2nd MadSESE Seminar

Programme - Abstracts

March 9th 2017 - Campus Sur Universidad Politécnica de Madrid
http://gregoriorobles.github.io/MadSESE/201703.html

Keynote

Michel R. V. Chaudron is a Professor at the Department of Computer Science and Engineering, Chalmers University and University of Gothenburg.

He was an Associate and Assistant Professor at Leiden University and TU Eindhoven.

Michel has been actively carrying out research in Software Architecting and Component-based Software Engineering, Model-driven development, and UML, and has published over 20 journal papers and over 100 conference papers.

Michel R. V. Chaudron has been involved in several national and international projects and has served on the PC of a number of international Software Engineering conferences.

Software Engineering Artifact in Software Development Process - Linkage Between Issues and Code Review Processes

Dorealda Dalijpaj, Jesús M. González Barahona and Gregorio Robles (URJC)

Researchers working with software repositories, often when building performance or quality models, need to recover traceability links between bug reports in issue tracking repositories and reviews in code review systems. However, very often the information stored in bug tracking repositories is not explicitly tagged or linked to the issues reviewing them. Researchers have to adopt various

heuristics to tag the data. We present two state-of-the-practice algorithms on how to link issues and reviews, and empirically compare the outcome of the two approaches.

Evolution of Permission Requests of Android Apps

Paolo Calciaty (IMDEA Software)

We present a preliminary study to understand how apps evolve in their permission requests across different releases. We analyze over 14K releases of 227 Android apps, and we see how permission requests change and how they are used. We find that apps tend to request more permissions in their evolution, and as previous studies report many of the newly requested permissions are initially overprivileged. Our qualitative analysis, however, shows that the results that popular tools report on overprivileged apps may be biased by incomplete information or by other factors. Finally, we observe that when apps no longer request a permission, it does not necessarily mean that the new release offers less in terms of functionalities.

Comparing linear regression approaches in software engineering

Daniel Rodríguez, Javier Dolado, Javier Tuya and Dietmar Pfahl (UAH, EHU, UniOvi, Univ Tartu)

Linear regression is one of the most known approaches for data analysis in many fields. Linear regression has been routinely applied in software engineering. In this work we show a preliminary approach of the comparison of the frequentist linear regression to the bayesian approach. We comment the problems encountered when applying the bayesian approach in software engineering datasets. We selected a dataset that was amenable for the three approaches analysed: linear regression with least squeares, bayesian with maximum a posteriori, and bayesian with non-informative priors.

Mining GitHub Searching for UML

Miguel Ángel Fernández and Gregorio Robles (URJC)

GitHub is the most used online code platform in the world, with 14 million users and more than 35 million repositories (data from April, 2016). Mining information from these millions of projects and analysing that data is very useful for both researchers and companies. It is shown a methodology for extracting information from these Free/Libre/Open Source Software repositories stored on GitHub, applied to a case study: the search of UML models for quantify and analise its use in this type of projects (Michel R. V. Chaudron, Gregorio Robles et al.). For that, it starts from a database provided by the GHTorrent project (which creates a scalable, queriable, offline mirror of data from the GitHub REST API), and a series of scripts are used for extracting metadata from the repositories in order to look for patterns and/or specific file extensions. Once the interesting projects have been identified through an external process, they are analysed with Perceval, a program which extracts metrics from them.

Measuring gender diversity in open source communities

Daniel Izquierdo, Jesús M. González Barahona and Alberto Pérez (Bitergia)

This talk aims at introducing the audience the topic of gender-diversity in open source communities. This will focus on the main motivation for this analysis, methodology, tooling, results and current work in progress. Some initial results in OpenStack and other projects show that women are barely 10% of the total population in any of them. And their percentage of actual work is even lower being around 9%. The analysis of the Linux Kernel and the Hadoop ecosystem show similar results. This talk will also provide information about the impact of this analysis in some of those communities, feedback retrieved and further work.

A Look into 30 Years of Malware Development from a Software Metrics Perspective

Alejandro Calleja Cortias, Juan Caballero and Juan Tapiador (UC3M and IMDEA)

During the last decades, the problem of malicious and unwanted software (malware) has surged in numbers and sophistication. Malware plays a key role in most of todays cyber attacks and has consolidated as a commodity in the underground economy. We analyzed the source code

of 151 malware samples and obtained measures of their size, code quality, and estimates of the development costs (effort, time, and number of people). In this talk, I will discuss the obtained results, providing quantitative evidences on how malware has evolved since the early 1980s to date from a software engineering perspective.

Model Driven Engineering: some collected experiments

Juan de Lara, Esther Guerra and Jesús Sánchez Cuadrado (UAM, UMU)

Model Driven Engineering (MDE) is a software engineering paradigm that promotes an active use of models in all development phases. This way, models are used to specify, test, verify, simulate and generate code for the final system, among other activities. Model management operations, like model transformations and code generators are hence essential to enable automated model manipulation. While models can be described using general purpose modelling languages, like the UML, it is common to define Domain-Specific Languages (DSLs), tailored to particular application domains.

In this talk, we will present an overview of the work that the Modelling & Software Engineering research group (http://miso.es) of the Universidad Autónoma de Madrid performs in the context of MDE, stressing the role of empirical evaluations. In particular, we will report on works analysing MDE artefacts on the one hand, and user evaluations on the other. Regarding artefact evaluations, we will describe the analysis of a wide collection of meta-models (including OMG specifications), aimed at discovering smells that signal opportunities for using multi-level modelling technologies [1]; and the analysis of a repository of model transformations, with the aim of discovering rule and typing errors [2]. Regarding user evaluations, we will report on the results of evaluating several of our tools for meta-modelling and domain-specific modelling [3, 4].

We will finish the talk with some reflections on the specific problems and challenges for performing empirical studies in the context of MDE.

References:

[1] Juan de Lara, Esther Guerra, Jesús Sánchez Cuadrado: When and How to Use Multilevel Modelling. ACM Trans. Softw. Eng. Methodol. 24(2): 12:1-12:46 (2014)

[2] Jesús Sánchez Cuadrado, Esther Guerra, Juan de Lara: Static Analysis of Model Transformations. IEEE Trans. Softw. Eng. 2017 (in press)

[3] Ana Pescador, Juan de Lara: DSL-maps: from requirements to design of domain-specific languages. Proc. ASE 2016: 438-443.

[4] Nicolás Buezas, Esther Guerra, Juan de Lara, Javier Martín, Miguel Monforte, Fiorella Mori, Eva Ogallar, Oscar Pérez, Jesús Sánchez Cuadrado: Umbra Designer: Graphical Modelling for Telephony Services. Proc. ECMFA 2013: 179-191

From Python to Pythonic

José Javier Merchante and Gregorio Robles (URJC)

Python is an interpreted, interactive, object-oriented programming language. The philosophy of Python emphasized in code readability and write programs in fewer lines of code. Many people choose this language because of the increased productivity it provides.

For this programming language, as for many others, there is always a way to code a task that is more concise, improves its readability, simplicity, and is more optimized. That is what Python community call Pythonic code. In the last years, Python has grown a lot and is one of the most used programming languages, from people introducing in programming to experienced programmers. Many of them are interested in make their code more Pythonic. Fact of that is that in StackOverflow nearly 1 million questions have at least one occurrence of the word Pythonic.

There are plenty of tools that provide some feedback about the code like PEP8, PyFlakes or PyLint, but none of them provide the mastery of Python a programmer has, or the use of some advanced idioms. In our research we want to build a tool that study the use of Pythonic elements. Identifying which idioms are used in projects, we plan to make some specific learning paths that allow Python programmers to improve their skills and knowledge.

Technical debt management

Carlos Fernández-Sánchez, Juan Garbajosa, Agustín Yage, Jennifer Pérez (UPM)

Technical debt, a metaphor for the long-term consequences of weak software development, must be managed to keep it under control. The main goal of this presentation is to describe the elements required to manage technical debt. The research method used to identify the elements was a systematic mapping, including a synthesis step to synthesize the elements definitions.

Additionally, the rigor and relevance for industry of the current techniques used to manage technical debt were studied. The elements were classified into three groups (basic decision-making factors, cost estimation techniques, practices and techniques for decision-making) and mapped according three stakeholders' points of view (engineering, engineering management, and business-organizational management). The definitions, classification, and analysis of the elements provide a framework that can be deployed to help in the development of models that are adapted to the specific stakeholders' interests to assist the decision-making required in technical debt management and to assess existing models and methods. Finally, challenges and future research are discussed, specially focused on empirical software engineering.

On Software Sustainability

Rafael Capilla y Carlos Carrillo (URJC, UPM)

Software sustainability is a new concept that encompasses different dimensions and ways to estimate the quality of systems at various abstractions levels (i.e., code, architecture, design decisions). Complementarily to technical debt and design debt measures that attempt to identify bad smells in code and architecture, the goal of software sustainability is to suggest new metric or combination of them to estimate how sustainable our systems are in order to extend their longevity and stability of code and architectures as well. This talk focuses on decisions that endure to produce stable designs and meta-models and metrics able to estimate the sustainability of the architecture knowledge as a new category of sustainability metrics.

Test oracles for Java from Javadoc comments

Alessandra Gorla (IMDEA Software)

In this talk I will present a technique that automatically creates test oracles for Java code from Javadoc comments. The technique uses a combination of natural language processing and program analysis. Our implementation, Toradocu, can be combined with a test input generation tool to automatically generate comple effectiveness of automatically-generated test suites, while at the same time it reduces the number of false positives.

# # #

SATToSE 2017

10th Seminar Series on Advanced Techniques & Tools for Software Evolution

07-09 June 2017, Madrid, Spain

The goal of SATToSE is to gather both undergraduate and graduate students to showcase their research, exchange ideas, and improve their communication skills.

SATToSE will host invited talks, paper presentations, tutorials, and a hackathon, fostering interactions among participants and stimulating lively debates and discussions around the topics of interest of the event. We expect attendees to be active participants and not just passive listeners. Presenters should be open to and encourage questions and discussions during their talks.

We welcome submissions in topics around:

- tools, processes, analytics, visualizations, and models for managing software evolution
- industrial needs, case studies, and experiences
- empirical studies in evolution and maintenance
- program transformation, refactoring, and migration
- program and/or data reverse engineering
- software artifact co-evolution
- negative research results in software evolution
- software ecosystem evolution

Important dates:

Submission deadline:	April 07
Notification of acceptance:	April 30
Registration deadline:	May 09
Workshop:	June 07 - 09

SATToSE.org/2017