by a student Software Quality Assurance (SQA) leader**
4. **Implementation and Deployment**
5. **Deployment Phase Software Security Assurance Case (touchpoint)**
6. **Deployment Phase Assurance Case Review, moderated by the student SQA leader**

These steps are shown in Figure 3.

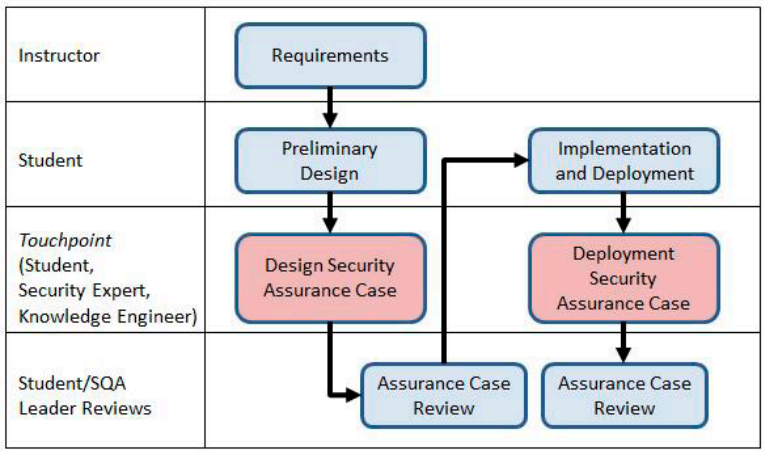


Figure 3 – Student Software Security Assurance Case Process

To develop the assurance cases for their services, the students were given four security requirements drawn from the literature [4]:

1. **Service is not subject to error handling vulnerabilities**
2. **Service supports the creation of transaction logs for access and changes to data**
3. **Service does not contain embedded authentication data**
4. **Service executes with no more privileges than necessary for proper operation.**

Participants were given an initial skeleton concept map presenting the four software security vulnerabilities as shown in Figure 4 to jump-start the knowledge elicitation process. Two knowledge engineering sessions were conducted, one at each touchpoint. During both of these sessions, the participant was interviewed by a security expert responsible for asking questions related to the security requirements and a knowledge engineer responsible for capturing knowledge through the construction of the concept map. For the post-design interview, the knowledge engineer started with the skeleton concept map and added concepts and notes for follow-up items for identified deficiencies in the design. During the post-implementation interview, the knowledge engineer added concepts, code samples and interface documents that would help to make the case that the code was secure. Following both interviews, the participants presented their concept map assurance cases for review by their peers.

Figure 5 is a screen shot showing a portion of one of the resulting assurance cases. This figure shows the analysis for the transaction logging security requirement mentioned earlier. This student did a PHP implementation of his service. He has chosen to use a combination of the Apache web server log, a text file log and a MySQL log to meet the requirement. The icon in the "logEvent()" box indicates that there is a sample of the code that can be inspected to confirm that all service operations actually call the logging function.

The second case study was done in a one-day on-site visit at a Northrop Grumman facility in Melbourne, Florida. The objective of the study was to see if participants could learn and apply concept mapping for the development of assurance cases in this short period. A total of eight participants took part in this study, four interns and four recent hires. Participants in the study had backgrounds in computer science and software development. They were familiar with graphical representations of conceptual knowledge such as UML class diagrams. Participants were provided with an introduction to concept mapping, knowledge modeling, and CmapTools. Following this introduction, the participants created concept maps on topics of their choosing to gain experience with concept mapping and with the CmapTools software.

Participants were then presented with information regarding how concept mapping and knowledge modeling might be used to build assurance cases. Participants then worked in teams to build concept maps pertaining to assurance cases. After each round of concept mapping, participants showed the maps they had created to the group with the goal of assessing strengths and weaknesses in the maps, and developing suggestions for their improvement. At the end of both case studies, each participant was asked to complete a survey with the five Likert statements shown in Table 1 and to provide comments to support their responses to the Likert statements.

Results and Discussion

The Likert statements were based on a 6-point scale with 1 representing strongly agree and 6 representing strongly disagree. Table 1 combines the results for the two groups.

For the first statement, some of the comments from the participants focused on how concept maps made the security

requirements and the design goals easier to understand. Other comments mentioned how the process helped the participants to learn the importance of secure software and to identify weaknesses in their designs along with understanding the design goal.

With the second statement, the feedback provided mentioned the process giving the participants an additional understanding of the vulnerabilities when looking for ways and methods to develop the evidence needed to prove the vulnerability had been addressed. Another comment that reflected the lower mean score for the second statement mentioned the knowledge engineering process increased their awareness, but did not provide an in-depth understanding.

For the third statement, participants commented that the CmapTools application was a great visual aid and the concept maps made their peer review presentation easier. Furthermore, the ability to drill down to the source code, configuration files and design documents was beneficial.

For the fourth statement, the participants mentioned that the collaboration during the interview sessions helped in the sharing of ideas and broadened the scope of available knowledge. Regarding the concept maps, the participants stated that the visual primitives on the maps allowed for comprehension of the vulnerabilities and the maps ensure that all the components of the cases are addressed and the relationships between the components are documented as well.

Out of all of the statements, the final statement had the least amount of agreement and the greatest dispersion of responses. Multiple responses focused on the need to continue to seek out assistance to ensure the system followed security requirements. Other comments stated how the process helped to instill confidence in the participants' ability to build secure software. While our process might not have provided the neophyte with the in-depth knowledge required to build a complete assurance case, the process is an important first step in becoming cognizant of an organizations software security requirements.

Conclusions

This paper has concentrated on the use of knowledge engineering in "first time" experiences with software security assurance cases. In the two studies described here, a total of 12 participants were introduced to concept mapping and its application to the development and presentation of software security assurance cases. Though the sample size is relatively small, we believe that the results are encouraging. Results from surveys administered to all participants indicate that knowledge engineering and concept maps could have a useful role in "breaking the ice" for software security assurance case practices and raising the level of security competence.

The approach of introducing a new developer to the idea of software security assurance cases through collaborations with a security expert and a knowledge engineer who captured the proceedings in real time, worked well. The primary additional cost of this approach is the time of the knowledge engineer in the interviews. Each interview lasted only an hour, or roughly 15 minutes for each security checklist item. The resulting concept maps were evaluated by participants to be an excellent communication tool for reviews and other subsequent uses. The

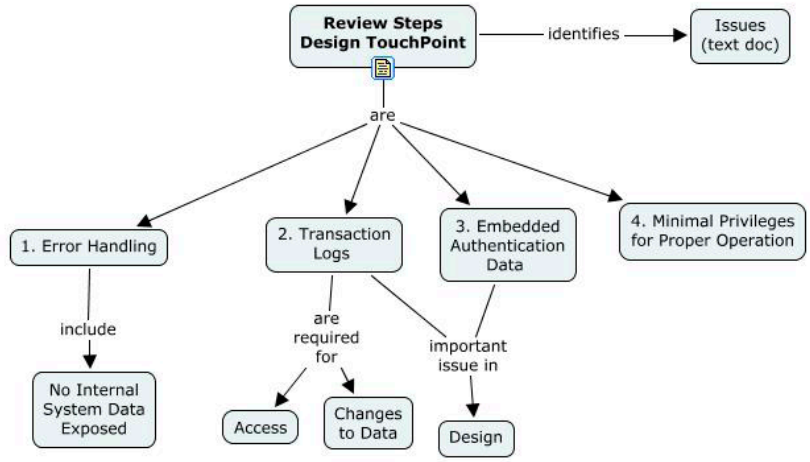


Figure 4. The initial skeleton concept map showing the four security vulnerabilities.

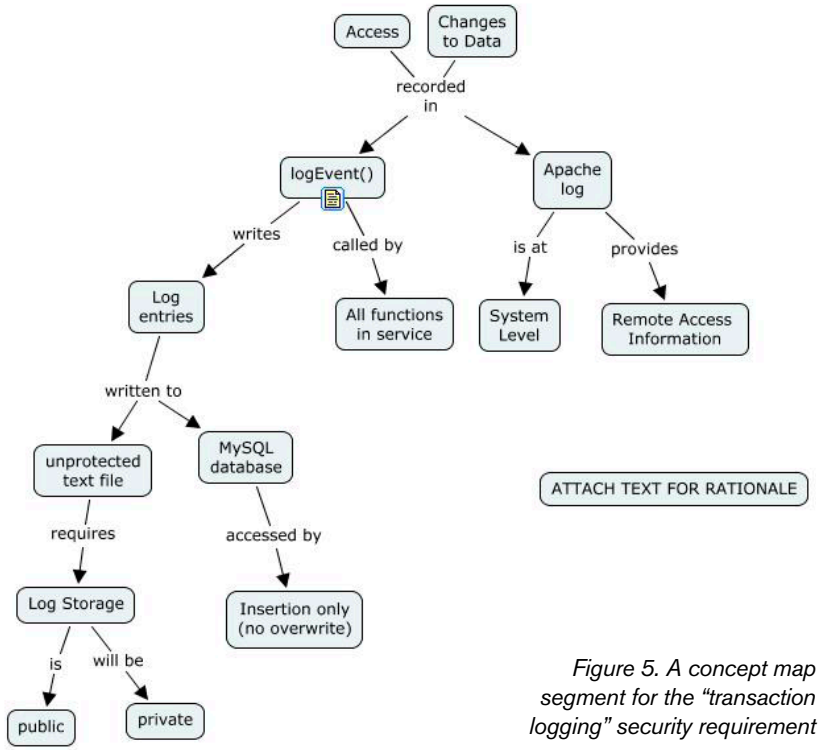


Figure 5. A concept map segment for the "transaction logging" security requirement

Statement Number	Statement 1 = *strongly agree*, 6 = *strongly disagree*	μ	σ
1.	On finishing the team-based process in this course, I feel that I understand how security assurance cases may be prepared and used as part of the software development process.	1.67	0.62
2.	On finishing the team-based process in this course, I feel that I understand the four software vulnerabilities and how to mitigate them.	2.00	1.08
3	The concept map security assurance case was easy to present in the peer review session.	1.50	0.76
4.	If in the future I need to prepare security assurance cases again, I would like to use this same team-based process.	1.42	0.49
5.	If for some future system I need to prepare a concept map security assurance case for these same vulnerabilities, I am confident I could do so with less assistance from the security expert and knowledge engineer.	2.58	1.26

Table 1 - Sample mean and standard deviation results from post-survey (N=12)

knowledge represented in the concept maps provided clarity in understanding software vulnerabilities and how to design and implement a system to handle these vulnerabilities. We found the participants adept at understanding the graphical primitives that constituted the concept map. One observation from our study at Northrop Grumman was the participants quick mastering of the creation of concept maps and the understanding of the graphical primitives because of their familiarity with linked graphical representations such as UML class diagrams. Furthermore, the concept maps helped the participants to share knowledge among their peers and supervisors.

In discussions with our collaborators at Northrop Grumman, we determined several additional potential benefits of using our knowledge engineering process to create concept maps for software security assurance cases. First, our process could help streamline the creation of testing abuse cases and unit tests for security assertions. Second, the process can assist with getting stakeholders on the same page in regards to interpreting security requirements. Furthermore, concept maps could help to communicate to all levels of management the cost implications of security requirements which would help to strike a balance between security and affordability, thus contributing to reducing overall program costs.

Acknowledgments

Funding for this project was provided by Northrop Grumman through the Security and Software Engineering Research Center and by the University of West Florida Foundation under the Nystul Eminent Scholar Endowment. We would also like to thank the students and interns who participated in the study, particularly Ms. Heather Dennison the SQA leader.

REFERENCES

1. Allen, J. H., et al. Software Security Engineering: A Guide for Project Managers. Addison-Wesley, 2008.
2. McGraw, G. Software Security: Building Security In. Boston: Addison Wesley, 2006.
3. Novak, J. D. and D. B. Godwin. Learning How to Learn. Cambridge University Press, 1984.
4. Bayuk, J. and A. Mostashari. "Measuring Systems Security." Systems Engineering 16.1 (2013): 1-14.
5. Vivas, J., I. Agudo and J. Lopez. "A Methodology for Security Assurance-Driven System Development." Requirements Engineering 16.1 (2010): 55-73.
6. Weinstock, C. B., H. F. Lipson and J. Goodenough. Arguing Security - Creating Security Assurance Cases. Jul. 2013. Aug. 2013 <https://buildsecurityin.us-cert.gov/articles/knowledge/assurance-cases/arguing-security---creating-security-assurance-cases>.
7. U.S. Department of Defense. Application and Security and Development Checklist, Version 3, Release 8 Jul. 2014.
8. U.S. Department of Defense. Instruction 8500.1 - Cybersecurity Implementation Mar. 2014.
9. U.S. Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 800-53, Recommended Security Controls for Federal Information Systems and Organizations, revision 4. Jan 2014.
10. Pramanik, S. "Increasing Security for Regulatory Constrained Systems, Through New Systems and Architecture Methodologies." Ph.D. Dissertation, C.S., U.C.C.S., Colorado Springs, CO, 2013.
11. Coffey, J. W. and T. Eskridge. "Case Studies of Knowledge Modeling for Knowledge Preservation and Sharing in the U.S. Nuclear Power Industry ." Journal of Information and Knowledge Management 7.3 (2008): 173-185.
12. Institute for Human and Machine Cognition. CmapTools Home Page. n.d. 31 Oct. 2013 <http://cmap.ihmc.us>.
13. Cañas, A. J., et al. "CMAPTOOLS: A Knowledge Modeling and Sharing Environment." Proc. of the First Int. Conf. on Concept Mapping. Pamplona, 2004.

ABOUT THE AUTHORS

Dallas Snider is an Assistant Professor of Computer Science at the University of West Florida. He received his Ph.D. in Integrated Computing and M.S. in Instrumental Sciences from the University of Arkansas at Little Rock. He received a B.A. in Physics from Hendrix College. Dr. Snider's teaching and research interests include data mining, data warehousing and cybersecurity.

Phone: 850-473-7348
E-mail: dsnider@uwf.edu

John Coffey holds an Ed.D. with an emphasis in Computer Science from the University of West Florida (UWF). He was one of the first members and ongoing collaborator with the Institute for Human and Machine Cognition. He is currently a professor in the UWF Computer Science Department and has over 100 publications. His research interests include advanced education technology, knowledge elicitation and representation, student modeling, web services and Service Oriented Architecture.

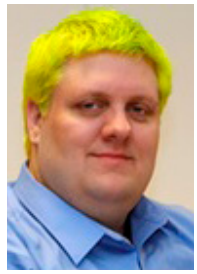
Phone: 850-474-3183
E-mail: jcoffey@uwf.edu

Dr. Thomas Reichherzer is an Assistant Professor in the Computer Science Department at the University of West Florida. He received a Ph.D. in Computer Science at Indiana University in 2009. Dr. Reichherzer's interest include knowledge representation, the Semantic Web, information retrieval, management, and visualization. More recently, he focused on sentiment analysis of unstructured text and AI approaches to smart home environments.

Phone: 850-474-2548
E-mail: treichherzer@uwf.edu

Norman Wilde received his Ph.D. from MIT in 1971. He spent a number of years working in developing countries overseas: in academic positions, with the World Health Organization, and as an independent systems consultant. Dr. Wilde is currently Nystul Chair and Professor of Computer Science at the University of West Florida. He has worked extensively with the Security and Software Engineering Research Center (S2ERC) on research projects in Software Maintenance and Software Security.

Phone: 850-474-2548
E-mail: nwilde@uwf.edu



Chris Terry has 14 years of experience working in the Information Technology industry and is currently the Senior Systems Engineer with Comnet Marketing Group, a national marketing and fundraising firm specializing in nonprofit organizations. As a software engineering student at the University of West Florida he won national awards for competitive programming and security analysis. He is currently overseeing a companywide virtualization and services orientated migration project and supervises day-to-day IT security operations.

Phone: 541-864-1470
E-mail: chris.terry@comnetmarketing.com

Joe Vandeville is an Embedded Software Engineer at Northrop Grumman Aerospace Systems, working in the area of system architecture. He has been associated with the development of software intensive systems for over 40 years, and has been involved in hardware and software development in both commercial and military systems. He has Masters Degrees in both Computer Science and Systems Engineering and is currently pursuing a PhD in Systems Engineering.

Phone: 321-951-5287
E-mail: joseph.vandeville@ngc.com

Allison Heinen is a member of the Northrop Grumman Aerospace Systems Software Products organization located in Melbourne, Florida. She has over 30 years of experience in Software Engineering and has held many leadership roles managing functional organizations, technical projects and process related initiatives. She is currently the Software Lead for process and tool related initiatives. Ms. Heinen holds a Bachelor of Science degree as well as two Master of Science degrees.

Phone: 321-726-7657
E-mail: allison.heinen@ngc.com

Sarah Pramanik is the lead for software information assurance (IA) on her current program. She is responsible for leading the development, integration and execution of IA activities. Dr. Pramanik has worked in multiple areas of cybersecurity, including vulnerability and penetration testing, information system security engineering, cybersecurity training and security architecting. She holds a Bachelors and Masters degree in Computer Science and a Ph.D. of Engineering in Security from The University of Colorado, Colorado Springs.

Phone: 516-346-7737
E-mail: Sarah.Pramanik@ngc.com