Teaching Predictive Modeling to Junior Software Engineers Seminar Format and Its Evaluation

Katsiaryna Labunets
University of Trento
Trento, Italy
katsiaryna.labunets@unitn.it

Andrea Janes
Free University of Bozen-Bolzano
Bolzano, Italy
andrea.janes@unibz.it

Michael Felderer
University of Innsbruck
Innsbruck, Austria
michael.felderer@uibk.ac.at

Fabio Massacci
University of Trento
Trento, Italy
fabio.massacci@unitn.it

Research Context and Questions

"If you went to bed last night as an industrial company, you're going to wake up today as a software and analytics company" (c) Jeff Immelt, CEO of General Electrics

– Machine learning techniques are widely used in research, but their usage is still limited in industry <= software developers lack for relevant knowledge.
– Software developers who work in company do not have time to attend long courses with lectures on prediction modeling.

Research Questions

RQ1: *How could a seminar to teach prediction models be designed to promote the use of the method among novices?*
RQ2: *How effective is the chosen format of the seminar to teach prediction models?*

Solution

Two-day seminar to teach machine learning-based predictions in software engineering. The seminar is based on the research paper by Scanariato et al. "Predicting vulnerabile software components via text mining" [1].


This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TSE.2014.2340398

Predicting Vulnerable Software Components via Text Mining

Riccardo Scandariato, James Walden, Aram Hovsepyan and Wouter Joosen

Abstract—This paper presents an approach based on machine learning to predict which components of a software application contain security vulnerabilities. The approach is based on text mining the source code of the components. Namely, each component is characterized as a series of terms contained in its source code, with the associated frequencies. These features are used to forecast whether each component is likely to contain vulnerabilities. In an exploratory validation with 20 Android applications, we discovered that a dependable prediction model can be built. Such model could be useful to prioritize the validation activities, e.g., to identify the components needing special scrutiny.

Index Terms—Vulnerabilities, prediction model, machine learning

1 Introduction

Verification and validation (V&V) techniques like security hotdog, code review and formal verification are becoming effective means to reduce the number of post-release vulnerabilities in software products [1]. This is an important achievement, as fixing a bug after the software has been released can cost much more than resolving the issue at development time [2]. However, V&V is not inexpensive. An early estimation assessed that V&V adds up to 30% to development costs [3]. Indeed, it has been noted that "it is possible to spend too much on software quality" [4]. In this respect, being able to predict where the vulnerabilities are likely to show up in a code base provides the opportunity to point V&V activities in the right direction and to manage and optimize cost.

Machine learning techniques have been used as means to build such prediction models [5, 6]. Prior research has primarily focused on finding relationships between vulnerabilities and properties of the source code–like cyclomatic complexity–or aspects of the code development process–like developer activity [5]. In those cases, a model is built based on the intuition that, for instance, a more complex software component (or one that is modified very often) is more likely to contain security issues. Less obvious intuitions like vulnerable software are being related to the use of certain libraries also turned out to be correct [6]. In the above examples, the choice of the features that are used as predictors is determined by the expectations of a knowledgeable individual.

In our work, we investigated a technique that relies less on a particular underlying axiom. Starting from the observation that a programming language is a language after all (like English) and that syntax tokens equate to words, we set out to analyze the source code by means of text mining techniques, which are commonplace in information retrieval. Text mining applied to source code was introduced by Hata et al. [7] for the prediction of software defects and is here applied to the domain of software vulnerabilities. We use the bag-of-words representation, in which a software component (a Java source file in this paper) is seen as a series of terms with associated frequencies. The terms are the features we use as predictors. Hence, the set of features used for modeling is not fixed or predetermined but rather depends on the vocabulary used by the developers. In this sense, this technique is less constrained or biased by an underlying theory of what is a-priori expected to happen.

As its main contribution, this paper explores the value of a technique backed by text mining and machine learning and applies the technique to a relevant class of applications, thus ensuring a potentially high impact in case of success. The approach presented here is applied to the problem of predicting software vulnerabilities. In particular, we analyzed 20 "apps" for the Android OS platform and followed their evolution over time. In total, we analyzed 182 releases, spanning a period of two years.

We applied the above-mentioned text mining technique in a series of three experiments of increasing complexity. In the first experiment, we focus on the first release of each application. We show that it is possible to build a classifier of good quality that predicts whether a file is vulnerable using term frequencies. In particular, we show that the model has good performance in terms of precision and recall. Informally, precision represents the probability that the classification of a file as vulnerable is correct, while recall represents the probability that the model finds a file which is known to be vulnerable. Therefore, high precision implies that the classification

Copyright (c) 2014 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

Why this paper?

1) one of the authos knew this work and it was easier to replicate it,
2) security is a fascinating topics for the students
   => higher interest to participate in the seminar,
3) the paper was published in IEEE TSE journal
   => this is the high quality study.

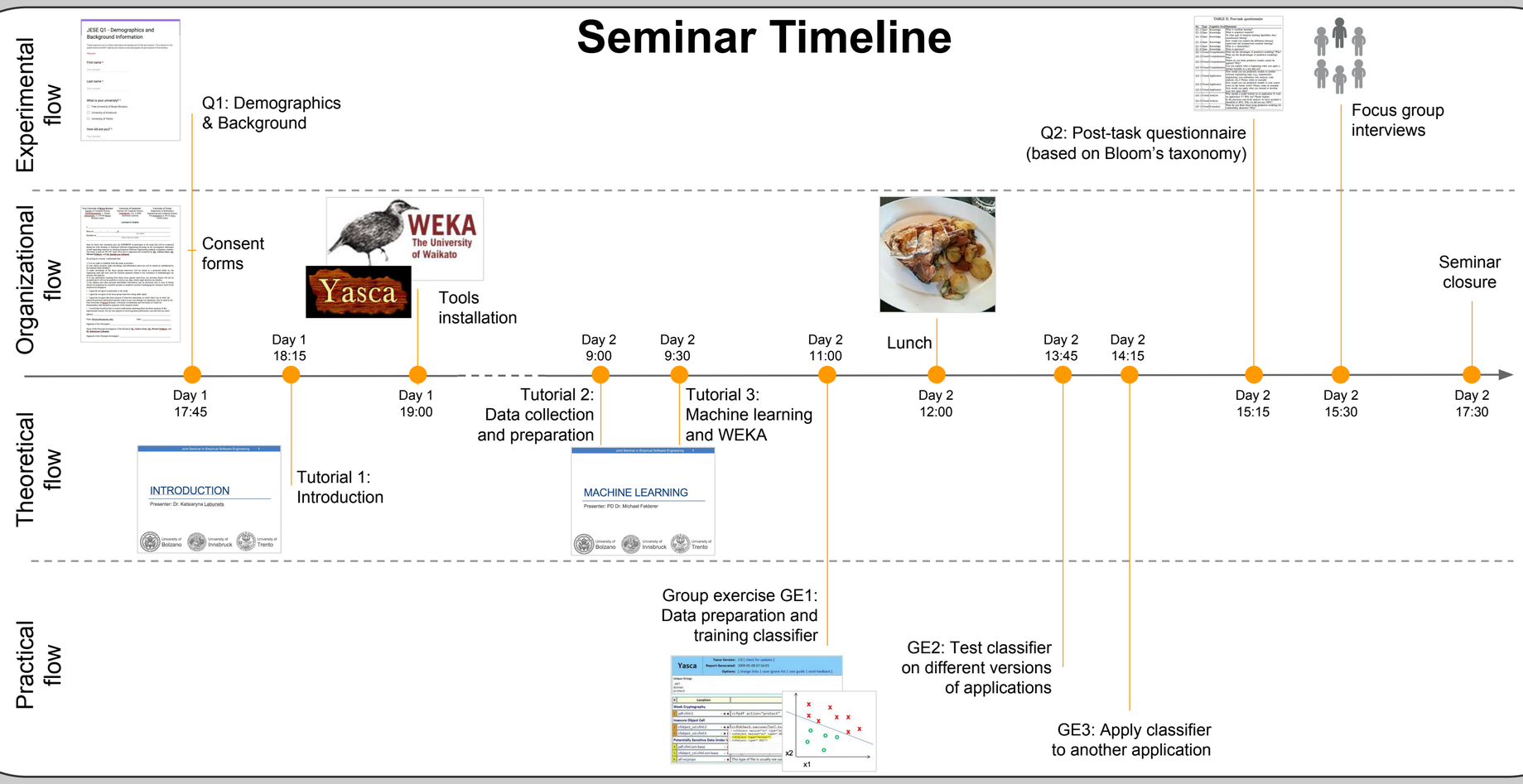
University of Bolzano

University of Innsbruck

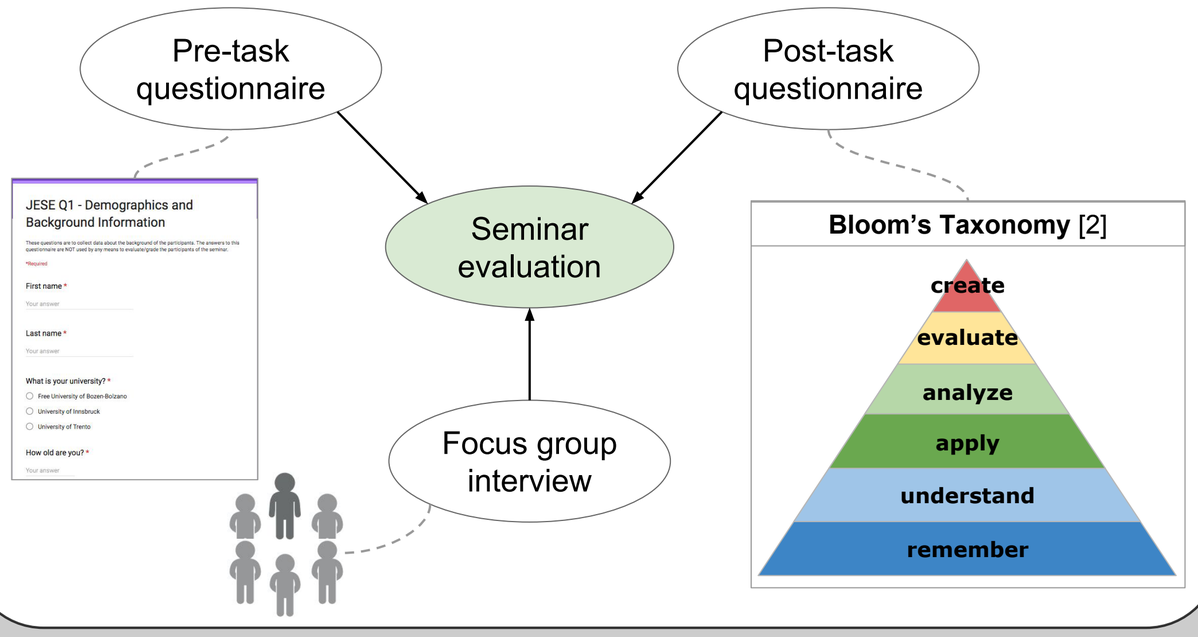
University of Trento

EUROPAREGION EUREGIO
Tirol Südtirol Trentino
Tirolo Alto Adige Trentino

Seminar Timeline



Evaluation



Results

RQ1: Two-day seminar with good balance between theoretical and practical components and based on a research paper as a scenario of the practial part.
Participants appreciated the practical illustration how machine learning techniques can be used to solve a problem that they would never think is possible to apply to.

RQ2: Participants demonstrated high overall quality at the level of Knowledge (78%), Comprehension (72%) and Application (68%), and medium quality of results at the Analysis level (59%).

References

[1] R. Scandariato, J. Walden, A. Hovsepyan, and W. Joosen. "Predicting vulnerable software components via text mining," IEEE T. Software Eng., vol. 40, no. 10, 2014.
[2] B. S. Bloom, "Taxonomy of education objectives: Teh classification of educational goals," 1956.

What to improve

1) The seminar should be at least two full days of work.
2) The seminar should provide a better explanation of "how things work" in prediction modeling as the participants experienced problem in this part.
3) The Analysis level requires better support by the seminar structure as participants showed fair quality of responses at this level. E.g. a more interactive format of the practical exercises can help participants to practice skills related to Application and Analysis levels.