Methodology for Maintainability-Based Risk Assessment

Walid M. Abdelmoez, West Virginia University
Katerina Goseva-Popstojanova, Ph. D., West Virginia University
Hany H. Ammar, Ph. D., West Virginia University

Key Words: Maintainability-based risk, software maintenance, software architectures, change propagation probability.

SUMMARY & CONCLUSIONS

A software product spends more than 65% of its lifecycle in maintenance. Software systems with good maintainability can be easily modified to fix faults or to adapt to changing environment. We define maintainability-based risk as a product of two factors: the probability of performing maintenance tasks and the impact of performing these tasks. In this paper, we present a methodology for assessing maintainability-based risk to account for changes in the system requirements. The proposed methodology depends on the architectural artifacts and their evolution through the life cycle of the system. We illustrate the methodology on a case study using UML models.

1. *INTRODUCTION*

The cost of software maintenance accounts for 60% - 80% of the overall software system cost [1]. Software maintenance is categorized as corrective, dealing with error corrections, perfective, trying to improve the quality of the system, and adaptive maintenance concerned with system changes as requirements and environment change. Software maintainers usually are not involved in the original software development cycle. They must learn how a program functions before they can change it. They often interact with complex and difficult to comprehend systems. The status of system documentation, programmer skill, and experience and the attributes of the system itself are some of the variables that affect the maintenance process.

Good maintainability of the system facilitates easy modifications when adapting to changes in the environment. Maintainability-based risk assessment should guide the management during the software maintenance process. In this paper, we address the problem of maintainability risk assessment using architecture metrics. In accordance with NASA-STD-8719 standard [2], we define maintainability-based risk assessment of a component as the product of the probability of changing the component and the maintenance impact of changing the component [3]. Maintainability-based risk can be used to achieve the following:

- Improve the maintainability of the system architecture.
- Identify risky components in terms of maintainability and assign them to the most experienced maintainers.
- Use maintainability risk to manage system maintenance process.

Many types of risk are ushered when software systems undergo maintenance. They are similar to those we face when developing new software systems, but with different level of risk. These types of risk are [4]:

- *Project risk* — Maintenance project cannot be carried out within the budget or on time due to ineffective maintenance process or lack of personnel and maintenance capabilities.
- *Usability risk* — Systems will cause problems and failures after the maintenance is conducted. Usability risk includes functionality, performance, financial and software failure risk.
- *Maintainability risk* —It will be difficult to maintain the system in the future because of the way we conducted this maintenance.

In this paper, we propose a methodology for estimating the maintainability-based risk of system components due to changes in system requirements. This study is a part of a wider effort that considers other architectural level risks such as reliability-based risk [5] and performance-based risk [6].

The paper is organized as follows. In Section 2, we briefly discuss the literature background of our study. In Section 3, we present the estimation methodology for maintainability-based risk due to changes in requirements. In Section 4, we illustrate how to estimate maintainability-based risk on a case study.

2. *BACKGROUND*

Several studies addressed the quantification of hardware maintainability but only few attempted to quantify software maintainability. One of the famous studies [7] introduced the Maintainability Index (MI) measure. The MI is calculated using a polynomial of widely used code level measures such as Halstead measures and McCabe's cyclomatic complexity. In [8], Muthanna et al. conducted a similar study, which used design level metrics to statistically estimate the maintainability of software systems. They constructed a linear model based on a minimal set of design level software metrics to predict Software Maintainability Index.

The estimation procedure of maintainability-based risk presented in this paper builds on our previous work on change propagation probabilities [9] and size of change. To estimate these metrics, we first analyze the architecture of the system under investigation using a structural diagram or a class

diagram. From these artifacts, we identify the components and the connectors of the component-based system architecture. Then, we need to analyze message protocols between every pair of components in the system to identify messages exchanged between components C_i and C_j. With the help of case tools, we get message sets for any pair of components in the system. This information can also be obtained from static analysis tools of the source code.

In the following, we briefly present the fundamentals of estimating the change propagation probabilities. Let us consider a software architecture modeled by components and connectors. We are interested in the maintainability of the products instantiated from it. Each element of the *Change propagation probability* $CP= [cp_{ij}]$ for an architecture is the conditional probability that a change originating in component C_i requires changes to be made to component C_j [9]. An architecture can be seen as a collection of components C_i, i=1,…,N. With every component C_i, we associate the set V_i of the interface elements of the provided functions of C_i. We determine the *usage coefficient* value π_v^{ij} for every interface element $v \in V_i$ and every other component C_j, $j \neq i$. They take binary values:

- π_v^{ij} =1, if the interface element v provided by C_i is required by C_j. This means that any signature change in component C_i associated with interface element v will propagate to component C_j.
- π_v^{ij} =0, otherwise.

For every pair of components C_i and C_j, i≠j, the change propagation probability cp_{ij} can be estimated based on the values of the *usage coefficients* π_v^{ij} by [9]:

$$cp_{ij} = \frac{1}{|V_i|} \sum_{v \in V_i} \pi_v^{ij} \quad (1)$$

where $|V_i|$ is the cardinality of interface elements set provided by C_i.

3. *THE PROPOSED METHODOLOGY*

Our methodology for estimating maintainability-based risk depends on architectural artifacts such as system requirements and system design and their evolution through the life cycle of the system, as shown in Figure 1. First, we estimate the requirement maturity and map it into components stability. Then, we estimate initial change probabilities of the components. Using the initial change probabilities of the components and change propagation probabilities between them, we get the unconditional probability of change of the components of the system. To get the impact of the maintenance tasks, we use the size of change between the components of the system. Finally, the maintainability-based component risk factor is the product of unconditional change probability and the maintenance impact. We present the detailed steps of the proposed methodology in the following subsections using UML models [10].

3.1 *Estimating requirements maturity index*

We estimate the requirements maturity by analyzing their evolution across the releases of the system. A software system is developed according to a set of requirements

$$RQ = \{rq_1, rq_2, \ldots, rq_p\} \quad (2)$$

where rq_i is a functional requirement. In UML, requirements are mapped into a set of use cases :

$$RQ(UML) = \{uc_1, uc_2, \ldots, uc_p\} \quad (3)$$

Use cases describe the functional behavior of the system. Each use case is realized through one or more sequence diagrams. Sequence diagrams describe the interactions among components to fulfill certain requirements.

Since it is not possible to account for all possible

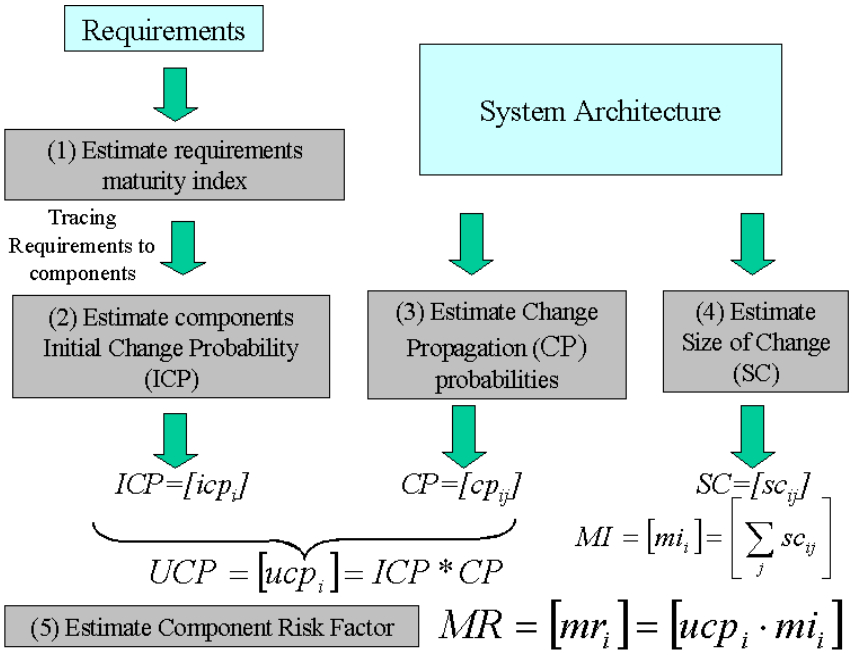

Figure 1 Maintainability-based risk estimation methodology

maintenance tasks, we only consider a maintenance profile *MP* [11] which consists of likely change scenarios

$$MP = \{cs_1, cs_2, \ldots, cs_s\} \quad (4)$$

A change scenario is defined by a set of requirement changes

$$cs_i = \{rq_{1c}, rq_{2c}, \ldots, rq_{tc}\} \quad (5)$$

where rq_{ic} is an addition, deletion or modification of use case uc_i.

The IEEE 982 standard [12] suggests *Software Maturity Index* to quantify properties of requirements evolution. In [13], the Software Maturity Index is adapted to *Requirements Maturity Index* (*RMI*) to measure the requirements stability. We adapt the metric to *Use Case Maturity Index* (*UCMI*) and use function points as a size measure for the use cases [14]. Thus, the *UCMI* of the use case uc_i is given by

$$UCMI = \frac{U_T - U_C}{U_T} \quad (6)$$

where U_T is the function point size of the use case uc_i in the current release; U_C is the function point size of the change in the use case uc_i in the next release from the current due to requirement change rq_{ic} of change scenario cs_m.

3.2 *Estimating components initial change probabilities*

In order estimate the probability of change due to a maintenance task, we use the sequence diagrams to identify the set of components that contribute to each use case. Then, we map the use case stability into components stability, which reflects the likelihood of making changes to components due to changes in the requirements. Consequently, we estimate *Initial Change Probabilities ICP* of the system components. For components that are part of multiple scenarios, we consider the maximum *ICP*, that is, we accounts for the worst-case scenario.

3.3 *Estimating unconditional change probabilities*

To account for the dependency among the components of the system, we multiply the initial change probabilities vector *ICP* of the components by the conditional change propagation probabilities matrix *CP* obtained from the system architecture. Hence, we calculate the unconditional probability UPC of change of each component of the system:

$$UCP = ICP * CP \quad (7)$$

3.4 *Estimating size of change*

To get the impact of the maintenance task, we estimate the size of change *SC* between pairs of the components of the system based on the architecture artifacts. Each element of the *Size of Change* $SC=[sc_{ij}]$ matrix is defined as the ratio between the number of affected methods of the receiving component caused by the changes in the interface elements of the providing components and the total number of methods in the receiving component. For every component C_j, we associate the set M_j of the methods of component C_j. We determine the *effect coefficient* value μ_m^{ij} for every method *m* in component C_j, $j \neq i$. They take binary values:

- μ_m^{ij} =1, if the method *m* is affected by any interface element $v \in V_i$ provided by C_i
- μ_m^{ij} =0, otherwise.

The *size of change* sc_{ij} can be estimated:

$$sc_{ij} = \frac{1}{|M_j|} \sum_{m \in M_j} \mu_m^{ij} \quad (8)$$

where $|M_j|$ is the cardinality of the methods set of component C_j.

Thus, the *Maintenance Impact MI*= $[mi_i]$ is estimated by:

$$mi_i = \sum_j sc_{ij} \; . \quad (9)$$

where mi_i is the maintenance impact of component C_i on the rest of the components of the system.

3.5 *Estimating components maintainability-based risk*

Finally, the components maintainability-based risk MR is given by

$$MR = [mr_i] = [\, upc_i \, . \, mi_i \,] \quad (10)$$

where mr_i is maintainability-based risk of a component C_i due to requirement changes.

Hence, the methodology provides the maintainer with an estimate of the maintainability-based risk of the components for different change scenarios of the maintenance profile. Therefore, the maintainer can identify the risky components and the risky change scenarios. Furthermore, the maintainer can manage the maintenance process by making a trade of between the risk associated with maintenance tasks estimated by the methodology and the added value anticipated from the planned requirements changes.

4. *CASE STUDY*

The maintainability-based risk assessment methodology is illustrated on CM1 case study from the Metrics Data Program [15]. CM1 is a software component of a data processing unit used in an instrument, which exploits data to probe the early universe. A UML-RT model for CM1 is constructed from the artifacts provided. The functional requirements of CM1 are captured in the use case model, as shown in Figure2.

From the use case model, we identify the set of functional requirement RQ as:

$$RQ(\text{CM1}) = \{\text{Transfer, RecvCmd, ChBound, CalcOrbitDrift, HeartBeat, HouseKeeping, TimeSync}\} \quad (11)$$

We estimate the requirement maturity by analyzing their evolution across the releases of the system. As, it is not possible to account for all possible maintenance tasks we only consider a maintenance profile *MP*. For simplicity, we consider a maintenance profile that has only one change scenario:

$$MP = \{cs_1\} \quad (12)$$

Assume that the change scenario that consists of adding a new transfer sequence, shown in Figure3, to the *Transfer* use case:

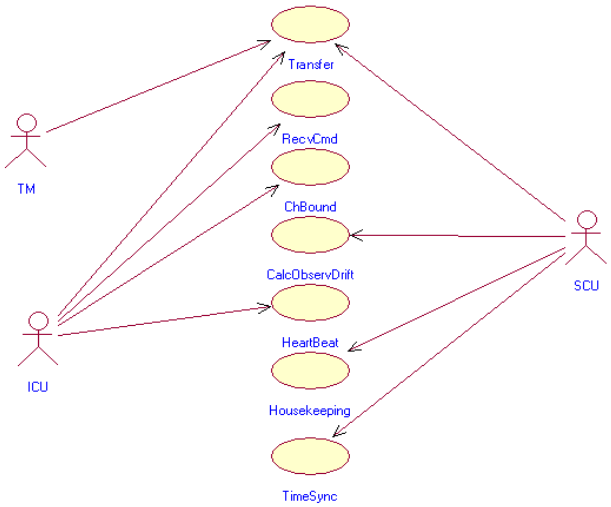

Figure **2** Use case diagram for CM1

$$cs_1 = \{ Transfer_c \} \quad (13)$$

We measure the function point size of the *Transfer* use case in the current release, and the function point size of the changes in the use case *Transfer* in the next release from the current due to $Transfer_c$ of change scenario cs_1. We follow the rules presented in [14] to estimate the function point size of the *Transfer* use case and the change in it. Then, we estimate the use case maturity index, according to equation (8). We find *UCMI(Transfer)=0.702*.

We map the use case maturity index into components stability using the sequence diagram $Transfer_c$. We determine how the stability of each component is affected according to the amount of contribution in the added sequence diagram $Transfer_c$. As component's stability and probability of change are inversely proportional, we make an estimate of initial change probabilities of the components, as shown in Figure 4.

The structure diagram of CM1 is shown in Figure 5. Using the software architecture artifacts of CM1, we estimate the change propagation probabilities and size of change, as shown in Figure 6 and Figure 7.

Substituting initial change probabilities of the components and change propagation probabilities between them in equation (8), we estimate the unconditional probability of change of the CM1 components. Then, we use the size of change between the components to account for the maintenance impact, based on equation (9). Using equation (10), the maintainability-based component risk factor for each CM1 component is estimated. The results are shown in Figure 8.

The most risky component is CCM even though it is not in the set of components of the initial change. This is due to the fact that CCM is coupled to all of the components of the initial change set, so it is likely to be affected by the

changes introduced in these components. Furthermore, CMM has a high maintenance impact on the rest of CM1 components. As it is coupled to other components in the system other than the components of the initial change set, the change is likely to propagate further. On the other hand, component DPA has the highest initial change probability value, but it is coupled to a limited number of components in

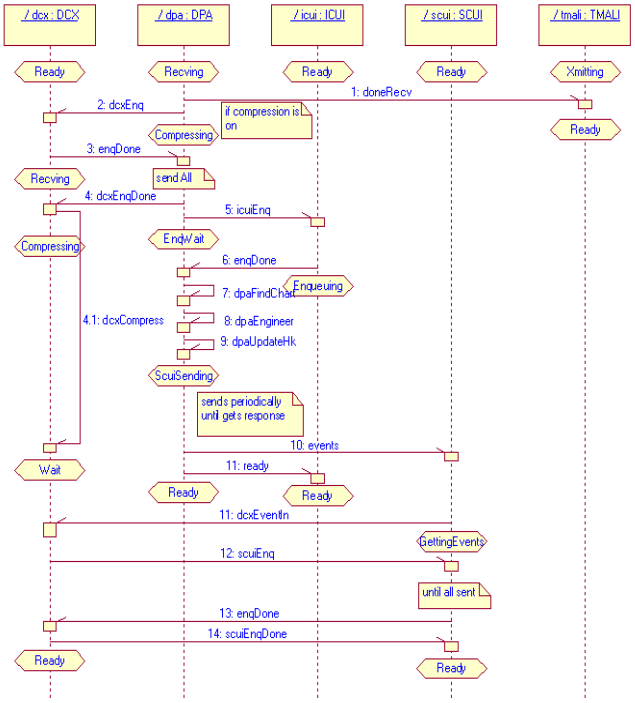

Figure 3 Sequence diagram $Transfer_c$ of added to transfer use case

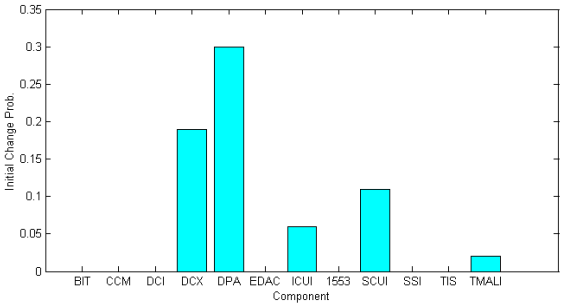

Figure 4 Initial change probabilities resulted from $Transfer_c$ for CM1 components

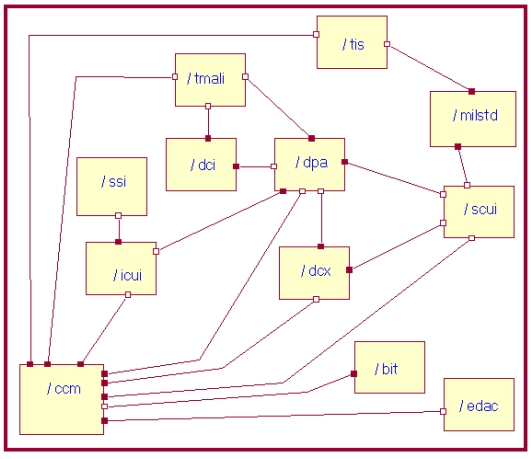

Figure 5 Structure diagram for CM1

CM1. Therefore, it has a limited maintenance impact and it is less risky in terms of maintainability.

Among our venues of future research, we consider

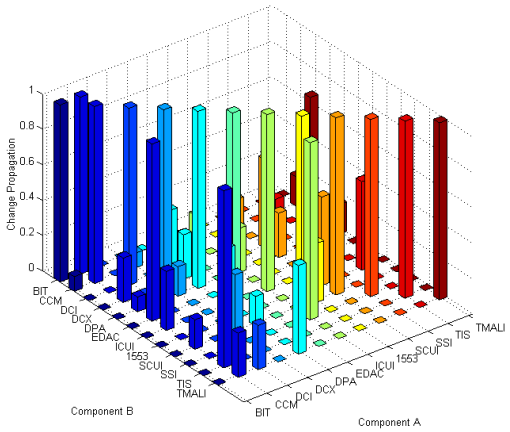

Figure 6 Change propagation probabilities for CM1

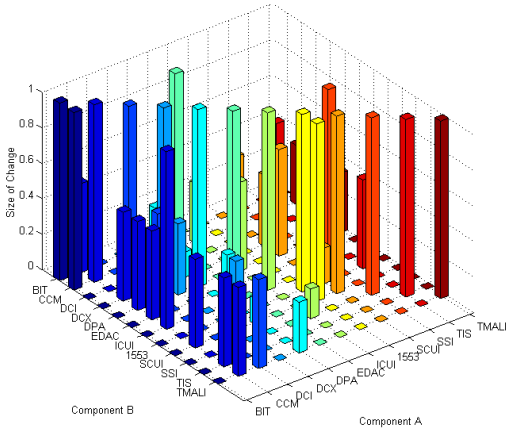

Figure 7 Size of change for CM1

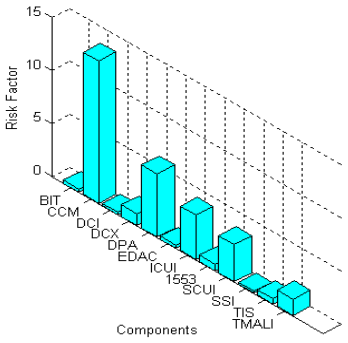

Figure 8 Components maintainability- based risk resulted from Transfer$_c$ for CM1 components

automating the steps of the maintainability-based risk methodology by extending Software Architectures Change Propagation Tool (SACPT) [16]. We also plan to apply the methodology on other case studies and compare the estimated risk with actual maintenance records.

ACKNOWLGEMENT

This work is supported by the National Science Foundation through ITR program grant No CCR 0296082, and by NASA through a grant from the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility, Fairmont, West Virginia.

REFERENCES

1. T.M. Pigoski, Practical Software Maintenance: Best Practices for Managing Your Software Investment, John Wiley & Sons, 1996.
2. NASA-STD-8719.13A,"Software Safety NASA Technical Standard", 1997.
3. W. AbdelMoez, I. Shaik, R. Gunnalan, M. Shereshevsky, K. Goseva-Popstojanova, H.H. Ammar, A. Mili, C. Fuhrman, "Architectural Level Maintainability Based Risk Assessment", IEEE International Conference on Software Maintenance poster proceedings (ICSM 2005), September 25-30,2005, Budapest, Hungray.
4. Sherer S., " Using Risk Analysis to Manage Software Maintenance," Software Maintenance: Research and Practice, Vol. 9, 345-364, 1997.
5. K Goseva-Popstojanova., A. Hassan, A. Guedem, W. Abdelmoez, D. Nassar, H. Ammar, A. Mili, "Architectural-Level Risk Analysis using UML", IEEE transaction on software engineering, Vol.29, No.10, October 2003, pp. 946-960.
6. V. Cortellessa, K. Goseva-Popstojanova, K. Appukkutty, A. Guedem, A. Hassan, R. Elnaggar, W. Abdelmoez, and H. Ammar, "Model-Based Performance Risk Analysis", *IEEE Transaction on Software Engineering,* Vol.31, No.1, January 2005, pp.3-20.
7. P. Oman, J. Hagemeister, "Constructing and Testing of Polynomials Predicting Software Maintainability", Journal of Systems and Software 24, 3 (March 1994), pp. 251-266.
8. S. Muthanna, K. Ponnambalam, K. Kontogiannis and B. Stacey, "A Maintainability Model for Industrial Software Systems Using Design Level Metrics", Seventh Working Conference on Reverse Engineering (WCRE'00), Brisbane, Australia, November 23 - 25, 2000.
9. W. AbdelMoez, M. Shereshevsky, R. Gunnalan, H.H. Ammar, Bo Yu, S. Bogazzi, M. Korkmaz, A. Mili , "Quantifying Software Architectures: An Analysis of Change Propagation Probabilties", ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 05), Cairo, Egypt, January 3-6, 2005.
10. Unified Modeling Language OMG Resource Page http://www.uml.org/
11. J. Bosch and P. Bengtsson, "Assessing Optimal Software Architecture Maintainability", Proc. of fifth European Conference on Software Maintenance and Reengineering, Lisbon, Portugal, March 2001.
12. IEEE Std 982.1- IEEE Standard Dictionary of Measures to Produce Reliable Software.
13. S. Anderson, M. Felici, "Quantitative Aspects of Requirements Evolution". In Proceedings of the 26th Annual International Conference on Computer Software

and Applications Conference, COMPSAC 2002, Oxford, England, 26-29th August 2002, IEEE Computer Society, pp. 27-32.
14. G. Cantone, D. Pace, G. Calavaro, "Applying Function Point to Unified Modeling Language: Conversion Model and Pilot Study" , Proc. of 10th International Symposium on (METRICS'04), September 11 - 17, 2004,Chicago, Illinois, pp.280-291.
15. Metrics Data Program, NASA IV&V Facility http://mdp.ivv.nasa.gov/
16. W. Abdelmoez, R. Gunnalan, M. Shereshevsky, H.H. Ammar, Bo Yu, M. Korkmaz, A. Mili, "Software Architectures Change Propagation Tool (SACPT)", *Proc. 20th IEEE International Conference on Software Maintenance (ICSM 2004)*, Chicago, IL, September 2004.

BIOGRAPHIES

Walid M. Abdelmoez
Lane Dept of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV26506-6109 USA

rabie@csee.wvu.edu

Walid Abdelmoez received the BSc degree in electrical engineering at Alexandria University, Egypt in 1995 and the MSc degree in electrical engineering at Arab Academy for Science and Technology, Alexandria, Egypt, in 2000. He is a PhD student in computer engineering program at West Virginia University and a graduate research assistant in the LDCSEE. His research interests are software metrics and software risk assessment. He is student member of the IEEE.

Katerina Goseva-Popstojanova, PhD
Lane Dept of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV26506-6109 USA

katerina@csee.wvu.edu

Katerina Goseva-Popstojanova is an assistant professor in the Lane Department of Computer Science and Electrical Engineering at West Virginia University, Morgantown. Prior to joining West Virginia University, she was a postdoctoral research associate in the Department of Electrical and Computer Engineering at Duke University, Durham, North Carolina. Her research interests include software reliability engineering, dependability, performance and performability assessment of software and systems, and computer security and survivability. She has published more than 50 journal and conference articles on these topics. Dr. Goseva-Popstojanova is a principal investigator on several projects funded by the NASA Office of Safety and Mission Assurance (OSMA), Software Assurance Research Program (SARP) managed through the NASA IV&V Facility in Fairmont, WV. She is also a recipient of the National Science Foundation CAREER award. She served and is currently serving on program and organizing committees of several prestigious conferences in her research areas. She is a senior member of the IEEE and member of the ACM.

Hany H. Ammar, PhD
Lane Dept of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV26506-6109 USA

ammar@csee.wvu.edu

Hany H. Ammar is a professor of computer engineering in the Department of Computer Science and Electrical Engineering at West Virginia University. His research interests are in software engineering, software architectures, software metrics, and identification technology. He is the director of the Software Architectures and High Performance Computing Lab at WVU. He is leading several projects funded by the US National Science Foundation under the Digital Government and ITR programs and NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility, Fairmont, West Virginia. He has published more than 100 articles in prestigious journals and conference proceedings. He served and is currently serving in the program and steering committees of several professional conferences and workshops. Dr. Ammar is a member of the IEEE Computer Society and the ACM.