Measuring Developers' Design Contributions in Evolved Software Projects

Maen Hammad
Department of Software Engineering, The Hashemite University, Zarqa, Jordan
Email: mhammad@hu.edu.jo

Mustafa Hammad
Department of Information Technology, Mu'tah University, AL Karak, Jorda
Email: hammad@mutah.edu.jo

Hani Bani-Salameh, Ebaa Fayyoumi
{Department of Software Engineering, Department of Computer Science and Applications}
The Hashemite University, Zarqa, Jordan
Emails: {hani, enfayyoumi}@hu.edu.jo

***Abstract*—The work presented in this paper measures the contribution of developers towards evolved structural design of software systems. Measuring the contribution of developers is useful for project managers who manage the development process. Project managers can empirically identify developers who made changes to the structural design and compare among them based on their contributions. The proposed measures help to understand the nature of developers' code changes activities. The process of calculating the measures is based on the historical code changes committed by developers. Specifically, code changes that affect the corresponding UML class diagram representation of the source code. Both type and amount of previous changes to class elements are used to measure the design contributions. The proposed measures are helpful for open source projects where no detailed information is available about various developers involved in the development process. A tool has been developed to automatically measure the contributions based on archived historical code changes. The measures have been applied on two open source projects. Results showed that only small number of developers have the major design contributions.**

***Index Terms*—Software development and design, software measurements, software evolution.**

I. INTRODUCTION

The development process of software systems is difficult to manage and control. Many developers with different skills, experiences, and roles are involved in the development process. Developers usually maintain software system via a set of commits, which are committed over time. As a result, not all developers have the same contributions in the evolved software projects. They vary in the number and the type of code changes that they committed. Some developers make major contributions, while others' contributions are minor and limited to small and specific code changes.

Measuring the contributions of developers helps project managers to get more information about their teams and about the nature of their coding activities. Analyzing design contributions of developers helps in identifying who are the senior developers and estimating the amount of their contributions to the evolved structural design.

Some software developers focus on developing test cases, while others focus on updating documents. In this paper, we focus on developers' contributions that are related to the design of software systems. The structural design of software systems is mainly represented by the UML class diagram of the source code that includes classes and their relationships. Therefore, the design elements are classes, methods and relationships between classes. A relationship between two classes can be a generalization, an association, or a dependency.

The measures best support open source projects, where detailed information about each developer is not always available. Besides, most developers are volunteers from different countries. This consequently increases the difficulty of identifying the design contribution of developers. It is also not an easy task to identify developers who make the largest contributions to the structural design of the system.

The question that needs to be addressed is: *how to measure developers' contributions in the evolved structural design of software systems?*

Measuring the contributions of developers in structural design provides many benefits to project managers. It leads to understand the nature of code changes activities of developers. For example, some developers do most structural design changes in one or two commits, while others make their changes in small chunks over longer period of time. It is important to project managers to identify developers who have made significant impacts on the structural design. In particular, they also need to know the contribution of a specific developer during a

doi:10.4304/jsw.9.12.3005-3011

specific period of time, or active developers in specific time duration.

Developers, who update class elements of the source code, contribute in building the structural design of the system. Measuring developers' contributions can be achieved by defining new quantitative metrics based on analyzing the previous code changes activities of developers.

The paper proposes two historic-based quantity measures for developers' contribution in the evolved structural design. The first measure reflects the added design elements by the developer, while second measure reflects deleted design elements. Code changes in software repositories are examined to calculate the contributions. The work is focused on measuring the contribution of developers to structural design of the source code which include classes, methods, and relationships. Minor code changes, such as adding a condition or renaming a variable, are not considered a design contribution.

The main research contributions that are addressed in this paper are:

1. Two historic-based measures for the contributions of developers towards evolved design.
2. A case study about applying the metrics on two open source projects.
3. A tool to automatically calculate design contributions for developers.

The paper is organized as follows. Section 2 explains the measured design elements. The proposed measures are described in Section 3. The developed that supports the measure is presented in Section 4. A detailed case study on two open source projects is discussed in Section 5. Related work is presented in Section 6 followed by conclusions and future work.

II. Design Contributions

Our concern is on measuring contributions of developers to the structural design of the source code. The structural design is mainly represented by classes and their relationships. The contents of classes include attributes and methods, while relationships include generalizations, associations and dependencies. The work presented in this paper is a continuation to our previous work that is presented in [1]. In [1], developers who updated design and their knowledge in the updated design are identified. It is shown that the identified design knowledge can help in handling high level change requests. In this paper, we defined measures for the contribution of developers who updated the structural design of the system.

The design contribution of a developer is measured by the total number of added or deleted design elements committed by the developer during specific time duration. In general, design contributions can be categorized into two categories; the addition of new design elements and the deletion of old design elements.

The benefits of measuring design contributions for developers can be summarized as follows:

- Identifying core developers who are important to the structural design of the system.
- Identifying the contribution of a developer during specific time duration.
- Understanding the nature of code changes activities for developers who updated the structural design.

The measured contributions include the following changing activities that directly affect the main elements of classes:

- Creating or deleting classes.
- Adding or deleting methods to/from classes.
- Adding or deleting relationships (generalizations, associations or dependencies) between classes.

These changes to the classes are extracted from code changes that are committed by developers. As a result, any developer committed at least one of the above changes has a design contribution to the structural design of the source code.

III. The Proposed Measures

This section introduces and explains the proposed measures for the design contributions of software developers. Two historical-based measures are proposed to quantify the detailed design contribution of developers. The proposed measures are calculated from the historical code changes committed by developers.

Based on the definition of design contributions introduced in Section II, we propose two measures. The first measure is the total number of Added Design Elements (ADE) by the developer during specific time duration. The second one is the total number of Deleted Design Elements (DDE) by the developer during specific time duration. The following two equations show how the two measures are calculated for a specific developer (d).

$$\mathrm{ADE}(d) = \mathrm{w1} \cdot \sum_{C=1}^{n} \mathrm{AC_c(d)} + \mathrm{w2} \cdot \sum_{C=1}^{n} \mathrm{AM_c(d)} + \mathrm{w3} \cdot \sum_{C=1}^{n} \mathrm{AR_c(d)}$$

$$\mathrm{DDE}(d) = \mathrm{w1} \cdot \sum_{C=1}^{n} \mathrm{DC_c(d)} + \mathrm{w2} \cdot \sum_{C=1}^{n} \mathrm{DM_c(d)} + \mathrm{w3} \cdot \sum_{C=1}^{n} \mathrm{DR_c(d)}$$

The first measure (ADE) sums the total number of added classes (AC), added methods (AM), and added relationships (AR) that are identified from all extracted commits C from 1 to n for a specific developer *d*. Meanwhile, the DDE is the summation of the total number of deleted classes (DC), deleted methods (DM), and deleted relationships (DR) for all extracted n commits of the developer.

One important factor in calculating the contribution is the weight of design change. For example, a developer who added three classes may be considered to have more

contributions to the design than the one who added three methods. Even if the ADE for both developers is three, adding a class may have more impact than adding a method. To include weights in ADE and DDE, each summation is multiplied by a weight value (w1, w2, or w3) such that:

$$\sum_{i=1}^{3} \mathrm{w_i} = 1$$

These weights are determined by project managers to rank design contributions based on their types. For example, weight values could be set as (w1=0.6, w2=0.3, w3=0.1). These weight values give the highest design contributions to added/deleted classes then added/deleted methods followed by added/deleted relationships. In this case, the contribution of adding one class is counted as twice compared to adding one method and six times compared to adding a relationship.

Another option is to consider an equal weight to all design contributions regardless of their types and their impact on the whole design system. In this case, weights are set to be equal to 0.33+ or simply ignore the weight concept.

The process of finding design contributions for developer *d* in project P is calculated as follows:

1. A set of commits are extracted from the software repository of project P.
2. The set of commits C_n that are committed by developer *d* are identified from the extracted commits in Step 1.
3. All added/deleted classes, methods and relationships are identified from the set of commits C_n.
4. The total number of added design elements identified in Step 3 is reported as the ADE design contribution value for developer *d*.
5. The total number of deleted design elements identified in Step 3 is reported as the DDE design contribution value for developer *d*.

The proposed measures are based on counting the number of added and deleted design elements across all commits. All added and deleted classes, methods and relationships (generalization, association, or dependency) are counted. For a specific developer, all his commits for a specific period of time are extracted. Then, code changes in these commits are analyzed to identify commits that have design changes. In the next step, the number of added and deleted design elements per commit, with design changes, is counted. Finally, the sum of all added design elements is reported as ADE value and the sum of all deleted design elements is reported as DDE value.

For example, consider a developer who committed three commits during a month. In the first commit, he created a new class with four methods. In the second commit, he deleted an association relationship. In the third commit, the developer added another new method to some class. Therefore, his design contribution is six based on the ADE measure because he added six new design elements (one class + four methods + one method). Based on the DDE measure his contribution is one because he deleted only one design element (an association) from the design.

Combining ADE and DDE together helps in determining the total design contribution of a specific developer to design. Consequently, the summation of the two measures ADE and DDE gives the Total Design Contribution (TDC) of a developer *d*.

$$TDC(d) = \mathrm{ADE\ (d)} + \mathrm{DDE\ (d)}$$

TDC represents the total number of design changes that are committed by a certain developer regardless of the change type.

Another variation to TDC is the measuring of Actual Total Design Contribution (ATDC) for certain developer. ATDC is counted by excluding any design element that is added and then deleted by the same developer. This design element presents an intersected element between the ADE(d) and DDE(d). Consequently, this intersected element represents an overlap that is counted twice. Therefore, it has to be excluded once. This is because that design element is not part of the current status of the design anymore. For instance, suppose a developer has added one method and later he deleted it. This change is counted zero based in ATDC and it is counted two based on TDC. The summation of the two measures ADE and DDE excluding the Overlap Design Elements (ODE) gives the Actual Total Design Contribution (TDC) of a developer *d*.

$$TDC(d) = \mathrm{ADE\ (d)} + \mathrm{DDE\ (d)} - \mathrm{ODE\ (d)}$$

Our focus in this paper is on measuring TDC for developers, which reflects the amount of their changes activities they applied.

Since ADE and DDE are numbers, it is also useful to show the total design contribution of a developer as a percentage of all other developers. The total design contribution of a specific developer is divided by the total number of design contributions committed by all developers. The result of the division represents the Percentage Total Design Contribution (PTDC) for the developer. The PTDC for developer *d* among *n* developers who involved in the development process during a specific period of time is calculated as shown in the following equation:

$$PTDC\ (d) = (\mathrm{TDC\ (d)} / \sum_{i=1}^{n} \mathrm{TDC_i}\ ) * 100$$

IV. Tool Support

A tool has been developed to automatically find the design contribution for developers. The tool supports both extracting commits, and measuring design contributions (ADE and DDE) based on specific weights determined by the user. The tool is named *Contributor*. Contributor finds the contributions values of a specific developer. Users can determine the URL of the open source repository, the time duration of the history, the weights values, and the developer's ID. Then, the

contribution is automatically calculated and shown to the user.

Contributor extracts commits from the determined repository and analyzes code changes of the extracted commits by using another tool called srcTracer [2, 3]. The srcTracer tool automatically analyzes C++ code changes to identify changes to design. The tool identifies added/deleted classes, methods, and relationships between classes.

The design contributions for all developers or a group of developers who are involved in the development process are calculated by the tool during predefined time duration. Also, developers can be ranked based on their design contributions (ADE, DDE or TDC). The tool is able to calculate the percentage total design contribution (PTDC) and actual total design contribution (ATDC) for a specific developer or a group of developers.

V. Case Study

We applied the proposed contribution measures, by using the Contributor tool, on two C++ open source projects. The results are analyzed and discussed in this section. The two studied open source projects are:

1. The quantitative finance library QuantLib (www.quantlib.org)
2. The cross-platform GUI library wxWidgets (www.wxwidgets.org).

Subset of commits has been extracted from each of the two projects. The goal of the study is to investigate the following issues:

- What is the design contribution of a specific developer?
- Who are the developers with major design contributions?
- Do all developers vary in their design contributions?

The extracted commits from the two projects have been analyzed by the Contributor tool to calculate the design contributions for all developers who committed the extracted commits. Tables 1 and 2 show the top ten developers of wxWidgets and QuantLib with the design contribution values for each developer. The contribution is measured by ADE and DDE measures. The total design contribution (TDC) is shown for each developer in the last column which is used to rank developers. The weights are ignored in calculating ADE and DDE (i.e. all design changes have the same weight).

For example, in wxWidgets, developer VZ is ranked first based on his total design contribution. He updated the design by adding 3994 (class, method or relationship) and deleting 1729 (class, method, or relationship). The total number of added and deleted design elements by VZ is 5723 (3994+1729). Tables 1 and 2 also show that developers vary in their contributions to the evolved structural design.

It is observed that most structural design changes are applied by small number of developers. To investigate this observation, we calculated the cumulative design contribution of top five developers of wxWidgets and top three developers of QuantLib. Five developers of wxWidgets represent 20% (5/25) of all developers who contributed to the structural design. Also, the top three developers of QuantLib represent 20% (3/15) of developers with design contributions. The cumulative design contributions were calculated by summing the total design contribution (TDC) for each developer. Top developers mean developers with highest contributions values. The cumulative design contribution for the top developers of the two projects is shown in Table 3.

TABLE I.

DESIGN CONTRIBUTIONS FOR TOP TEN DEVELOPERS OF QUANTLIB

Developer	ADE	DDE	TDC (ADE+DDE)
Lballabio	5758	624	6382
Nando	2541	993	3534
fdv1	1285	59	1344
Giorfa	711	194	905
Klausspanderen	559	57	616
Markjoshi	348	0	348
Chiforna	300	22	322
Kmanzoni	234	51	285
Cduminuco	182	48	230
Mariopucci	165	30	195

TABLE II.

DESIGN CONTRIBUTIONS FOR TOP TEN DEVELOPERS OF WXWIDGETS

Developer	ADE	DDE	TDC (ADE+DDE)
VZ	3994	1729	5723
JS	2478	155	2633
RR	1701	575	2276
SC	1088	390	1478
VS	861	427	1288
RD	1015	90	1105
ABX	669	386	1055
BIW	491	135	626
MW	270	108	378
KO	177	37	214

TABLE III

CUMULATIVE CONTRIBUTIONS FOR TOP FIVE DEVELOPERS OF WXWIDGETS AND TOP THREE DEVELOPERS OF QUANTLIB

wxWidgets	TDC	QuantLib	TDC
Top 5 developers	13398 (76%)	Top 3 developers	11260 (77%)
Others developers	4242 (24%)	Others developers	3318 (23%)

We found that the top five developers of wxWidgets, contributed by adding and deleting 13398 design elements which represent 76% of all updated elements by all developers. The same observation is valid on QuantLib developers. Three developers only contributed about 77% of design changes in QuantLib project. These top developers may be considered as the core designers of the system because of their large contributions.

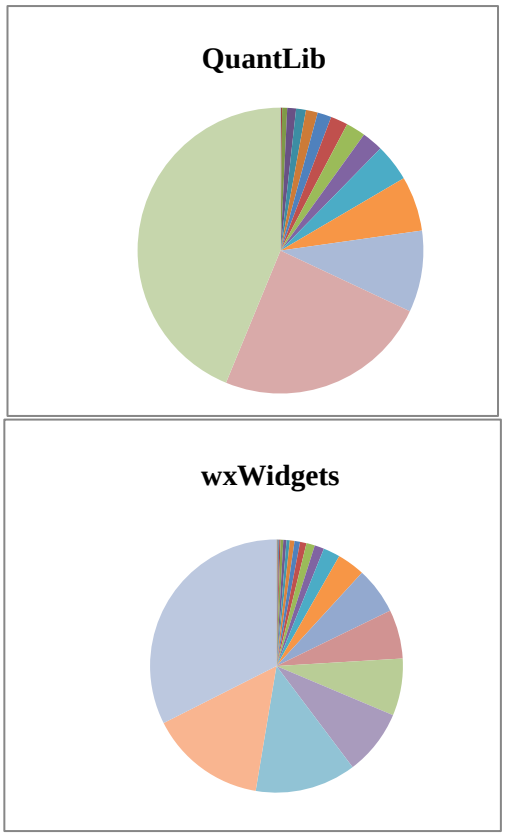

Figure 1. Distribution the percentages of total design contributions (PTDC) of wxWidgets and QauntLib developers

Fig. 1 visualizes the PTDC values of all developers based on the extracted commits. Developers with no design contributions are ignored. The PTDC of each developer is presented by a pie in the chart. The distribution of PTDC values for all developers is shown for both projects. The charts show that most developers have minor design contributions towards evolved structural design. For example, in QuantLib project shown in Fig. 1, only one developer contributed more than 25% of design changes. The same observation is applied for wxWidgets. Only two developers have contributed about 50% of all design changes. In summary, only small number of developers contributes the most changes activities towards evolved structural design. Project managers should give great attention to those developers due to their importance to the organization or to the development process.

VI. Related Work

Ramil and Lehman [4] described some models that predict effort as a function of a suite of metrics of software evolution. A close work to ours is proposed by Gousios et al. [5]. They proposed a model for evaluating developer contribution. The model takes advantage of software development repositories to extract contribution indicators. The model assigns a weight to each action type by the developer. The approach measures many development activities but not contributions related to the structural design of source code. Di Penta and German [6] introduced a method to track the names of contributors, including those explicitly listed as copyright owners from licensing statements in source code file. Kpodjedo et al. [7] investigated the usefulness of elementary design evolution metrics to identify defective classes. The metrics include the numbers of added, deleted, and modified attributes, methods, and relations. The metrics are used to recommend a ranked list of classes likely to contain defects for a system.

Many studies as in [8, 9] showed that most of code changes are contributed by small number of developers. Bird et al. [10] mined e-mail archives to analyze the communication and co-ordination activities of the participants. In our work, we showed that most structural design changes are made by small number of developers.

Del Rosso [11] used collaborations and interactions between knowledge-intensive software developers to build a social network. Sowe et al. [12] presented a framework for investigating Free and Open Source Software (F/OSS) developers' activities in both source code and mailing lists repositories. Kagdi et al [13] presented an approach with a tool named xFinder. The approach recommends expert developers by mining version archives of a software system. The approach is based on the idea that the developers who contributed more in changing a specific part of source code in the past are likely to best assist in its current or future changes. The approach is used to recommend experts for files. The approach has been used in [14, 15] to recommend a ranked list of expert developers to assist in the implementation of software change requests.

Ben et al. [16] examined the contribution characteristics of developers in open source environment based on visual analysis, and presented approaches from three aspects-influencing factors, time characteristics and region characteristics. They found that the code which newcomers started to contribute with more people engaged in would lead to less contribution in some degree. They also found that there's a relation between developers' early and later period contribution. Oosterman et al. [17] introduces a tool named EVOJAVA for extracting static software metrics from a Java source code repository. For each version of a program, EVOJAVA builds a comprehensive model of the semantic features described by Java code, and tracks the identity of these features as they evolve through sequential versions. This allows software metrics to be recorded over time. Eden and Mens [18] introduced the notion of evolution complexity and demonstrated how it can be used to measure and compare the flexibility of programming paradigms, architectural styles and design patterns. Canfora et al. [19] [20] investigated the relationship of source code complexity and disorganization. They used three factors which includes the number of contributors that modified the source code file. They found that entropy tends to increase with the number of file committers.

Our work differs in the type of the measured developer's activities. We measure the contribution of developers based on their updates on the structural design of the system.

VII. Conclusions And Future Work

Two measures have been proposed to measure the contribution of software developers in the evolved structural design of software systems. The presented measures consider only code changes that impact the corresponding UML class diagram representation of the source code. The measures are based on mining software repositories to analyze the historical code changes for developers. The number of updated design elements in the structural design is used for calculating the contributions. Measuring design contributions for developers provide useful information about the amount of contributions committed by developers to the structural design. The measures help project managers in locating developers who have major contributions to the structural design and empirically identify the amount of their contributions. The proposed contribution measures have been applied on two open source project. Results show that developers vary in their design contributions. It is also shown that very small number of developers contributes in most of design changes.

The future work aims to introduce new metrics for developers that are based on different code activities. Possible measurements that are under investigation include the number of added/deleted packages and the amount of code refactrings. We are currently working on defining general metrics to measure the evolution of software systems.

References

[1] M. Hammad, M. Hammad and H. Bani-Salameh, "Identifying Designers and their Design Knowledge," *International Journal of Software Engineering and Its Applications*, vol. 7, no. 6, 2013, pp. 277-288.

[2] M. Hammad, M.L. Collard and J.I. Maletic, "Automatically Identifying Changes that Impact Code-to-Design Traceability," *Proc. 17th IEEE International Conference on Program Comprehension (ICPC'09)*, 2009, pp. 20-29.

[3] M. Hammad, M.L. Collard and J.I. Maletic "Automatically Identifying Changes that Impact Code-to-Design Traceability during Evolution," *Software Quality Journal* 2010, pp. (under review).

[4] J.F. Ramil and M.M. Lehman, "Metrics of Software Evolution as Effort Predictors - A Case Study," *Proc. Proceedings of the International Conference on Software Maintenance (ICSM'00)*, 2000.

[5] G. Gousios, E. Kalliamvakou and D. Spinellis, "Measuring Developer Contribution from Software Repository Data," *Proc. Proceedings of the 2008 international working conference on Mining software repositories (MSR'08)* 2008, pp. 129-132.

[6] M.D. Penta and D.M. German, "Who are Source Code Contributors and How do they Change?," *Proc. 16th Working Conference on Reverse Engineering*, 2009.

[7] S. Kpodjedo, F. Ricca, P. Galinier, Y.-G. Guéhéneuc and G. Antoniol, "Design evolution metrics for defect prediction in object oriented systems," *Empirical Software Engineering*, vol. 16, no. 1, 2011, pp. 141-175.

[8] J. Geldenhuys, "Finding the Core Developers," Proc. Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'10), 2010, pp. 447-450.

[9] D.M. German, "A study of the contributors of PostgreSQL," Proc. Proceedings of the 2006 international workshop on Mining software repositories (MSR'06), 2006.

[10] C. Bird, A. Gourley, P. Devanbu, M. Gertz and A. Swaminathan, "Mining Email social networks," *Proc. International Workshop on Mining Software Repositories (MSR '06)*, 2006, pp. 137–143.

[11] C.D. Rosso, "Comprehend and analyze knowledge networks to improve software evolution," *Journal of Software Maintenance and Evolution: Research and Practice (JSME)*, vol. 21, no. 3, 2009, pp. 189–215.

[12] S.K. Sowe, S. Ioannis, S. Ioannis and A. Lefteris, "Are FLOSS developers committing to CVS/SVN as much as they are talking in mailing lists? Challenges for Integrating data from Multiple Repositories," *Proc. 3rd Workshop on Public Data about Software Development (WoPDaSD 2008)*, 2008.

[13] H. Kagdi, M. Hammad and J.I. Maletic, "Who can help me with this source code change?," *Proc. IEEE International Conference on Software Maintenance (ICSM'08)*, 2008, pp. 157–166.

[14] H. Kagdi, M. Gethers, D. Poshyvanyk and M. Hammad, "Assigning change requests to software developers," *JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS*, vol. 24, no. 1, 2012, pp. 3–33.

[15] H. Kagdi and D. Poshyvanyk, "Who can help me with this change request?," *Proc. The 17th IEEE International Conference on Program Comprehension (ICPC'09)*, 2009, pp. 273–277.

[16] X. Ben, S. Beijun and Y. Weicheng, "Mining Developer Contribution in Open Source Software Using Visualization Techniques," *Proc. Third International Conference on Intelligent System Design and Engineering Applications (ISDEA'13)*, 2013, pp. 934 - 937.

[17] J. Oosterman, W. Irwin and N. Churcher, "EvoJava: A Tool for Measuring Evolving Software," *Proc. Thirty Fourth Australasian Computer Science Conference (ACSC2011)*, 2011.

[18] A.H. Eden and T. Mens, "Measuring Software Flexibility," *IEE Software*, vol. 153, no. 3, 2006, pp. 113–126.

[19] G.C. Canfora, L.; Di Penta, M.; Pacilio, F., "An Exploratory Study of Factors Influencing Change Entropy," *Proc. Proceedings of the IEEE 18th International Conference on Program Comprehension (ICPC'10)*, 2010, pp. 134,143.

[20] G. Canfora, L. Cerulo, M. Cimitile and M.D. Penta, "How changes affect software entropy: an empirical study," *Empirical Software Engineering*, vol. 19, no. 1, 2014, pp. 1-38.

Maen Hammad is an Assistant Professor in Software Engineering Department at The Hashemite University, Jordan. He completed his Ph.D. in computer science at Kent State University, USA in 2010. He received his Master in computer science from Al-Yarmouk University—Jordan and his B.S. in computer science from The Hashemite University—Jordan. His research interest is Software Engineering with focus on software evolution and maintenance, program comprehension and mining software repositories.

Mustafa Hammad is an Assistant Professor at Information Technology department in Mu'tah University, Al Karak - Jordan. He received his PhD. in computer science from New Mexico State University, USA in 2010. He received his Masters degree in computer science from Al-Balqa Applied University, Jordan in 2005 and his B.Sc. in computer science from The Hashemite University, Jordan in 2002. His research interest is

Software Engineering with focus on static and dynamic analysis and software evolution.

Hani Bani-Salameh is an Assistant Professor in the Software Engineering Department at The Hashemite University since 2011. He holds a B.Sc. in Computer Science (1995), a M.Sc. in Computer Science (2007) from New Mexico State University (NMSU), and a Ph.D. in Computer Science (2011) from the University of Idaho (UI). His research interests include software engineering, computer supported cooperative work (CSCW), software development environments, collaborative software development in virtual environments, and social networking and social media. He studies social interactions in social networks and online environments, including Facebook and SourceForge.

Ebaa Fayyoumi received the B.Sc. degree from The Hashemite University, Zarqa, Jordan, in 2000, the M.Sc. degree from the University of Jordan, Amman, Jordan, in 2002, and the Ph.D. degree from Carleton University, Ottawa, ON, Canada, in 2008. She has been with the Faculty of Prince Hussein Bin Abdalla II for Information Technology, The Hashemite University, since November 2008, where she is currently an Assistant Professor at the Computer Science Department. Her current research interests include statistical syntactical pattern recognition, micro-aggregation techniques, secure statistical databases, automata learning, applied algorithm, mobile application, data mining and e-learning. She got many awards during her academic life; one of them is Carleton University Medal on outstanding graduate work in 2008.