VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kansliets noteringar Kod	Dnr
2014-48274-115757-57	2014-4925

2014
Project Research Grant

Area of science
Natural and Engineering Sciences
Announced grants
Research grants NT April 9, 2014
Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
1637	1669	1727	1767	

APPLICANT

Name(Last name, First name)
Weyuker, Elaine

Date of birth
451122-

Gender
Female

Email address
daniel.sundmark@mdh.se

Academic title
Professor

Position
Visiting Professor

Phone
021103145

Doctoral degree awarded (yyyy-mm-dd)
1977-04-10

WORKING ADDRESS

University/corresponding, Department, Section/Unit, Address, etc.
Mälardalens Högskola
Akademin för innovation, design och teknik
Inbyggda System
Box 883
72123 Västerås, Sweden

ADMINISTRATING ORGANISATION

Administrating Organisation
Mälardalens Högskola

DESCRIPTIVE DATA

Project title, Swedish (max 200 char)
EXACT - Experimentell Analys av Kopplingseffekthypotesen i Programvarutestning

Project title, English (max 200 char)
EXACT - Experimental Analysis of the Coupling Effect Hypothesis in Software Testing

Abstract (max 1500 char)
In software testing, mutation analysis is a technique that systematically inserts simple bugs into a program under test. Once a set of buggy programs, known as mutants, has been created, they are run on a set of test cases. If all mutants fail, the set of test cases (or the technique selecting them) is deemed good. Fundamental to mutation analysis is the so-called coupling effect hypothesis. The hypothesis states that a set of test cases detecting most simple bugs in a program will also detect most complex faults in the same program. If the coupling effect does not exist for real software systems, then mutation analysis is not a reliable way of deciding whether or not a program is thoroughly tested. This would also cast doubt on the use of mutation analysis in research as a way of assessing the effectiveness of testing techniques.

The validity of any evidence on the coupling effect is limited by the extent to which the studied programs and faults are representative of real-world programs and naturally-occurring faults. However, nearly all existing studies of the coupling effect use small programs and artificial faults generated for the sake of the experiment. In this project, we will empirically study the coupling effect in large software systems with naturally-occurring faults. We expect to provide significant evidence supporting or refuting the hypothesis, allowing researchers and testers to determine whether mutation analysis is a meaningful way to assess their work.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Name of Applicant
Weyuker, Elaine

Date of birth
451122-

Abstract language
English

Keywords
software, testing, mutation analysis, coupling effect, empirical studies

Review panel
NT-2, NT-13

Project also includes other research area

Classification codes (SCB) in order of priority
20206, 20207,

Aspects

Continuation grant
Application concerns: New grant
Registration Number:

Application is also submitted to

similar to: **identical to:**

ANIMAL STUDIES

Animal studies
No animal experiments

OTHER CO-WORKER

Name(Last name, First name)
Sundmark, Daniel

University/corresponding, Department, Section/Unit, Addressetc.
Mälardalens Högskola
Akademin för innovation, design och teknik

Date of birth
750409-6954

Gender
Male

Academic title
Associate professor

Doctoral degree awarded (yyyy-mm-dd)
2008-02-15

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

Name(Last name, First name)
,

University/corresponding, Department, Section/Unit, Addressetc.

Date of birth

Gender

Academic title

Doctoral degree awarded (yyyy-mm-dd)

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Name of Applicant
Weyuker, Elaine

Date of birth
451122-

ENCLOSED APPENDICES

A, B, B, C, C, N, S

APPLIED FUNDING: THIS APPLICATION

Funding period (planned start and end date)
2015-01-01 -- 2018-12-31

Staff/ salaries (kSEK)

Main applicant	% of full time in the project	2015	2016	2017	2018	2019
Elaine Weyuker	40	919	947	975	1005	
Other staff						
Daniel Sundmark, Male, 1975, Year of PhD: 2008, Monthly salary: 42.600	50	630	649	669	689	
Total, salaries (kSEK):		1549	1596	1644	1694	

Other project related costs (kSEK)	2015	2016	2017	2018	2019
Equipment	20	5	15	5	
Travel and networking	68	68	68	68	
Total, other costs (kSEK):	88	73	83	73	

Total amount for which applied (kSEK)

2015	2016	2017	2018	2019
1637	1669	1727	1767	

ALL FUNDING

Other VR-projects (granted and applied) by the applicant and co-workers, if applic. (kSEK)

Project title	Applicant	Funded 2014	Funded 2015	Applied 2015
PERTES: Performance Testing of Embedded Software	Paul Pettersson			4868

Funds received by the applicant from other funding sources, incl ALF-grant (kSEK)

POPULAR SCIENCE DESCRIPTION

Popularscience heading and description (max 4500 char)

Programvara är en stor del av våra liv. Den kör våra tåg, flyger våra flygplan, styr våra telefoner och finns inuti i princip varje maskin och pryl vi äger. All denna programvara behöver på något sätt kontrolleras innan den börjar användas för att se till att den fungerar på rätt sätt. Normalt sett sker denna validering genom testning. Ofta, även i säkerhetskritiska system, görs tester manuellt av mänskliga testare utan hjälp av någon särskild metod eller systematik. Det betyder inte att det görs dåligt eller slarvigt. Testare är ofta experter som har flera

års erfarenhet och en intuition som kommer från en djup förståelse för produkten de testar. Det finns emellertid få automatiserade valideringstekniker som är användbara för testning av mer komplexa produkter. Det är inte alls ovanligt att programvarusystem innehåller miljontals rader kod, som alla måste testas på flera nivåer för både funktion och andra viktiga egenskaper såsom prestanda.

Eftersom programvarusystem oftast valideras med hjälp av stora mängder testfall, är det viktigt att ha konkreta och meningsfulla metoder för att avgöra när det är säkert att sluta testa. Vi vill på ett säkert sätt veta om programvaran fortfarande innehåller fel. Av denna anledning har flera forskare föreslagit mutationsanalys som ett sätt att bedöma om programvara är tillräckligt testad. Mutationsanalys är en teknik som systematiskt injicerar förändringar i ett program. Dessa förändringar syftar till att representera enkla vanliga fel som en kompetent programmerare skulle kunna göra. De förändrade programmen kallas mutanter. Genom att testa mutanterna med en mängd testfall så kan man få en bild av hur många av de enkla felen som hittas av testfallen. Mutationsanalys har ofta använts i forskning om programvarutestning som ett sätt att experimentellt bedöma effektiviteten hos olika testtekniker.

En grundläggande princip i mutationsanalys är hypotesen om den så kallade kopplingseffekten (coupling effect hypothesis). Hypotesen säger att om en mängd testfall är tillräckligt heltäckande för att upptäcka en fördefinierad uppsättning av enkla fel, kommer den också att kunna upptäcka förekomsten av mer komplexa fel. Givet att hypotesen är sann och kopplingseffekten existerar, så är mutationsanalys ett adekvat sätt att bedöma vilka testtekniker och testfall som är effektiva för att hitta enkla såväl som komplexa fel. Om hypotesen är falsk, så vilar en stor del av kunskapen om olika testteknikers effektivitet på felaktiga antaganden. För att kunna undersöka huruvida kopplingseffekten gäller eller inte, behöver man undersöka i vilken utsträckning testfall som hittar enkla fel i riktiga program också hittar komplexa fel i samma program. De studier som undersökt kopplingseffektens existens använder nästan uteslutande väldigt små program med artificiellt injicerade fel. Dessutom undersöker de typiskt program som bara innehåller ett enda fel, istället för flera fel som oftast är fallet i verkliga programvarusystem.

Detta projekt syftar till att bringa klarhet kring kopplingseffekthypotesen. Vi kommer att utföra ett antal större experimentella studier för att ge bevis för att stödja eller motbevisa hypotesen om kopplingseffektens existens. I och med detta kommer vi också undersöka huruvida en stor del av kunskapen om olika testteknikers effektivitet är tillförlitlig, eller om den vilar på en felaktig teoretisk grund. Våra studier kommer att skilja sig från existerande arbeten genom att vi undersöker riktiga och komplexa industriella programvarusystem. Programvaran kommer att innehålla riktiga icke-artificiella fel, och vi förenklar inte genom att titta på ett fel i taget. Vi planerar även att replikera dessa studier på ett eller flera större system med öppen källkod. Vi förväntar oss solid evidens av stor betydelse för forskningen inom programvarutest, liksom för användningen av de testtekniker som forskas fram.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Name of applicant
Weyuker, Elaine

Date of birth
451122-

Title of research programme
EXACT - Experimental Analysis of the Coupling Effect Hypothesis in Software Testing

Appendix A

Research programme

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

EXACT - Experimental Analysis of the Coupling Effect Hypothesis in Software Testing

Elaine Weyuker

1 Purpose and Aims

The Software Engineering research community contains many active researchers who propose interesting new testing strategies. However, before practitioners can consider using these techniques, they need convincing evidence that they are both effective at detecting bugs and can be used on large-scale, continuously-running systems. In most scientific and engineering fields, this evidence is provided by empirical studies run on artifacts and in an environment similar to the intended one. Unfortunately, in our community, while there is a growing awareness of the need for empirical studies, many are still being done using unrealistically small programs that have been created solely for the purpose of experimentation [24]. Furthermore, studied programs are often written by students rather than professional developers, and contain synthetic or seeded bugs, rather than ones that occurred naturally. The evidence provided by such studies is often unconvincing to practitioners and consequently proposed techniques do not get adopted.

In the research proposed here, we will assess a major hypothesis in the software testing research literature first introduced more than 35 years ago. Our goal is to provide concrete empirical evidence using large-scale industrial and/or open-source systems with naturally-occurring faults that either supports or refutes the hypothesis.

1.1 Background and Motivation

When testing is done in practice, it is important to have concrete and meaningful ways of determining whether it is safe to stop. While time and money are reportedly often the only criteria used, i.e., the release date has been reached or the testing budget has been expended, satisfying these criteria obviously does not guarantee that the software is bug-free, or even close to acceptable for use. Therefore, many researchers have proposed so-called test data adequacy criteria. These are rules that allow testers to determine whether their testing has been thorough, and hence it is safe to stop testing [26]. Most of these adequacy criteria involve code coverage, such as statement or branch coverage or one of the many data flow criteria [21, 22] which involve the assessment of test suites by looking at whether they cover sequences of statements.

It is easy to argue for the necessity of comprehensive control-flow and/or data-flow coverage. Clearly if some parts of the software have never been executed (i.e. covered) we have no idea whether they are doing the right thing. The important issue is whether these types of coverage criteria are sufficient, i.e. whether knowing that a test suite has exercised every statement or branch of the software implies that it has really been thoroughly tested and hence we can expect it to be close to defect-free. Therefore, when we use a coverage criterion, we are using it as a proxy for what we really want to know - does the software still contain bugs?

Mutation analysis is a technique that systematically inserts changes intended to represent simple common bugs into a subject program. Most often, they are inserted one at a time, creating a set of programs known as **mutants**. Once this set of buggy versions of a program has been created, they are run on a test suite and the tester observes whether they fail. Mutation analysis was originally introduced as a test data selection technique [10], and when used for that purpose it is commonly referred to as **mutation testing**. However, mutation analysis has also frequently been used by software researchers as a means of assessing the effectiveness of other testing techniques. If all of the mutants are "killed" or fail when they are run on the selected

test suite, then the testing technique is deemed good. If many mutants survive (i.e., don't fail) then the proposed technique is deemed ineffective.

1.2 Problem Formulation and Aims

One of the basic tenets of mutation analysis is the so-called **coupling effect hypothesis** [10]. This hypothesis states that *"Test data that distinguishes all programs differing from a correct one by only simple errors is so sensitive that it also implicitly distinguishes more complex errors."*

If the coupling effect does not hold for real software systems, then the usefulness of using mutation analysis as a way of deciding whether or not a program has been thoroughly tested is of little value. Furthermore, this would call into question the meaningfulness of research studies that used mutation analysis as a way of assessing the effectiveness of testing techniques.

There have only been a few studies that directly try to assess the coupling effect hypothesis, with most being done 25 - 30 years ago using very small synthetic programs with hypothetical bugs [15, 16, 18]. It is questionable whether such small studies provide meaningful evidence of the validity of the coupling effect hypothesis. More recently, a number of papers have considered the related question of whether mutation analysis is an appropriate way of assessing software testing strategies [2, 9, 17]. A discussion of these papers and their relevance to our proposed work is included in Section 2 below.

In this project, we propose to perform systematic empirical studies using one or more large industrial software systems with naturally-occurring faults to provide evidence to support or refute the coupling effect hypothesis. If appropriate industrial artifacts are not available, or have significant restrictions on the distribution of our research results, we will perform our studies using one or more large open-source software systems which have most of the same characteristics as (closed-source) industrial systems. Open-source systems have the additional desirable characteristic of being freely available and hence permitting easy replication of results.

2 Survey of the Field

Mutation analysis has been extensively studied in the software testing research literature for more than 35 years. A 2011 review of mutation testing by Jia and Harman [13] lists 35 empirical studies focusing on different aspects of mutation analysis, and several studies (e.g., [6, 12, 17]) have been published since [13] appeared. Relatively few of the empirical studies, however, focus specifically on the coupling effect, or the extent to which mutation testing is effective in detecting real-world faults. We will categorize these studies in Sections 2.1 through 2.4. While many of these studies do not address the coupling hypothesis directly, they all address the related questions of whether mutation analysis provides a meaningful way of either testing software or evaluating other ways of testing software.

2.1 Studies Explicitly Focusing on the Coupling Effect

To our knowledge, only three studies have explicity focused on experimentally supporting or refuting the coupling effect hypothesis. Lipton and Sayward [15] provided evidence that the coupling effect held by checking whether test suites designed to detect all single mutation faults, also detected all faults in programs containing several mutations (i.e., higher order mutants). They found that this was the case, and considered the results as evidence in favor of the coupling effect hypothesis. This work was later expanded by Offutt [18, 19], who reached a similar conclusion. Note, however, that these studies used as subject programs, tiny artifacts designed explicitly for experimentation. None of the subject programs was larger than 30 lines of code. Moreover, no real, naturally-occurring faults were used in the studies. Instead, higher-order

mutants were used as proxies for complex real-world faults. Consequently, these studies do not provide substantial evidence regarding the hypothesis.

2.2 Studies of the Appropriateness of Using Mutation Analysis to Evaluate Test Strategies

This class of studies addresses the question of whether it is appropriate to use mutation analysis to evaluate other testing techniques. It should be noted that when a study determines the success of a technique in this way, it is tacitly assuming that the coupling effect hypothesis holds since it argues that if a test suite kills most mutants it is also capable of identifying many real faults. As discussed in Section 3, one must however be careful to insure the realism of the testing environment before drawing such conclusions. Additionally, we point out that just because one mutation-adequate test set identifies many of the real faults, it does not necessarily support the hypothesis in general, since other mutation-adequate test sets may expose many fewer faults.

Several published studies focus on this question. Andrews et al. [2] report on an experiment investigating whether mutation scores (i.e., the number of killed mutants divided by the total number of mutants) are good predictors of actual fault detection ratios. The study uses the 6000 lines of European Space Agency C code as its sole subject program. The program has been heavily used in software testing research, and comes with 34 well-documented real faults. The study used 34 real-fault variants of the original space program, each containing a single injected fault. Using four standard mutant operators, 11.379 mutants of the original program were generated, of which 10% were randomly selected and used in the experiment. These mutants were executed on a large number of generated test suites.

Analysis was then performed on the difference between the fault detection ratio and the mutation detection ratio for each test suite. All differences between the two were found to be either statistically insignificant or statistically significant but of negligible magnitude. The authors concluded that the mutation score of a test suite accurately predicts its fault detection effectiveness. This result can be viewed as lending support for the coupling effect hypothesis. However the use of a single, relatively small program, does limit one's ability to extrapolate from these results to much larger systems containing many faults simultaneously.

The study by Andrews et al. was a continuation of an earlier study [1] that investigated a larger set of programs, using a less detailed analysis. The subject programs used in [1] that were excluded from [2] were smaller in size and lacked real faults. Instead, hand-seeded faults were used for those programs. Both versions of the study found that mutation scores correlated with real fault detection. Low external validity was however acknowledged by the authors, as was the fact that the subject program was small by industrial standards. One issue not discussed was the fact that, if mutation analysis is to be used in practice, faults in the program under test do not typically appear in isolation. Generally, there are multiple faults in a program entity and these faults may well interact. Therefore the fault detection ratios computed for the individually seeded real-fault variants may either under-approximate or over-approximate reality.

Andrews' experiments were revisited by Namin and Kakarla [17] in an effort to investigate the threats to external validity related to using mutation analysis in testing experiments. The authors replicated the experiments from [1] using a more comprehensive set of mutation operators and also used a set of Java programs. Their main finding was that the use of mutation analysis in testing experiments is highly sensitive to external threats. In particular, they found that the extent to which the mutation score and fault detection correlate strongly depends on factors like test suite size, programming language, and mutation operators used.

Finally, Do and Rothermel [11] experimentally studied the effectiveness of regression test-

ing prioritization techniques using a variety of mostly small or medium-size Java and C/C++ programs. They evaluated the prioritization schemes' effectiveness in terms of the fault detection ability using two different fault seeding techniques: hand-seeded faults and mutation faults. Their conclusions focus on the effectiveness of prioritization algorithms and the cost savings that can be effected using mutation faults rather than hand-seeded faults in experimentation. As such, they do not provide evidence to either support or refute the hypothesis.

2.3 Studies Focusing on the Effectiveness of Mutation Testing

Daran and Thévenod-Fosse [9] performed a study investigating the relationship between error behaviors resulting from "real faults", and error behaviors resulting from mutants. The authors began with a specification for software from the civil nuclear field, for which a student wrote roughly 1.000 lines of code containing 12 "real faults" which were the subject of this study. While they found that 85% of the error behaviors produced by these faults could also be produced by a set of 24 generated mutants, the extent to which student-created faults are representative of faults found in professionally-written software is unknown. Moreover, the result that mutants and real faults give rise to the same error behaviors does not imply that a test suite that detects most of the mutants will also detect most of the real faults. In fact, the authors state that their study *"is not intended to investigate whether or not mutations constitute a fault model representative of real faults. Indeed, the answer to this question is likely to be negative"*. Hence, the results of the study does not provide evidence supporting the coupling effect hypothesis.

More recently, Baker and Habli [6] investigated the use of mutation testing as a means of complementing more traditional coverage-based testing in safety-critical software. The study used 47 Ada and C "code items" for avionics software varying in size from 3 to 46 lines of code. The study did not investigate the relation between mutants and real faults, but rather investigated which mutants were undetected by traditional testing, and why. While the study indicated that there exist cases for which mutation analysis and testing can be beneficial for revealing certain types of faults not easily found by coverage-based testing or manual review, its results again do not support the validity of the coupling effect hypothesis.

2.4 Theoretical Studies of the Coupling Effect

Theoretical studies of the coupling effect are limited, but a handful of papers have been published on this topic. Examples of theoretical analyses of the coupling effect include the works of Morell [16], Wah [25] and Kapoor [14]. All of these contributions provide formal proofs or theoretical support of the coupling effect under a set of highly limiting assumptions on the faults and programs involved. For example, Wah's work assumes the use of a single test, and Kapoor's work is restricted to some classes of logical faults. None of these papers provide significant information relevant to whether or not the coupling effect hypothesis holds for real programs.

2.5 Summary and Observations

The validity of any empirical result on the coupling effect hypothesis is inherently limited by the extent to which the subject programs used are representative of real-world programs, and by the extent to which the faults used are representative of naturally-occurring faults. When considering the existing body of research, while empirical studies related to the validity of the coupling effect are not uncommon, nearly all of these studies make use of **small programs** as their subjects, often written **explicitly for experimentation**, and/or contain faults that are **artificially generated** for the sake of the experiment. In addition, they typically examine versions of the subject programs which contain just a **single fault**, rather than multiple faults as one typically finds in real software systems. Consequently, the extent to which the results

of prior studies extend to large software systems with naturally-occurring faults has not been clearly established.

3 Project Description

In this project, we will perform systematic empirical studies to provide concrete evidence to support or refute the coupling effect hypothesis. We will do so by investigating the coupling effect in several different ways. Each experimental procedure variant will follow a similar sequential process beginning with some initial version of the software, followed by **mutant creation**, **test set creation**, and **fault detection analysis**.

Our process starts with a version of a large industrial or open-source software system containing naturally-occurring faults. Because we will be using a real, fully-tested system as our study subject, we will have a "correct version" of the software, namely the one from which all faults that have been identified have been removed prior to field release. We will refer to this as the **correct version** in the remainder of the proposal. We will also have one or more **faulty versions** of the software, from which the known faults have not yet been removed.

Next, we will select a set of first-order mutation operators that are appropriate for the software's implementation language and create a family of mutants by applying each of the mutation operators individually. We will start with either the correct version of the software or the faulty version, depending on the goal of the study. Initially we will apply each mutation operator at the individual unit level for each unit comprising the system under study.

Using a greedy algorithm, we will then create a test set to kill as many mutants as possible. Our process will be to select a test case, run each of the mutants on the test case and mark as killed, each mutant that produces an incorrect answer. We will then add a second test case, and see which additional mutants are killed. If a selected test case does not kill any mutants, it is not added to the test set. We iterate the process until all inequivalent mutants have been killed.

There are two subcases depending on whether we start with the correct or faulty version of the software. In the former case, each mutant contains exactly one fault, namely the one that has been deliberately inserted into the correct version of the software. This is the process that has been used in most mutation studies to date. In the latter case, each mutant will typically contain multiple real faults plus one additional seeded fault. Note that if mutation testing is to be used in the field, the faulty version is the only version that exists. For each variant of our studies, we will note whether mutants will be created from the correct or faulty version.

The test set that contains the test cases that killed all the inequivalent mutants will be called **mutation-adequate**, and will be near-minimal in the sense that every test case in the set will have been added because it killed a mutant that had not been previously killed. Once we have created the mutation-adequate test set we will run the original faulty program on this set to see whether each of the real faults is elicited, leading to failures.

Based on this general experimental procedure, we propose several variant studies to assess the validity of the coupling effect hypothesis (see Figure 1 for an overview of the variants).

We note that for those cases in which we are evaluating the coupling effect hypothesis based on the percentage of real faults exposed by a mutation-adequate test set, we need to take the following into account. If only a small or moderate percentage of the faults in the real system are found using the mutation-adequate test set, we will have concrete evidence that the coupling effect hypothesis does not hold. If, however, all or most of the faults are exposed by the test set, then the coupling effect hypothesis will have been shown to hold **for this software system and this mutation-adequate test set**. However, in order to provide substantial evidence of the general acceptance of the hypothesis, or even for this particular system, it will be necessary to

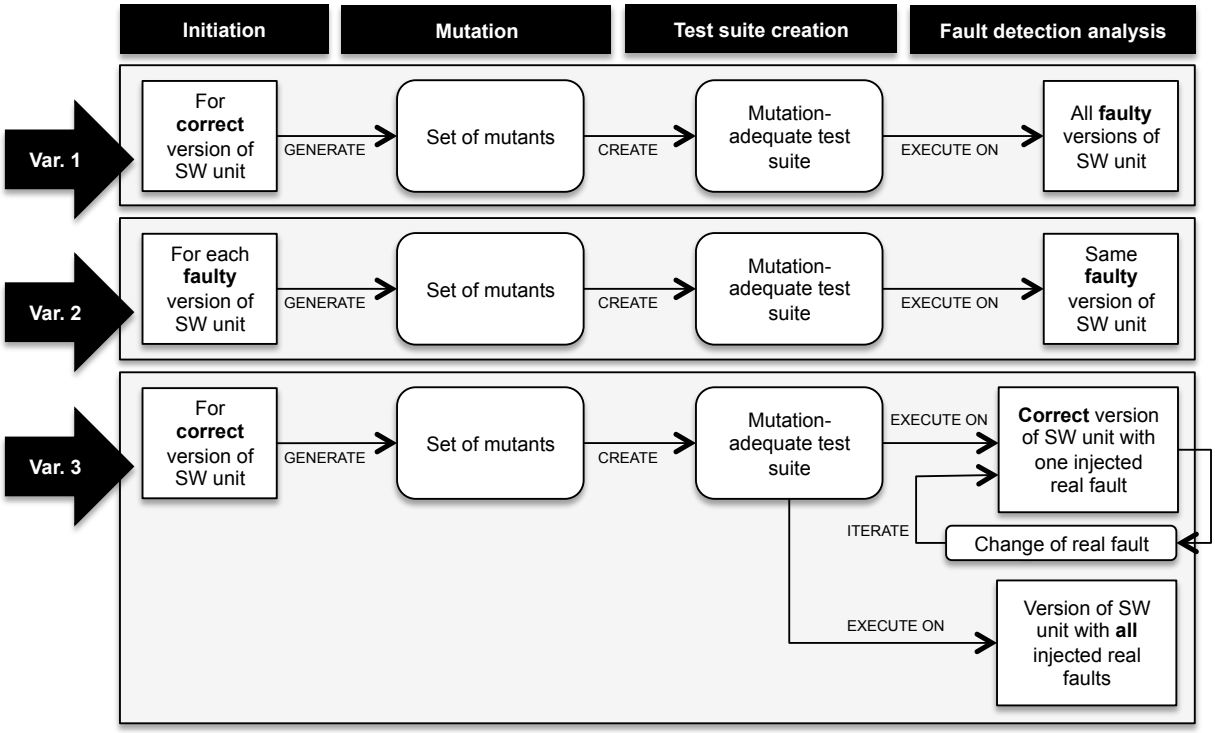

Figure 1: Experimental Procedure and Variants.

consider multiple mutation-adequate test sets, and multiple orderings of a given test set since the order in which faults are removed from the system may impact the number of faults identified. Therefore, in many of our studies it will be necessary to generate multiple mutation-adequate test sets and have a process for modifying the order in which test cases are run.

Var. 1: Our first and primary variant is to determine what proportion of the real faults in a system have been detected by a mutation-adequate test set.

In industrial settings, it is common to keep track of all test cases, their results when run, a determination of whether the results were correct, and if the program failed on a test case, what the fault was determined to be. In addition, it is common to have a version control system that tracks each change to the system as it is debugged. As testing progresses, each successive version of the system ideally has a smaller number of faults, although sometimes debugging leads to the addition of new faults, which then have to be identified.

For each available version of the subject system, we will run the faulty version on the mutation-adequate test set created from the correct version, until no more faults are exposed by the test set. These different versions represent the software as faults are successively removed. As noted in the general process outlined at the beginning of this section, while a small or moderate percentage of the faults being identified by a mutation-adequate test set provides concrete evidence that the coupling effect hypothesis **does not hold**, the opposite is not necessarily true. Even if a large percentage of faults are exposed by a particular mutation-adequate test set, other ones or other fault removal orders may be less successful. Therefore, if our studies do not provide refutation of the hypothesis, multiple studies must be performed to provide sufficient support for the hypothesis.

Var. 2: The second variant of our research will consider mutation analysis when it is used for test set design (i.e., when performing mutation testing). In this case, there is no existing correct version of the program since any test case selection strategy starts with a faulty version of a program and uses the test cases to find and remove the faults. If one is to investigate the validity of the coupling effect in this context, it is necessary to create mutants based on the faulty version of the program, create test cases that kill those mutants, and again execute those test cases on the unmutated faulty version of the software as outlined above to see whether the original faults are uncovered by the mutation-adequate test set.

Var. 3: The third variant of our research will involve determining the appropriateness of the process of assessing the fault detection rate based on program versions containing individual faults rather than the multiple simultaneous faults in the original faulty version. This process was used by a number of research groups including [2, 9, 17]. We will do this by comparing the percentage of faults identified by a mutation-adequate test set when real faults are inserted individually into the code, with the percentage of faults identified by the same mutation-adequate test set when all the real faults occur simultaneously in the code. If the percentage of faults found when they occur in the code simultaneously is substantially lower than the percentage of faults found when they are individually reinserted, that will call into question the appropriateness of this process. If the percentages are similar, it lends credence to this way of using mutation analysis to evaluate other testing techniques. One of the challenges of this phase of the research will be devising a measure of the percentage of faults identified when all are in the code simultaneously.

Var. 4: One other issue needs to be addressed, namely the testing level, and hence the software artifacts, that will be used. Mutation analysis was designed to be used at the unit testing level, applied to small to medium-sized units. Therefore, mutation operators and mutants will be applied to each such unit in the larger subject system during our empirical studies.

However, we would also like to develop a process for determining whether the coupling effect holds at the system level for large systems. Often during system testing, a tester encounters faults spread across multiple units. Therefore, it would be valuable to understand whether test sets that identify faults represented by single mutations are able to identify the complex faults that are encountered during system testing or when a system fails in the field. This will be investigated during the final phase of this research. Because the detailed development of the experimental strategy is part of the research planned in the project, it does not appear in Figure 1.

4 Significance

The coupling effect hypothesis was originally proposed more than 35 years ago by DeMillo et al. [10]. During all the intervening years and despite hundreds of research papers written on mutation analysis, there have been few empirical assessments aimed specifically at supporting or refuting the coupling effect hypothesis. In those studies, either the programs were small or written specifically for experimentation, or contained synthetic faults or the real faults were inserted individually, rather than having the software contain multiple faults at once as happens in practice. In contrast, our proposed studies will provide evidence to support or refute the hypothesis under realistic conditions.

As Jia and Harman [13] discussed in their recent comprehensive survey article, there has been extensive and growing interest in mutation analysis among software testing researchers.

Furthermore, Andrews et al. [2] investigated whether a high mutation score is a good indication that a test set will uncover most real faults in the software. They list a substantial number of papers that use some form of mutation analysis to assess the effectiveness of newly-designed testing strategies. Therefore, our research will provide a clear benefit to the software testing research community by providing evidence whether this is a satisfactory way to assess their work. Additionally, if mutation testing is to ever become a practical technique that can be used by software testing practitioners, there must be a firm basis for believing that mutation-adequate test sets are likely to uncover all or most faults in real software systems. Our results should also clarify that issue.

5 Preliminary Results

Elaine Weyuker has published more than 160 refereed papers on all aspects of software testing, evaluation and measurement during the past 35 years, with much of her recent work focusing on large industrial empirical studies using software systems containing naturally-occurring faults, as a way of validating her work.

In particular, for the past decade, much of her research has focused on the development of accurate statistical models that can be used to predict where faults will be in the next release of large industrial software systems. Her research group built a fully-automated tool to make the predictions that was designed for practicing software testers. It does not require any knowledge of statistics, modeling, or the relevant code and history characteristics used to make the predications. She has performed empirical studies on a total of 170 different releases of nine industrial systems, with different characteristics in such areas as functionality, programming languages used, size, years in the field, numbers of faults, and development process. The systems studied ranged from 281,000 executable lines of code (ELOC), to over 2,116,000 ELOC. A sampling of the more than 25 papers published describing their fault prediction research containing industrial empirical studies include [7, 8, 20, 23, 27, 28].

In [20], for example, the team applied their fault prediction model to two large systems. The first contained 538,000 ELOC and had 17 quarterly releases for which they made predictions. The second system contained 438,000 ELOC and 9 releases. For each release of these systems, they used their model to identify the 20% of the files predicted to have the largest numbers of faults. They then identified the percent of real faults contained in each of those high-fault files during the appropriate release. They found that the 20% of the files predicted to contain the largest numbers of faults actually contained between 71% and 92% of the faults that were detected during late-stage testing and field release, with an overall rate of 83%.

In addition, Weyuker has published widely in the area of software performance testing including large industrial empirical studies. A sampling of these research studies include [3, 4, 5].

Daniel Sundmark has published over 40 refereed papers on different aspects on software engineering, most notably in the field of software testing. He has experience doing large-scale industrial empirical studies on software testing and quality assurance conducted at Scania CV, Bombardier Transportation, ABB Robotics, Infosys, and Ericsson.

6 Collaboration

This project application aims at an extension and an intensification of the partnership between Prof. Weyuker and Mälardalen University. Through the support of the Swedish Knowledge Foundation, Elaine Weyuker currently serves as a visiting professor at Mälardalen University from September 2013 - August 2014. In this role, Weyuker has been collaborating with Prof. Paul Pettersson, Assoc. Profs. Daniel Sundmark and Cristina Seceleanu, Postdocs Adnan Čaušević,

Wasif Afzal and Guillermo Rodriguez-Navas, and Ph.D. Students Eduard P. Enoiu and Raluca Marinescu. As a visiting professor, Weyuker has also taken part in collaborations between Mälardalen University and industrial partners Bombardier Transportation and Ericsson. In particular the ongoing work with PhD student Eduard Enoiu and PostDoc Adnan Čaušević on empirical studies performed at Bombardier Transportation, using mutation analysis will likely contribute to the results of this project. A mutation generation tool is currently being designed for a language commonly used for embedded systems. Weyuker and Sundmark intend to involve these researchers, plus one or more PhD students in the proposed research.

On a national level, Prof. Weyuker and Dr. Sundmark collaborate with members of TOCSYC, a Swedish distributed research environment on software testing headed by Prof. Paul Pettersson at Mälardalen University, and funded by the Swedish Knowledge Foundation. Members of TOCSYC include notable testing researchers such as Prof. Robert Feldt (Blekinge Institute of Technology) and Assoc. Prof. Richard Torkar (Chalmers), Prof. Jeff Offutt (George Mason University and University of Skövde), Dr. Simon Poulding (Blekinge Institute of Technology), Dr. Sigrid Eldh (Ericsson AB), and Dr. Thomas Ostrand (DIMACS, Rutgers University).

In addition to the above, Weyuker has had extensive international collaborations in research on software testing for several decades. Most notably, Weyuker and Dr. Thomas Ostrand (DIMACS, Rutgers University) have co-authored many software testing and fault prediction papers over a period of 35 years. Robert Bell, a statistician at AT&T Labs, collaborated on some of this research. Weyuker has also collaborated extensively with Alberto Avritzer, a researcher at Siemens Corporate Research in Princeton, NJ, on areas of software performance testing, rejuvenation, and scalability over the past 20 years. Throughout her career, Weyuker has collaborated with several dozen researchers.

References

[1] J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 402–411, New York, NY, USA, 2005. ACM.

[2] James H. Andrews, Lionel C. Briand, Yvan Labiche, and Akbar Siami Namin. Using mutation analysis for assessing and comparing testing coverage criteria. *IEEE Transactions on Software Engineering*, 32(8):608–624, 2006.

[3] Alberto Avritzer, André Bondi, and Elaine J. Weyuker. Ensuring system performance for cluster and single server systems. *J. Syst. Softw.*, 80(4):441–454, April 2007.

[4] Alberto Avritzer, Robert G. Cole, and Elaine J. Weyuker. Methods and opportunities for rejuvenation in aging distributed software systems. *J. Syst. Softw.*, 83(9):1568–1578, September 2010.

[5] Alberto Avritzer and Elaine J. Weyuker. The role of modeling in the performance testing of e-commerce applications. *IEEE Trans. Softw. Eng.*, 30(12):1072–1083, December 2004.

[6] Richard Baker and Ibrahim Habli. An empirical evaluation of mutation testing for improving the test quality of safety-critical software. *IEEE Trans. Softw. Eng.*, 39(6):787–805, June 2013.

[7] R.M. Bell, E.J. Weyuker, and T.J. Ostrand. Assessing the impact of using fault prediction in industry. In *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*, pages 561–565, March 2011.

[8] RobertM. Bell, ThomasJ. Ostrand, and ElaineJ. Weyuker. The limited impact of individual developer data on software defect prediction. *Empirical Software Engineering*, 18(3):478–505, 2013.

[9] Murial Daran and Pascale Thévenod-Fosse. Software error analysis: A real case study involving real faults and mutations. In *Proceedings of the 1996 ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA '96, pages 158–171, New York, NY, USA, 1996. ACM.

[10] R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, April 1978.

[11] Hyunsook Do and Gregg Rothermel. On the use of mutation faults in empirical assessments of test case prioritization techniques. *IEEE Trans. Softw. Eng.*, 32(9):733–752, September 2006.

[12] Gordon Fraser and Andrea Arcuri. Achieving scalable mutation-based generation of whole test suites. *Empirical Software Engineering*, pages 1–30, 2014.

[13] Yue Jia and M. Harman. An analysis and survey of the development of mutation testing. *Software Engineering, IEEE Transactions on*, 37(5):649–678, Sept 2011.

[14] Kalpesh Kapoor. Formal analysis of coupling hypothesis for logical faults. *Innovations in Systems and Software Engineering*, 2(2):80–87, 2006.

[15] Richard J. Lipton and Frederick Gerald Sayward. The status of research on program mutation. In *Proceedings of the Workshop on Software Testing and Test Documentation*, pages 355–373, December 1978.

[16] Larry Joe Morell. *A Theory of Error-Based Testing.* phdthesis, University of Maryland at College Park, College Park, Maryland, 1984.

[17] Akbar Siami Namin and Sahitya Kakarla. The use of mutation in testing experiments and its sensitivity to external threats. In *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, ISSTA '11, pages 342–352, New York, NY, USA, 2011. ACM.

[18] A. Offutt. The coupling effect: Fact or fiction. In *Proceedings of the ACM SIGSOFT '89 Third Symposium on Software Testing, Analysis, and Verification*, TAV3, pages 131–140, New York, NY, USA, 1989. ACM.

[19] A. Jefferson Offutt. Investigations of the software testing coupling effect. *ACM Trans. Softw. Eng. Methodol.*, 1(1):5–20, January 1992.

[20] Thomas J. Ostrand, Elaine J. Weyuker, and Robert M. Bell. Predicting the location and number of faults in large software systems. *IEEE Trans. Softw. Eng.*, 31(4):340–355, April 2005.

[21] S. Rapps and E.J. Weyuker. Selecting software test data using data flow information. *Software Engineering, IEEE Transactions on*, SE-11(4):367–375, April 1985.

[22] Sandra Rapps and Elaine J. Weyuker. Data flow analysis techniques for test data selection. In *Proceedings of the 6th International Conference on Software Engineering*, ICSE '82, pages 272–278, Los Alamitos, CA, USA, 1982. IEEE Computer Society Press.

[23] Yonghee Shin, Robert M. Bell, Thomas J. Ostrand, and Elaine J. Weyuker. On the use of calling structure information to improve fault prediction. *Empirical Softw. Engg.*, 17(4-5):390–423, August 2012.

[24] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.-K. Liborg, and A.C. Rekdal. A survey of controlled experiments in software engineering. *Software Engineering, IEEE Transactions on*, 31(9):733–753, Sept 2005.

[25] K.S. How Tai Wah. An analysis of the coupling effect i: single test data. *Science of Computer Programming*, 48(23):119 – 161, 2003.

[26] E J Weyuker. Axiomatizing software test data adequacy. *IEEE Trans. Softw. Eng.*, 12(12):1128–1138, December 1986.

[27] Elaine J. Weyuker, Thomas J. Ostrand, and Robert M. Bell. Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models. *Empirical Softw. Engg.*, 13(5):539–559, October 2008.

[28] Elaine J. Weyuker, Thomas J. Ostrand, and Robert M. Bell. Comparing the effectiveness of several modeling methods for fault prediction. *Empirical Softw. Engg.*, 15(3):277–295, June 2010.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Name of applicant
Weyuker, Elaine

Date of birth
451122-

Title of research programme
EXACT - Experimental Analysis of the Coupling Effect Hypothesis in Software Testing

Appendix B

Curriculum vitae

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Elaine J. Weyuker

phone: 732 549 8118
cell: 908 612 4906
email: weyuker@gmail.com
homepage: http://elaineweyuker.yolasite.com

Education

- Ph.D., Computer Science, Rutgers University, 1977.
- M.S.E., Computer & Information Sciences, Moore School Electrical Engineering, University of Pennsylvania, 1968.
- B.A., Mathematics, Harpur College, State University of New York at Binghamton, 1966.

Experience

- 2013-present, Visiting Professor, Mälardalen University, Västerås, Sweden
- 2012-present, Independent Researcher and Consultant
- 1996–2012, Distinguished Member of Technical Staff and AT&T Fellow, AT&T Labs, Florham Park, NJ.
- 1993–1996, Distinguished Member of Technical Staff, AT&T Bell Labs, Murray Hill, NJ.
- 1977–1995, Professor of Computer Science, Courant Institute of Mathematical Sciences, New York University.
- 1969–1975, Lecturer in Computer Science, Richmond College, City University of New York.
- 1968–1969, Systems Engineer, I.B.M.

Professional Leadership

- Member of several National Academy of Sciences study panels, providing guidance to both government and industry.
- Director of Graduate Studies, Computer Science Department, NYU, 1985-1988. Responsible for most aspects of the 700 student Masters and PhD programs.
- Chair of the ACM Women's Council, 2004-2012. ACM-W is a large volunteer organization supported by the Association of Computing Machinery aimed at promoting activities that result in more equal representation of women in Computer Science.

Major Honors

- Member of the US National Academy of Engineering. (2002)
- ACM Fellow. (1997)
- IEEE Fellow. (2003)

Major Awards

- ACM President's Award. For tireless efforts in the development and growth of the ACM Women's Council. (2010)
- ACM SIGSOFT Retrospective Impact Paper Awards. For a highly influential paper that has continued to have impact on the field for more than 25 years. (2009)
- Anita Borg Institute, Technical Leadership Award. For outstanding research and technical leadership. (2008)
- ACM SIGSOFT Outstanding Research Award. For deep and lasting contributions and impact to software engineering as a discipline. (2007)
- IEEE Harlan D. Mills Award. For long-standing, sustained, and meaningful contributions to the theory and practice of the information sciences. (2004)
- AT&T Chairman's Diversity Award. (2004)
- Rutgers University 50th Anniversary Distinguished Alumni Award. For outstanding accomplishments and leadership in mentoring of women and minorities. (2003)
- YWCA Woman of Achievement Award. (2001)
- AT&T Fellow. (2000)

Recent Keynote Addresses and Invited Lectures

- Looking for Bugs in All the Right Places, Invited Lecture, 2013 Grace Hopper Celebration of Women in Computing, Oct 2013,
- Three Stories, Three Endings, Some Morals, Keynote Address, IEEE TAIC-PART Conference March 2013.
- You Can Run, But You Cannot Hide, Distinguished Lecture Series, University of Delaware, 2012.
- Women in Computing - What Do We Have to Look Forward To? Invited Lecture, Northern New Jersey Branch of the Women in Computing, 2011.
- How Well Can We Predict Future Bugs? Invited Lecture, Ericsson Corp, Sweden, 2011.
- Empirical Software Engineering Research - The Good, The Bad, The Ugly. Keynote Address, Empirical Software Engineering Symp, Banff, Canada, 2011.
- The Challenges of Doing Large Scale Empirical Studies in an Industrial Environment. Keynote Address, i-Promise Conference, Toronto, Canada, 2011.
- Predicting Faults in Large Software Systems. Invited Lecture, Princeton ACM/IEEE meeting, 2011.
- The Exterminator - Helping to Catch Those Bugs, Invited Lecture, Stevens Institute of Technology, 2010.
- Predicting Where Bugs Are - A Statistical Model and Tool, Invited Lectures, Summer School in Software Engineering, Salerno, Italy, 2010.
- Finding the Needle in the Haystack, Annual Distinguished Lecture, Seton Hall Univ, 2010.
- Faulty Towers - Software Defect Prediction Models, CREST Wksp, Kings College London, UK, Keynote, 2009.
- Bugs - Find Them Before They Find You! Grace Hopper Distinguished Lecture, University of Pennsylvania, 2009.
- Invited Series of Lectures, LASER Summer School, Elba, Italy, 2009.

Professional Activities

- Member US National Academy of Sciences Committee to Evaluate the Next Generation Software of the US Air Transportation System for the FAA (2012 – present)
- Member of the Rutgers University Graduate School Advisory Board. (2004-present)
- Member of the Executive Board of the Coalition to Diversify Computing (2003-present)
- Chair ACM Women's Council (2004-2012)
- ACM Council (2008-2012)
- Member of the Stevens Institute of Technology President's Advisory Board on the Status of Women (2012)
- Editorial Board, Journal of Empirical Software Engineering (1995-present)
- Advisory Editorial Board, Journal of Software and Systems (1997-present)
- Editorial Boards: IEEE Transactions on Software Engineering (2004-2007), IEEE Transactions on Dependable and Secure Computing (2004-2008), IEEE Spectrum (2004 - 2007), ACM Transactions on Software Engineering and Methodology (1989-2001)
- Member, Women in Academia Committee of the National Academies (2005-2007)
- Member of the Board of Directors of the Computing Research Association (2000-2005)
- Association for Computing Machinery (ACM) Fellow Selection Committee (1998-2003), Chair (2002-3).
- Secretary/Treasurer Association for Computing Machinery (ACM) SIGSOFT (1989-1992)
- Member of the Executive Committee of the IEEE Computer Society Technical Committee on Software Engineering (1984-1987, 1991-1995)
- IEEE Senior member Selection Committee (2001)
- Member ACM Committee on the Status of Women and Minorities in Computer Science (1990-1995)
- Member Computing Research Association Committee on the Status of Women in Computer Science (CRA-W) (1990-1993)

Curriculum Vitae, Dr. Daniel Sundmark

Karl Daniel Sundmark (April 9, 1975)
School of Innovation, Design and Engineering, Mälardalen University
+46-21-10 31 45 (office), Daniel.Sundmark@mdh.se
http://www.idt.mdh.se/∼dsk01/

Professional Preparation:

- Docent in Computer Science, Mälardalen University, 2013.
- PhD in Computer Science, Mälardalen University, 2008.
- Licentiate in Computer Science, Mälardalen University, 2004.
- Master of Science in Information Technology, Uppsala University, 2002.

Positions:

- Senior Lecturer in Software Engineering, at Mälardalen University, Sweden (since 2010; on 100% parental leave Apr. 2012 to Aug. 2012).
- Senior Researcher, Software and Systems Engineering (SSE) Laboratory, SICS Swedish ICT, Sweden (part-time, since 2011)
- Researcher in Software Engineering, at Mälardalen University, Sweden (2007–2010; on 20% parental leave Sep 2008 to Jun. 2009).
- PhD student, at Mälardalen University, Sweden (2001–2007).
- Software Developer, ZealCore Embedded Solutions, Sweden (2001)

Appointments:

- Head of Software Testing Laboratory research group at Mälardalen University (since 2013).
- Member of the Faculty Board of Natural Sciences and Engineering at Mälardalen University (2004–2007).
- Chair of the PhD student council at Mälardalen University (2005–2006).

Thesis supervision:

- Co-advisor: Adnan Čaušević (Ph.D. Thesis in 2013), Mathias Ekman (since 2011), Kristian Wiklund (since 2011), Eduard Enoiu (since 2011), and Sara Abbaspour (since 2013).

Other merits of relevance:

- Community services:
  - Program committee member for the International Conference on Lean Enterprise and Software Systems (LESS 2010), the International Conference on Software Engineering Advances (ICSEA 2011), the International Conference on Agile Software Development (XP 2011), and the Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2013).
  - Session co-chair in the 13th International IEEE Conference on Emerging Technologies in Factory Automation (ETFA), as well as in the International Conference on Software Engineering Advances (ICSEA 2010).
  - Referee for several journals, including the IEEE Transactions on Software Engineering, the Springer Journal of Software and Systems Modeling, the IEEE Transactions on Industrial Informatics, Elseviers Journal on Information and Software Technology, and Elseviers Science in Computer Programming.
  - Reviewer for several major conferences and workshops, including ASE, RTSS, EMSOFT, and ICST.
  - Invited as guest speaker on the topic of software testing by industries and associations like SAST, MINST Innovation, Öhlins racing, Scania CV, Maquet Critical Care, and IBC Euroforum.
  - Opponent at public defenses: Ph.D. Nina Elisabeth Holt (University of Oslo, 2012), Fil.lic. Jesper Pedersen Notander (Lund University, 2013), and Andreas Eriksson (half-time seminar, University of Skövde, 2014).
- Industrial Visits and Collaboration:
  - 2012–2013: Main lecturer for five instances of a one-week contract education course in software testing for Ericsson AB.
  - Jan. 2010 - Jun 2011: Industrial visit (50%) at Scania CV, Södertälje. Industrial research, focusing on test- and test process improvement within the Scania electrical system development process.
  - Oct. 2007 May 2008: Industrial visit (40%) at ABB Corporate Research, Västerås. Industrial research, focusing on distributed large-scale agile development processes.
- Funded Applications:
  - PINT (Parallelization of Integration Tests), Scania/SICS project funded by Vinnova/FFI, 2.1 MSEK, Sept 2013 Aug 2015 (as main applicant)
  - TOCSYC (Testing of Critical System Characteristics) phase 1, SIDUS Distributed Research Environment funded by KKS, 0.5 MSEK, Nov 2012 Mar 2013 (as co-applicant)
  - TOCSYC (Testing of Critical System Characteristics) phase 2, SIDUS Distributed Research Environment funded by KKS, 27 MSEK, Sept 2013 Aug 2018 (as co-applicant)
- Author/co-author of over 40 scientific publications accepted and presented at international journals, conferences and workshops.

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Name of applicant
Weyuker, Elaine

Date of birth
451122-

Title of research programme
EXACT - Experimental Analysis of the Coupling Effect Hypothesis in Software Testing

Appendix C

List of publications

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

Elaine J. Weyuker - List of Publications

Elaine J Weyuker is the author/co-author of over 170 peer-reviewed journal conference/workshop articles. Her works have been cited 8913 times and her h-index is 42. Further and updated publication and citation data is available at http://scholar.google.com/citations?user=xXjQoLIAAAAJ&hl=en. All citation data was obtained from Google Scholar April 4, 2014.

The following is a list of the peer-reviewed journal articles, international conference and workshop articles written by Elaine Weyuker during the last eight years. In accordance with the directions, the five most important publications for the project have been marked (*). The indicated number of citations for each publication has been obtained from Google Scholar.

1. Peer-reviewed original articles

1. T Ostrand, RM Bell and EJ Weyuker. The Impact of Individual Developers on Software Defect Prediction. *Empirical Software Eng. Journal*, Vol18, No3, June 2013.
   Number of citations: 2
2. Y Shin, RM Bell, TJ Ostrand, EJ Weyuker. On the Use of Calling Structure Information to Help Predict Software Fault Proneness. *Empirical Software Eng. J*, Vol17, No4-5, Aug 2012.
   Number of citations: 4
3. A Avritzer, ES e Silva, RMM Leão and EJ Weyuker. Automated Generation of Test Cases Using using a Performability Model, Special Issue of *IET Journal*, 2011.
   Number of citations: 3
4. A Avritzer, RG Cole and EJ Weyuker. Methods and Opportunities for Rejuvenation in Aging Distributed Software. *Journal of Systems and Software*, vol 83, no9, 2010, pp. 1568 – 1578.
   Number of citations: 5
5. EJ Weyuker and TJ Ostrand and RM Bell. Comparing the Effectiveness of Several Modeling Methods for Fault Prediction *Empirical Software Eng. Journal*, June 2010. (*)
   Number of citations: 23
6. EJ Weyuker, TJ Ostrand and RM Bell. Do Too Many Cooks Spoil the Broth? Using the Number of Developers to Enhance Defect Prediction Models. *Empirical Software Eng. Journal*, October 2008. (*)
   Number of citations: 76
7. A Avritzer, A Bondi and EJ Weyuker. Ensuring System Performance for Cluster and Single Server Systems. *Journal of Systems and Software*, Vol 80,No. 4, April, 2007.
   Number of citations: 11

2. Peer-reviewed conference contributions

1. TJ Ostrand and EJ Weyuker. Can File Level Characteristics Help Identify System Level Fault-Proneness? *Proc. Haifa Verification Conference 2011 (HVC11*), Haifa, Israel, December 2011.
   Number of citations: 0
2. TJ Ostrand and EJ Weyuker. Predicting Bugs in Large Industrial Software Systems. ISSSE, Vol 7171, Lecture Notes in Computer Science, pp.71-93,

Springer 2011.
Number of citations: 0

3. RM Bell, TJ Ostrand and EJ Weyuker. Does Measuring Code Change Improve Fault Prediction? *Proc. 7th Int'l Conference on Predictive Models in Software Engineering (Promise2011*), Banff, Canada, September 2011.
Number of citations: 9
4. EJ Weyuker. Empirical Software Engineering Research - The Good, The Bad, The Ugly. *Proc. IEEE Int'l Symp on Empirical Software Engineering and Measurement*, Banff, Canada, Sept 2011.
Number of citations: 4
5. EJ Weyuker, RM Bell and TJ Ostrand. Replicate, Replicate, Replicate. *Proc. Third Int'l Conf on Software Testing, Verification*, Banff, Canada, Sept 2011.
Number of citations: 1
6. RM Bell, EJ Weyuker and TJ Ostrand. Assessing the Impact of Using Fault-Prediction in Industry. *Proc Testing: Academic & Industrial Conference (TAIC 2011)*, Berlin, Mar 2011.
Number of citations: 1
7. T Ostrand and E Weyuker. Software Testing Research and Software Engineering Education. *Proc. Wksp Future of Software Eng and Research Santa Fe,* NM, November 2010.
Number of citations: 5
8. TJ Ostrand, EJ Weyuker and RM Bell. Programmer-based Fault Prediction. *Proc. Int'l Conference on Predictive Models (PROMISE10)*, Romania, September, 2010.
Number of citations: 15
9. TJ Ostrand and EJ Weyuker. Software Fault Prediction Tool. *Proc. Internat'l Symp on Software Testing and Analysis (ISSTA2010)*, Trento, Italy, July 2010.
Number of citations: 5
10. EJ Weyuker, RM Bell and TJ Ostrand. We're Finding Most of the Bugs, but What Are We Missing? *Proc. Third Internat'l Conf on Software Testing, Verification and Validation (ICST2010*), Paris, France, April 2010.
Number of citations: 14
11. A Avritzer, R Tanikella, K James, RG Cole and E Weyuker. Monitoring for Security Intrusion using Performance Signatures *Proc. ACM/WOSP-SIPEW 2010*, Jan 2010.
Number of citations: 7
12. A Avritzer and EJ Weyuker. The Automated Generation of Test Cases using an Extended Domain Based Reliability Model *Proc. Wksp on the Automation of Software Test (AST 2009)*. Vancouver, BC, Canada, May 2009
Number of citations: 3
13. Y Shin, R Bell, T Ostrand and E Weyuker. Does Calling Structure Information Improve the Accuracy of Fault Prediction? *Proc. Sixth Int'l Working Conference on Mining Software Repositories (MSR 2009*). Vancouver, BC, Canada, May 2009
Number of citations: 23
14. A Avritzer, RG Cole and EJ Weyuker. Methods and Opportunities for Rejuvenation in Aging Distributed Software Systems. *Proc. Int'l Wksp on Software Aging and Rejuvenation*, Seattle, Washington, November, 2008.
Number of citations: 0
15. TJ Ostrand and EJ Weyuker. Progress in Automated Software Defect Prediction. *Proc. Haifa Int'l Conference, Haifa*, Israel October 2008.
Number of citations: 0

16. EJ Weyuker. Comparing the Effectiveness of Testing Techniques. in *Formal Methods and Testing*, R. Hierons, J. Bowen, M. Harman, eds. Published as Lecture Notes in Computer Science 4949, Springer, 2008.
    Number of citations: 8
17. EJ Weyuker and TJ Ostrand. Comparing Methods to Identify Defect Reports in a Change Management Database. *Proc. Defects in Large Software Systems Wksp (DEFECTS08*), Seattle, Washington, July 2008.
    Number of citations: 7
18. EJ Weyuker, TJ Ostrand and RM Bell. Comparing Negative Binomial and Recursive Partitioning Models for Fault Prediction. *Proc. Predictive Models in Software Engineering (PROMISE'08*), Leipzig, Germany, May 2008.
    Number of citations: 8
19. EJ Weyuker and TJ Ostrand. What Can Fault Prediction Do For YOU? *Proc. Second Int'l Conference on Tests and Proofs (TAP08)*, Prato, Italy, April 2008. Published as *Lecture Notes in Computer Science 4966*, Springer.
    Number of citations: 3
20. EJ Weyuker. Software Engineering Research - From Cradle to Grave. *Proc. 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symp on the Foundations of Software Engineering*, Dubrovnik, Croatia, Sept 2007.
    Number of citations: 2
21. TJ Ostrand and EJ Weyuker. How to Measure Success of Software Prediction Models. *Proc. Fourth Int'l Wksp on Software Quality Assurance*, Dubrovnik, Croatia, Sept 2007.
    Number of citations: 31
22. TJ Ostrand, EJ Weyuker and RM Bell. Automating Algorithms for the Identification of Fault-Prone Files. *Proc ACM/Int'l Symp Software Testing and Analysis (ISSTA07)*, London, July 2007. (*)
    Number of citations: 44
23. EJ Weyuker, TJ Ostrand and RM Bell. Using Developer Information as a Factor for Fault Prediction. *Proc. IEEE/Third Int'l Promise Wksp*, Minneapolis, May 2007. (*)
    Number of citations: 42
24. TJ Ostrand and EJ Weyuker. An Industrial Research Program in Software Fault Prediction. *Proc. Wksp on Software Testing – From Research to Practice*, Hamburg, Germany, March 2007.
    Number of citations: 1
25. A Avritzer, RG Cole and EJ Weyuker. Using Performance Signatures and Software Rejuvenation for Worm Mitigation in Tactical MANETs. *Proc. ACM Sixth Int'l Wksp on Software and Performance (WOSP2007)*, Buenos Aires, Argentina, Feb 2007.
    Number of citations: 6
26. A Avritzer, TJ Ostrand and EJ Weyuker. Experience Developing Software Using a Globally Distributed Workforce. *Proc. IEEE/First Int'l Conference on Global Software Engineering (ICGSE06)*, Florianopolis, Brazil, October 2006.
    Number of citations: 3
27. EJ Weyuker, TJ Ostrand and RM Bell. Adapting a Fault Prediction Model to Allow Widespread Usage *Proc. IEEE/Second Int'l Promise Wksp*, Philadelphia, September, 2006.
    Number of citations: 10
28. TJ Ostrand and EJ Weyuker. On the Automation of Software Fault Prediction *Proc. IEEE/Testing: Academic and Industrial Conference - Practice and Research*

Techniques (TAIC PART), Windsor, England, Aug 2006.
Number of citations: 2

29. RM Bell, TJ Ostrand and EJ Weyuker. Looking for Bugs in All the Right Places. *Proc. ACM/Int'l Symp on Software Testing and Analysis (ISSTA2006)*, Portland, Maine, July 2006. (*)
Number of citations: 56
30. A Avritzer, A Bondi, M Grottke, KS Trivedi and EJ Weyuker. Performance Assurance via Software Rejuvenation: Monitoring, Statistics and Algorithms *Proc. IEEE Performance and Dependability Symp (PDS)*, Philadelphia, PA, June 2006.
Number of citations: 34
31. EJ Weyuker. Empirical Studies as a Basis for Technology Transfer. Empirical Software Engineering Issues. June 2006. Published as *Lecture Notes in Computer Science 4336, Springer*.
Number of citations: 2

3. Book chapters

1. EJ Weyuker and TJ Ostrand. An Automated Fault Prediction System. In Making Software: What Really Works and Why We Believe It ed. A. Oram & G. Wilson, O'Reilly Media, Oct 2010.
Number of citations: 0

4. Patents

1. Tool for Predicting Fault-Prone Software Files. No. 8151146, 3 April 2012, US.

5. Popular science articles/presentations

EJ Weyuker has given numerous keynote addresses and invited talks; several of these address a wider audience and have elements of popular science character (see Cv for a few examples).

6. Five most cited articles

1. S Rapps and EJ Weyuker. Selecting software test data using data flow information. Software Engineering, IEEE Transactions on, 367-375, 1985.
Number of citations: 1081
2. EJ Weyuker. Evaluating software complexity measures. Software Engineering, IEEE Transactions on 14 (9), 1357-1365, 1988.
Number of citations: 709
3. PG Frankl and EJ Weyuker. An applicable family of data flow testing criteria. Software Engineering, IEEE Transactions on 14 (10), 1483-1498, 1988.
Number of citations: 564
4. EJ Weyuker. On testing non-testable programs. The Computer Journal 25 (4), 465-470, 1982.
Number of citations: 379
5. TJ Ostrand, EJ Weyuker and RM Bell. Predicting the location and number of faults in large software systems, Software Engineering, IEEE Transactions on 31 (4), 340-355, 2005.
Number of citations: 364

C. Publication List of Daniel Sundmark

The following is a list of the journal articles, international conference and workshop articles written by Daniel Sundmark during the last eight years and which have been subject to peer review.

In accordance with the directions, the five most important publications for the project have been marked (*). The indicated number of citations for each publication has been obtained from Google Scholar.

1. Peer-reviewed original articles

1. Daniel Sundmark, Andreas Ermedahl and Johan Stärner. Pinpointing Interrupts in Embedded Real-Time Systems using Hashed Execution Contexts. *IEEE Transactions on Industrial Informatics*, June 2009. (*)
   Number of citations: 2

2. Peer-reviewed conference contributions

2. Kristian Wiklund, Daniel Sundmark, Sigrid Eldh and Kristina Lundqvist Impediments for Automated Testing - An Empirical Analysis of a User Support Discussion Board. *Seventh IEEE International Conference on Software Testing (ICST), Verification and Validation*, Apr. 2004.
   Number of citations: 0
3. Eduard Paul Enoiu, Daniel Sundmark and Paul Pettersson Using Logic Coverage to Improve Testing Function Block Diagrams. *International Conference on Testing Software and Systems (ICTSS)*, Nov. 2013.
   Number of citations: 1
4. Adnan Causevic, Sasikumar Punnekkat and Daniel Sundmark TDDHQ: Achieving Higher Quality Testing in Test Driven Development. *39th Euromicro Conference on Software Engineering and Advanced Applications*, Sept. 2013.
   Number of citations: 0
5. Kristian Wiklund, Daniel Sundmark, Sigrid Eldh and Kristina Lundqvist Impediments in Agile Software Development: An Empirical Investigation. *14th International Conference of Product Focused Software Development and Process Improvement (PROFES)*, June 2013.
   Number of citations: 1
6. Adnan Causevic, Rakesh Shukla, Sasikumar Punnekkat and Daniel Sundmark Effects of Negative Testing on TDD: An Industrial Experiment. *International Conference on Agile Software Development, XP2013*, June 2013.
   Number of citations: 1
7. Eduard Paul Enoiu, Kivanc Doganay, Markus Bohlin, Daniel Sundmark and Paul Pettersson MOS: An Integrated Model-based and Search-based Testing Tool for Function Block Diagrams. *35th International Conference on Software Engineering (ICSE) - First International*

Workshop on Combining Modelling and Search-Based Software Engineering, May 2013.
Number of citations: 2

8. Kristian Wiklund, Sigrid Eldh, Daniel Sundmark and Kristina Lundqvist Can we do useful industrial software engineering research in the shadow of Lean and Agile? *35th International Conference on Software Engineering (ICSE) - First Intl. Workshop on Conducting Empirical Studies in Industry (CESI)*, May 2013.
Number of citations: 0
9. Eduard Paul Enoiu, Daniel Sundmark and Paul Pettersson Model-based Test Suite Generation for Function Block Diagrams using the UPPAAL Model Checker. *International Conference on Software Testing, Verification and Validation (ICST) - Advances in Model Based Testing (A-MOST 2013)*, Apr. 2013.
Number of citations: 3
10. Adnan Causevic, Sasikumar Punnekkat and Daniel Sundmark Quality of Testing in Test Driven Development. *Quality of Information and Communications Technology (QUATIC), 2012 Eight International Conference on the*, Sept. 2012.
Number of citations: 3
11. Ove Sundmark, Daniel Sundmark, Stefan Cedergren and Mats Jackson Performance in the Public Sector - Efficiency and Effectiveness of Payroll Services in Three Municipalities. *Proceedings of the 2012 Performance Measurement (PMA2012) Conference*, July 2012.
Number of citations: 0
12. Holger Kienle, Daniel Sundmark, Kristina Lundqvist and Andreas Johnsen Liability for Software in Safety-Critical Mechatronic Systems: An Industrial Questionnaire. *Proceedings of the 2nd International Workshop on Software Engineering for Embedded System*, June 2012.
Number of citations: 3
13. Adnan Causevic, Daniel Sundmark and Sasikumar Punnekkat Impact of Test Design Technique Knowledge on Test Driven Development: A Controlled Experiment. *International Conference on Agile Software Development, XP2012*, May 2012.
Number of citations: 4
14. Adnan Causevic, Daniel Sundmark and Sasikumar Punnekkat Test Case Quality in Test Driven Development: A Study Design and a Pilot Experiment. *International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, May 2012.
Number of citations: 6
15. Kristian Wiklund, Sigrid Eldh, Daniel Sundmark and Kristina Lundqvist Technical Debt in Test Automation. *In proceedings of the 7th Testing: Academic & Industrial Conference (TAIC-PART)*, Apr. 2012.
Number of citations: 4
16. Sigrid Eldh and Daniel Sundmark Robustness Testing of Mobile Telecommunication Systems: A Case Study on Industrial Practice and Challenges. *In proceedings of the 7th Testing: Academic & Industrial Conference (TAIC-PART)*, Apr. 2012.
Number of citations: 0

17. Daniel Sundmark, Kai Petersen and Stig Larsson An Exploratory Case Study of Testing in an Automotive Electrical System Release Process. *Proceedings of the 6th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2011.
    Number of citations: 3
18. Adnan Causevic, Daniel Sundmark and Sasikumar Punnekkat Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review *International Conference on Software Testing, Verification and Validation (ICST)*, Mar. 2011.
    Number of citations: 16
19. Hongyu Pei-Breivold, Daniel Sundmark, Peter Wallin and Stig Larsson What Does Research Say About Agile and Architecture? *The Fifth International Conference on Software Engineering Advances (ICSEA)*, Aug. 2010.
    Number of citations: 20
20. Adnan Causevic, Daniel Sundmark and Sasikumar Punnekkat An Industrial Survey on Contemporary Aspects of Software Testing. *International Conference on Software Testing, Verification and Validation (ICST)*, Apr. 2010.
    Number of citations: 20
21. Rikard Land, Daniel Sundmark, Frank Lüders, Iva Krasteva and Adnan Causevic Reuse with Software Components - A Survey of Industrial State of Practice. *11th International Conference on Software Reuse*, Sept. 2009.
    Number of citations: 16
22. Adnan Causevic, Iva Krasteva, Rikard Land, Abdulkadir Sajeev and Daniel Sundmark A Survey on Industrial Software Engineering. *Poster session at International Conference on Agile Processes and eXtreme Programming in Software Engineering (XP)*, May 2009.
    Number of citations: 1
23. Daniel Sundmark, Jan Carlson, Sasikumar Punnekkat and Andreas Ermedahl Structural Testing of Component-Based Systems. *Proceedings of the 11th International Symposium of Component Based Software Engineering (CBSE)*, Oct. 2008. (*)
    Number of citations: 0
24. Daniel Sundmark and Henrik Thane Pinpointing Interrupts in Embedded Real-Time Systems using Context Checksums. *Proceedings of the 13th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept. 2008.
    Number of citations: 3
25. Daniel Sundmark, Anders Pettersson, Christer Sandberg, Andreas Ermedahl and Henrik Thane Finding DU-Paths for Testing of Multi-Tasking Real-Time Systems using WCET Analysis. *Seventh International Workshop on Worst-Case Execution Time Analysis, (WCET)*, July 2007. (*)
    Number of citations: 1
26. Anders Pettersson, Daniel Sundmark, Henrik Thane and Dag Nyström Shared Data Analysis for Multi-Tasking Real-Time System Testing. *In proceedings of IEEE Second International Symposium on Industrial Embedded Systems (SIES)*, July 2007.
    Number of citations: 4

27. Sigrid Eldh, Hans Hansson, Sasikumar Punnekkat, Anders Pettersson and Daniel Sundmark A Framework for Comparing Efficiency, Effectiveness and Applicability of Software Testing Techniques. *Testing: Academic and Industrial Conference (TAIC-PART)*, Aug. 2006. (*)
Number of citations: 22

3. Review articles

4. Book Chapters

28. Hans Hansson, Mikael Sjödin, Thomas Nolte and Daniel Sundmark Real-Time in Networked Embedded Systems. In *Embedded Systems Handbook, Second Edition*, CRC Press, July 2009.
Number of citations: 0

5. Patents

6. Open access computer programs that you have developed

7. Popular science articles/presentations

Invited as guest speaker on the topics of software engineering and software testing by many industries and organizations. See CV for details.

8. Five Most Cited Publications

(1) Henrik Thane, Daniel Sundmark, Joel Huselius and Anders Pettersson Replay Debugging of Real-Time Systems Using Time Machines. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2003. (*)
Number of citations: 48

(1i) Sigrid Eldh, Hans Hansson, Sasikumar Punnekkat, Anders Pettersson and Daniel Sundmark A Framework for Comparing Efficiency, Effectiveness and Applicability of Software Testing Techniques. *Testing: Academic and Industrial Conference (TAIC-PART)*, Aug. 2006.
Number of citations: 22

(iii) Henrik Thane, Anders Pettersson and Daniel Sundmark The Asterix Real-Time Kernel. *13th Euromicro International Conference on Real-Time Systems, Industrial Session*, June 2001.
Number of citations: 22

(iv) Hongyu Pei-Breivold, Daniel Sundmark, Peter Wallin and Stig Larsson What Does Research Say About Agile and Architecture? *The Fifth International Conference on Software Engineering Advances (ICSEA)*, Aug. 2010.
Number of citations: 20

(v) Adnan Causevic, Daniel Sundmark and Sasikumar Punnekkat An Industrial Survey on Contemporary Aspects of Software Testing. *International Conference on Software Testing,*

Verification and Validation (ICST), Apr. 2010.
Number of citations: 20

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Name of applicant
Weyuker, Elaine

Date of birth
451122-

Title of research programme

Appendix N

Budget and research resources

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se

N. Budget and Resources

The following table summarizes the budget for the proposed project:

Type of cost	**2015**	**2016**	**2017**	**2018**	**total**
Salary Elaine Weyuker (40%)	919	947	975	1005	3847
Salary Daniel Sundmark (50%)	630	649	669	689	2637
Travel	68	68	68	68	272
Equipment	20	5	15	5	45
Total	1637	1669	1727	1767	6800

VR is proposed to cover 100% of the project costs. The salary costs are based on the actual current salaries for Prof. Weyuker and Dr. Sundmark. We assume a salary increase of 3% per year for the duration of the project. The LKP (estimated to 53%) and overheads (calculated as 57% of the direct salary costs) are included in the salary costs. The budgeted overheads are the ones charged to research projects at MDH as of the date of writing, and they include costs for office space, local administration at the school, and central university administration. Travel costs are for visiting conferences and similar.

Current Research Funding of the Group

Between September 2013 and August 2014, Elaine Weyker holds a 50% position as visiting professor at Mälardalen University. This position is funded by a grant from the Knowledge Foundation (KKS). This project application aims at an extension and an intensification of the partnership between Prof. Weyuker and Mälardalen University.

Daniel Sundmark is currently active in the following research projects:

- ATAC - Advanced Test Automation for Complex and Highly-Configurable Software-intensive Systems (Oct. 2011 – Sept. 2014), ITEA2/Vinnova, Main applicant Paul Pettersson, 3.85 MSEK. Daniel Sundmark serves as node manager for Mälardalen University and is funded 10% by the grant. This project ends before the proposed starting date of the project proposed here.
- PINT - Parallelization of Integration Tests (Sept. 2013 – Aug. 2015), Vinnova FFI, Main applicant Daniel Sundmark, 2.1 MSEK. Daniel Sundmark is funded 10% by the grant. The last eight months of this projct overlaps with the project proposed here.
- TOCSYC - Testing of Critical System Characteristics (Sept. 2013 – Aug. 2018), KKS, Main applicant Paul Pettersson, 30.1 MSEK. Daniel Sundmark is funded 40% by the grant.

New Project Applications

Elaine Weyuker and Daniel Sundmark are both currently involved in a VR framework grant application on performance testing, led by Prof. Paul Pettersson at Mälardalen University (deadline April 9th).

VETENSKAPSRÅDET
THE SWEDISH RESEARCH COUNCIL

Kod
2014-48274-115757-57

Dnr
2014 -

Name of applicant
Weyuker, Elaine

Date of birth
451122-

Reg date
2014-04-09 11:17:49

Project title
EXACT - Experimental Analysis of the Coupling Effect Hypothesis in Software Testing

NT

Research grants NT April 9, 2014
Project Research Grant
*Övrigt generellt
Mälardalens Högskola

NT-2, NT-13

Appendix S

Signatures

2015	2016	2017	2018	2019
1637	1669	1727	1767	

2015-01-01 -- 2018-12-31

Applicant

Date

Head of department at host University

Clarifi cation of signature

Telephone

A signature on the application is required not only from the applicant but also from the authorised representative of the administrating organisation/institution or equivalent (normally the head of the department or establishment where the research is to be conducted). The signature confirms that the department can accommodate the proposed research, position or equipment; that the costing in the application is approved for the department's part, that any proposed experimentation on human or animal subjects has been reported, and that the applicant has reported any secondary occupations and commercial ties (s)he may have, and nothing inconsistent with good research practice has thereby emerged. The applicant must have discussed these conditions with the representative of the administrating organisation/institution or equivalent before the latter approves and signs the application.

NOTE the exceptions!
Please note that a different procedure applies for the following types of grants:

- Infrastructure- Operation Grants
- Infrastructure- Grants for Expensive Equipment
- Infrastructure- Grants for Large Databases
- Framework Grant SIMSAM
- International Recruitment

For these types of grants the vice-chancellor, instead of the head of department, must sign Appendix S. Thereby (s)he confirms the commitments of the administrative organization. The details are specified in the Call for Proposals, published at www.vr.se.

Vetenskapsrådets noteringar
Kod

2014-48274-115757-57

Vetenskapsrådet, Box 1035, SE-101 38 Stockholm, tel. +46 (0)8 546 44 000, vetenskapsradet@vr.se