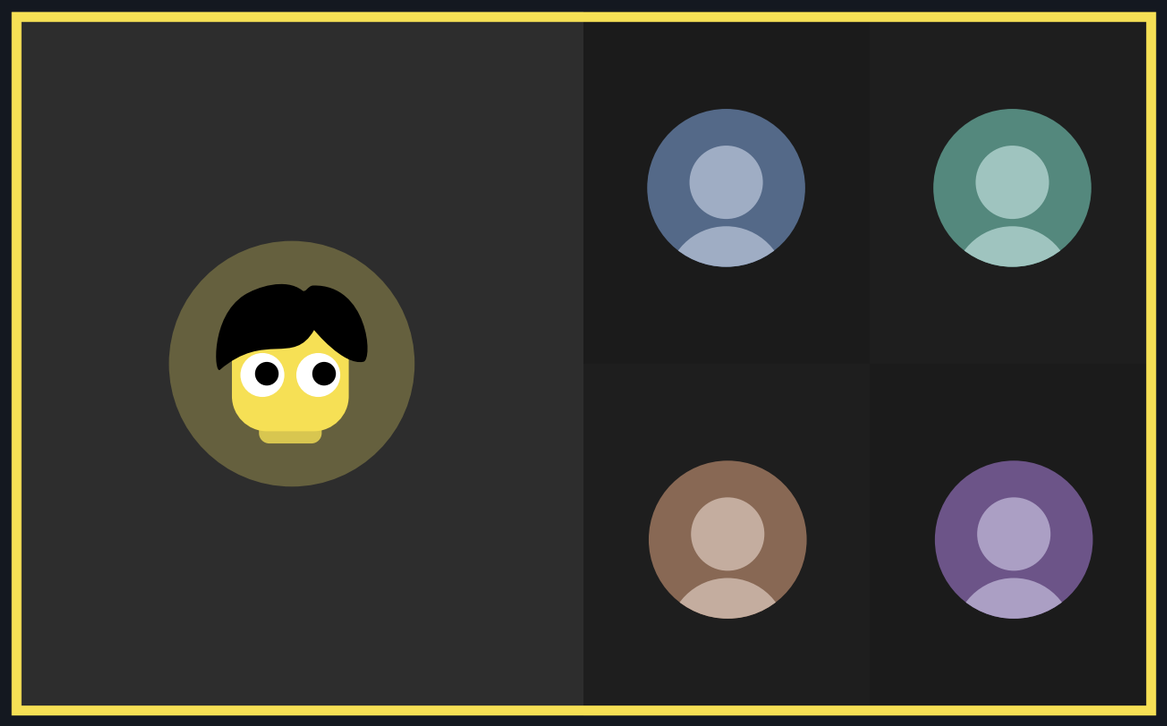
SACLA 2021



THE 50TH ANNUAL CONFERENCE OF THE SOUTHERN AFRICAN COMPUTER LECTURERS' ASSOCIATION

A VIRTUAL CONFFERENCE HOSTED BY
THE ACADEMY OF COMPUTER SCIENCE AND
SOFTWARE ENGINEERING
UNIVERSITY OF JOHANNESBURG

UNIVERSITY OF JOHANNESBURG

BOOK OF ABSTRACTS

SACLA 2021

POST PANDEMIC PEDAGOGY

Publication Chair: Marijke Coetzee

Conference Chair: Wai Sze (Grace) Leung

Programme Chair: Duncan Coulter

Logistics Chair: Deon Cotterrell

SPONSORED BY

IITPSA
Institute of Information Technology Professionals South Africa

TABLE OF CONTENTS

SACLA

50 YEARS' HISTORY

Prof Andre Calitz

The first Southern African Computer Lecturers' Association (SACLA) conference was held in 1971. The initial conferences were organised by IBM, the supplier of mainframe computer equipment in South Africa. IBM started an initiative, called "Teach the Teachers", where they brought out lecturers from the United Kingdom to give a week's classes to everyone who was embarking at that time on teaching Computer Science. The annual SACLA conferences from 1974 were organised by the Computer Science department at South African universities, rotating annually with an inland conference, followed by a coastal university organising the conference the following year. The papers presented at the annual conference were subjected to double-blind peer reviews and published in the annual SACLA Conference proceedings.

In 1984, at my first SACLA conference in Hluhluwe in Natal, a heated debate took place at the AGM between academics from universities and Technikons. The university academics wanted SACLA papers to focus on Computer Science and indicated that SACLA would not focus on the 'practical' teaching as done at the Technikons. The IT academics from the various Technikons in South Africa then formed their own academic body, called the Technikon Computer Lecturers Association (TECLA). The members of TECLA met annually at a similar conference to SACLA, discussing Information Technology (IT) diploma curriculum programmes, industry liaison and related topics. TECLA ceased to exist in 2005, when the South African Government merged Technikons with selected universities and introduced the current Higher Education landscape. Academics from the Comprehensive Universities and Technical Universities teaching computer related programmes re-joined SACLA during the 2000's.

During the 1990's-2000's, the SACLA conferences focused on educational issues, introduction of new concepts and technologies in the curricula and the teaching of Computer Science (CS) and Information Systems (IS). At the 2012 SACLA conference AGM, it was decided that the 2013 conference would take place in Botswana and transferred the SACLA funds to the University of

Botswana. The 2013 SACLA conference however, did not take place and no contingency plans were in place. In 2014, Prof Andre Calitz revived SACLA, formalised the constitution, management structure and financial banking arrangements. A SACLA management committee was established, with a formal President, Treasurer, Secretary and the Chairs of the past and following conferences. The conference arrangements and hosting, the appointment of an annual Conference Chair, conference organisers and a formal international paper review panel, were established and documented. A HoD Colloquium at the annual SACLA conference was further introduced, as well as the publishing annual conference proceedings and selected papers in a Springer accredited journal publication.

The SACLA Constitution and conference documentation for the past years are available at www.sacla.org.za. The official SACLA mailing list is managed by Prof Mac van der Merwe from UNISA. The Institute of IT Professionals of South Africa (IITPSA) have become a major sponsor of the annual SACLA conferences and joint custodians, with SACLA and SAICSIT of the South African Computing Accreditation Board (SACAB). The COVID-19 pandemic forced the 2020 SACLA conference organisers, Rhodes University, to host the 49th SACLA conference virtually, followed by the 50th SACLA conference in 2021, organised by the University of Johannesburg. The 2022 SACLA conference will be hosted by Stellenbosch University, hopefully in a hybrid format.

Prof Andre Calitz
SACLA President

Common Code Writing Errors Made by Novice Programmers: Implications for the Teaching of Introductory Programming

Mokotsolane Ben Mase and Liezel Nel

University of the Free State,
South Africa
masemb@ufs.ac.za, nell@ufs.ac.za

Novices tend to make unnecessary errors when they write programming code. Many of these errors can be attributed to the novices' fragile knowledge of basic programming concepts. Programming instructors also find it challenging to develop teaching and learning strategies that are aimed at addressing the specific programming challenges experienced by their students. This paper reports on a study aimed at (1) identifying the common programming errors made by a select group of novice programmers, and (2) analysing how these common errors changed at different stages during an academic semester. This exploratory study employed a mixed-methods approach based on the Framework of Integrated Methodologies (FraIM). Manual, structured content analysis of 684 programming artefacts, created by 38 participants and collected over an entire semester, lead to the identification of 21 common programming errors. The identified errors were classified into four categories: Syntax, semantic, logic, and type errors. The results indicate that semantic and type errors occurred most frequently. Although common error categories are likely to remain the same from one assignment to the next, the introduction of more complex programming concepts towards the end of the semester could lead to an unexpected change in the most common error category. Knowledge of these common errors and error categories could assist programming instructors in adjusting their teaching and learning approaches for novice programmers.

Keywords: Novice programmer, common programming errors, CS1, computer science education

Mapping Computational Thinking Skills to the South African Secondary School Mathematics Curriculum

Karen Bradshaw and Shannon Milne

Rhodes University,
South Africa
k.bradshaw@ru.ac.za

Computational thinking (CT) is gaining recognition as an important skill for learners in both Computer Science (CS) and several other disciplines, including mathematics. In addition, researchers have shown that there is a direct correlation between poor mathematical skills and the high attrition rate of CS undergraduates. This research investigates the use of ten core CT skills in the South African Grades 10 to 12 mathematics curriculum by mapping these skills to the objectives given in each of the topics in the curriculum. The artefact developed shows that all the core CT skills are required in the curriculum. Given this mapping, future research on interventions to develop these skills through mathematics at secondary school, should result in school leavers with a better mathematical grounding.

Keywords: Computational Thinking, Mathematics Curriculum, Computer Science.

Mapping the problem-solving strategies of novice programmers to Polya's framework: SWOT analysis as a bottleneck identification tool

Pakiso J. Khomokhoana and Liezel Nel

University of the Free State,
South Africa
khomokhoanap@ufs.ac.za, nell@ufs.ac.za

The development of problem-solving skills continues to be a challenge in various disciplines including Computer Science. In this study, we used the principles of the Decoding the Disciplines (DtDs) paradigm to better understand the mental processes that novice programmers follow while answering source code comprehension (SCC) related questions. This understanding can be fundamental in helping novices to overcome problem-solving related challenges. While focusing on step 1 of the DtDs paradigm, the aim of this study was threefold: Firstly, we explored the problem-solving strategies utilised by novice programmers while attempting to answer SCC related questions. Secondly, the identified problem-solving strategies were mapped onto Polya's four problem-solving steps. Finally, we utilised a SWOT analysis as a tool for the identification of problem-solving related learning bottlenecks. This study utilised an integrated methodological approach where data was collected by means of asking questions, observations, and artefact analysis. Thematic analysis of the collected data revealed a range of problem-solving strategies utilised by novice programmers during various SCC tasks. These strategies were then mapped onto Polya's problem-solving steps. Based on a SWOT analysis of these strategies, we were able to identify six problem-solving bottlenecks which point to difficulties not sufficiently addressed in introductory CS courses.

Keywords: Decoding the disciplines, problem solving, source code comprehension, novice programmers, computer science education, Polya's framework, SWOT analysis.

Minimising Tertiary Inter-Group Connectedness over Successive Rounds

Andrew Broekman and Linda Marshall

University of Pretoria,
South Africa
andrew.broekman@up.ac.za, lmarshall@cs.up.ac.za

Rocking the boat is a teaching strategy to rapidly teach tertiary computer science students the required group and communication skills for software engineering. This strategy proposes the introduction of high-risk factors into the group dynamics over short time periods. Group instability is regarded as a risk factor. It is introduced by reshuffling groups between successive rounds. The main examples of allocation methods applied during reshuffling include random allocation, academic standing, participation level and Belbin roles. The reshuffling of groups should ensure that subsequent groups remain heterogeneous with regards to contact between students, that is minimise the inter-group connectedness. Current group formation methods and related software do not focus on the inter-group connectedness over successive group allocations. The construction and tracing of groups by hand to ensure a minimum inter-group connectedness is time-consuming and prone to error. This paper provides a genetic algorithm from the subset of evolutionary algorithms to minimise inter-group connectedness. The proposed algorithm reduces the time and error in constructing groups based on random allocation over successive rounds.

Keywords: Group Formation, Teaching Teamwork, Evolutionary Algorithms, Genetic Algorithm, Optimisation

Project-based learning guidelines for IT higher education

JT Janse van Rensburg

North-West University,
South Africa
jt.jansevanrensburg@nwu.ac.za

It is challenging to adapt higher education at the same pace as the fast-changing nature of the information technology (IT) sector. Creative ways are needed to accommodate the continuous changes in the IT industry, without needing to change the structure of the curriculum. As the IT industry comprises of project teams in project environments, a suitable instructional approach for IT higher education is project-based learning (PBL). A PBL strategy allows for the instruction of current technologies and contributes to skills development that is required in the IT industry. General guidelines and characteristics for PBL are followed across the board for all disciplines on a trial and error basis. Some guidelines may not be directly applicable to IT higher education, while other needed guidelines do not exist because they are domain-specific. The purpose of this paper is to present guidelines for PBL that is relevant to IT higher education. The guidelines are formulated using popular characteristics of PBL from literature and adapting them to the context of IT projects. Additional guidelines are added based on specific requirements of IT projects that are relevant to professional practice. A literature review matrix is presented as evidence of sources where the guidelines were adapted from. The guidelines can be used by educators who find it challenging to assess whether they are implementing a suitable PBL approach. Additionally, following the guidelines may contribute towards bridging the skills gap between IT higher education and the IT industry.

Keywords: Project-based learning, information technology, IT higher education, guidelines

Teaching in a Time of Uncertainty – A Practical Guide

Geoffrey Dick

St John's University,
United States of America
gfdick@aol.com

In March of last year many of us were required to move our classes to an online mode – often with very little notice. For many faculty and students this was the first time we found ourselves in a learning environment that was not only unexpected but for many, not what they wanted. Special measures were called for, and unfortunately, it seems very possible we may need to continue to move between face-to-face and online classes as the pandemic ebbs and flows. Drawing on many years of diverse experience in online teaching, this paper provides some guidance for those inexperienced in online teaching in regards to the pedagogical changes that are necessary – what can work and what is likely to be problematic. The paper is written in two sections – the first is focused using the Learning Management System to facilitate a quick move to online learning and in the second part covers some of the longer-term difficulties that should be considered as we progress or move to the new environment.

Keywords: online classes, pedagogy, LMS, online assessment, pandemic, covid19

Ten years in the Trenches of a Doubtful Science: An Authoethnographic Investigation of Five Challenges of Teaching in Information Systems

Daniel le Roux

Stellenbosch University,
South Africa
dbleroux@sun.ac.za

The field of Information Systems (IS) is characterised by a number of properties which combine to create particular challenges for teaching and learning. These properties include its close ties with rapidly advancing digital technology, its interdisciplinary nature and its general lack of a strong, broadly agreed upon theoretical core. In the present study I undertake an autoethnographic investigation of five key challenges for teaching in IS which result from one or more of these properties. To provide theoretical scaffolding for the investigation I adopt Weick's theory of sensemaking and apply it to investigate both my own and my students' processes of sensemaking. I propose that, to effectively navigate these challenges and prepare students for careers in IS, lecturers should aim to develop their students' technical and social skills sets such that they are able to navigate the uncertainty and ambiguity that characterise socio-technical systems in practice. To this end I outline the strategies I have adopted in my attempts to achieve this aim.

Keywords: Information systems, teaching, challenges, autoethnographic, sensemaking

Understanding the Significance of Enterprise Resource Planning Education in Zambia: A Case of an ERP Short Course at University of Zambia

Mampi Lubasi and Lisa F. Seymour

University of Cape Town,
South Africa
mampi.lubasi@gmail.com, Lisa.seymour@uct.ac.za

This paper investigates the significance of ERP Education on postgraduate students who took an ERP short course at the University of Zambia. The paper identifies the benefits of ERP education in Zambia as well as the contextual factors that impacted ERP education. These contextual factors were identified as industry, course and student constraints. Thematic networks showing the benefits of ERP education, the constraints on ERP education and the preliminary relationships are also presented. Universities seeking to integrate ERP systems into their curriculum and organisations seeking to hire ERP graduates would therefore benefit from these findings.

Keywords: ERP systems, ERP education, Inductive approach

Utilizing Computational Thinking in Programming to Reduce Academic Dishonesty and Promote Decolonisation

Suné van der Linde and Janet Liebenberg

North-West University,
South Africa
sune.vanderlinde@nwu.ac.za, janet.liebenberg@nwu.ac.za

Higher education in South Africa has been in the spotlight in the past few years with calls for decolonisation of the curriculum and other matters. We teach a first-year programming module that is challenging to decolonise, since the origin of programming languages is inherently Western. Students often do not resonate with some examples used, let alone abstract concepts of programming in general. During COVID-19, emergency remote teaching and learning were adopted and we had to be mindful of various limitations, such as data usage and bandwidth. We experienced difficulty expanding each student's frame of reference. Furthermore, increased academic dishonesty occurrences were encountered. This paper focuses on contextualising the module content, promoting computational thinking, and reducing academic dishonesty. This was achieved in an action research cycle through enriching our assessment practices by creating a weekly assignment where the principles of computational thinking were applied within a problem-solving learning environment. It was found that most students had positive perceptions about the intervention and their views and experiences are reported.

Keywords: Assessment in programming, Academic misconduct, Decolonisation, Introductory programming, Computational thinking, Assessment, Action Research