**What are the different tools introduced by researchers for security in programming languages?**

1. SDV (static driver Verifier) ships with the Windows Driver Development Kit. Its purpose is to help driver developers find defects earlier in the development cycle and improve the reliability of windows device drivers.
2. Internally, SDV has used SLAM for statically exploring behaviours of programs. The SDV/SLAM system is one of the major success stories of verification technology.
3. Corral accepts programs in an intermediate verification language called Boogie. In addition to the usual answers of “verified” or “Bug found”, corral can also give up when it hits a user-supplied bound on the number of loop iterations to be explored.
4. However, when we applied SDV-corral combination in a production environment, the number of false defects and missed defects were both at an unacceptable level.
5. Corral being accepted as the engine that powers SDV in the Windows 8.1 release, replacing SLAM engine that had been used inside SDV for the past decade.

---------------------------------------------------------------------------------------------------------------------------

1. Software testing

**PRADA: Prioritizing Android Devices for Apps by Mining Large-Scale Usage Data**

The heavily fragmented distribution of Android devices makes it challenging to select a few

major device models out of thousands of models available on the market. Currently app

developers usually rely on some reported or estimated general market share of device models. However, these estimates can be quite inaccurate, and more problematically, can be Irrelevant to the particular app under consideration. To address this issue, we propose **PRADA**, the first approach to **pr**ioritizing **A**ndroid **d**evice models for individual **a**pps, based on mining large-scale usage data. PRADA adapts the concept of operational profiling (popularly used in software reliability engineering) for mobile apps – the usage of an app on a specific device model reflects the importance of that device model for the app. PRADA includes a collaborative filtering technique to predict the usage of an app on different device models, even if the app is entirely new (without its actual usage in the market yet), based on the usage data of a large collection of apps.

-------------

**Measuring Code Behavioural Similarity for Programming and Software Engineering Education**

In this paper, we envision that the capability of quantifying behavioural similarity between programs is helpful for teaching and learning programming and software engineering, and propose three metrics that approximate the computation of behavioural similarity Specifically, They lever-age random testing and dynamic symbolic execution to generate test inputs, and run programs on these test inputs to compute metric values of the behavioural similarity.

---------------

**Advances in Unit Testing: Theory and practice**

Parameterized unit testing, recent advances in unit testing, is a new methodology extending the previous industry practice based on traditional unit tests without parameters. A parameterized unit test is simply a test method that takes parameters, calls the code under test, and states as sections. Parameterized unit testing allows the separation of two testing concerns the specification of external, black-box behaviour (i.e., assertions or specifications) by developers and the generation and selection of internal, white-box test inputs (i.e., high-code-covering test inputs) by tools. PUTs have been supported by various testing frameworks. Various open source and industrial testing tools also exist to generate test inputs for PUTs.

-------------

**To Be Optimal or Not in Test-Case prioritization**

Software testing aims to assure the quality of software under test. To improve the efficiency of software testing, especially regression testing, test-case prioritization is proposed to schedule the execution order of test cases in software testing. Among various test-case prioritization techniques, the simple additional coverage-based technique, which is a greedy strategy, achieves empirical results. Researchers have conducted an empirical study for comparing the optimal technique with the simple additional coverage-based technique. Finally from their empirical study, both the optimal technique and the additional technique significantly outperform the ideal technique in terms of coverage, but the latter significantly outperforms the former two techniques in terms of fault detection. Researchers may need take cautions in pursuing the optimal techniques in test-case prioritization with intermediate goals.

------------

**Program-input generation for testing database applications using existing database states**

Testing is essential for quality assurance of database applications. To cover a specific program-code portion (e.g., block), appropriate program inputs also need to be generated for the given existing database state. To address this issue, in this paper, they proposed a novel approach that generates

Program inputs for achieving high code coverage of a database application, given an existing database state. In this approach they used symbolic execution to track how program inputs are transformed before appearing in the executed SQL queries and how the constraints on query results affect the application’s execution. One significant challenge in this problem context is the gap between program-input constraints derived from the program and from the given existing database state; satisfying both types of constraints is needed to cover a specific program-code portion.

-------------

**Cooperative Software Testing and Analysis: Advances and Challenges**

In this paper they said that in earlier years to maximize the value of software testing and analysis, they have proposed the methodology of cooperative software testing and analysis to enable testing and analysis tools to with their users, and enable one tool to cooperate with another tool. Such cooperations are motivated by the observation that a tool is typically not powerful enough to address complications in testing or analysis of complex real-world software, and the tool user or another tool may be able to help out some problems faced by the tool. To enable tool-human or tool-tool cooperation, effective mechanisms need to be developed 1) for a tool to communicate problems faced by the tool to the tool user or another tool, and 2) for the tool user or another tool to assist the tool to address the problems. Such methodology of cooperative testing and analysis

forms a new research frontier on synergistic cooperations between humans and tools along with cooperations between tools and tools. This article presents recent example advances and challenges on cooperative testing and analysis.

------------

**Guided Test Generation for Database Applications via Synthesized Database Interactions**

In this article they explained about testing database methods. Testing database applications typically requires the generation of tests consisting of both program inputs and database states. A testing technique called Dynamic Symbolic Execution (DSE) has been proposed to reduce manual effort in test generation for software applications. Applying DSE to generate tests for database applications faces various technical challenges. For example, the database application under test needs to physically connect to the associated database, which may not be available for various reasons.

The program inputs whose values are used to form the executed queries are not treated symbolically, posing difficulties for generating valid database states or appropriate database states for achieving high coverage of query-result-manipulation code. So, they have proposed an approach called synDB that synthesizes new database interactions to replace the original ones from the database application under test like query-construction constraints, query constraints, database schema constraints, and query-result-manipulation constraints. For .NET from Microsoft Research they have applied a state of the art DSE engine called Pex to generate both program inputs and database states. By their approach of tests can achieve higher code coverage than existing test generation for database applications.

------------

**Detecting high-quality posts in community question answering sites**

In this article they explained about how Community question answering (CQA) has become a new paradigm for seeking and sharing information. In CQA sites, users can ask and answer questions, and provide feedback to these questions/answers. In this article, they proposed the early detection of high-quality CQA questions/answers. Such detection can help discover a high-impact question that would be widely recognized by the users in these CQA sites, as well as identify a useful answer that would gain much positive feedback from site users. Specifically, they see the post quality from the point of view of the voting result. Initially, our key instinct is that the voting score of an answer is emphatically decidedly related with that of its inquiry, and they have checked such relationship in two genuine CQA information sets. Second, equipped with the confirmed relationship, they proposed a group of calculations to mutually distinguishing the top notch inquiries and answers not long after they are posted in the CQA destinations. They have directed broad exploratory assessments to show the adequacy and productivity.

--------------

**Educational Programming Systems for Learning at Scale**

This article says that learning programming at scale underlies computer science education ranging from basic programming to advanced software engineering topics. There are strong needs of providing effective system supports for learning programming at scale. There are some systems which supports students to write programs via an online integrated development environment. To aim for such effective system supports for learning programming at scale, some research teams from peking university have developed two systems one is POP (denoting Peking University Online programming System) and another one is POJ (denoting Peking University Online Judge System).

These two systems have achieved high impact among students around the world. In this paper, they have presented the overview of the two systems, along with their on going and future work on extending the systems for achieving higher effectiveness in supporting learning programming at scale.

--------------

**Tutorial: Text Analytics for Security**

In this article they have mentioned that computing systems that make security decisions often fail to take into account human expectations. This failure occurs because human expectations are typically drawn from in textual sources and are hard to extract and codify. As this article says that researchers in security and software engineering have begun using text analytics to create initial models of human expectation. The researchers have used natural language processing (NLP) and text mining in applying text analytics to security problems.

-------------

**Educational Software Engineering: Where Software Engineering, Education, and Gaming Meet**

In this article they have defined that the subfield of educational software engineering , which develops software engineering technologies for general educational tasks, going beyond educational tasks for software engineering. In this subfield, gaming technologies often play an important role together with software engineering technologies. Instructive programming designing can and will contribute critical arrangements to address different basic difficulties in instruction particularly

MOOCs, for example, programmed reviewing, clever mentoring, issue era, and literary theft identification. In this position paper, they defined educational software engineering and illustrate Pex for Fun (in short as Pex4Fun).

-------------

**Relation Extraction for Inferring Access Control Rules from Natural Language Artifacts**

In this paper they have mentioned that over forty years of use and refinement, access control, often in the form of access control rules (ACRs), continues to be a significant control mechanism for information security. However, ACRs are typically either buried within existing natural language (NL) artifacts or elicited from subject matter experts. To address the first situation, researchers objective is to help engineers who actualize ACRs by deducing ACRs from NL curios. To aid in rule inference, they proposed a methodology that concentrates relations from NL curios such as prerequisites reports. Not at all like existing methodologies, their approach joins systems from data extraction and machine learning. They have built up an iterative calculation to find designs that speak to ACRs in sentences. they seed this calculation with every now and again happening things coordinating a subject–action–asset design all through a report. The calculation then scans for extra mixes of those things to find extra examples. They used their methodology on reports from three frameworks in three spaces: gathering administration, instruction, and medicinal services.

-----------------

**Transferring an Automated Test Generation Tool to Practice: From Pex to Fakes and Code Digger**

In this paper, they reported experiences on successful technology transfer of pex and its relatives from Microsoft Research and lessons gained from over eight years of exploration endeavors by the Pex group as a team with the scholarly world. The quantity of download checks of Pex and its lightweight rendition called Code Digger has achieved many thousands inside maybe a couple years. Pex4Fun which is derived from Pex, an instructive gaming , has accomplished high instructive effects, reflected by the quantity of snaps of the "Ask Pex!" catch . Developed from Pex4Fun, the Code Hunt site has been utilized as a part of a vast programming rivalry.

---------------

**Constructing Coding Duels in Pex4Fun and Code Hunt**

Pex is an automatic white-box test-generation tool for .NET. They have set up that amusements can be based on top of Pex to open the apparatus to understudies and to the overall population. Specifically, they have discharged Pex4Fun and its successor Code Chase as electronic instructive gaming situations for educating and learning programming and programming building. In Pex4Fun and Code Hunt, the principle diversion sort is a coding duel, where a player composes code in a technique to accomplish the same usefulness as the mystery technique execution, in view of criticism gave by the basic Pex tool. They have additionally utilized the diversion sort for rivalries with a huge number of players, and have found that it separates well amongst apprentices and top coders. This tool demonstrates how coding duels in Pex4Fun and Code Hunt can be built and utilized as a part of instructing and preparing programming and programming designing.

--------------

**Software Analytics for Incident Management of Online Services: An Experience**

In this paper researchers have designed a set of novel data-driven techniques and developed an industrial system called the service analysis studio targeting real scenarios like online services become more and more popular, incident management has become a critical task that aims to minimize the service downtime and to ensure high quality of the provided services.SAS has been sent to overall item datacenters and generally utilized by available to come back to work engineers for episode administration.

-------------

**Inferring Project-Specific Bug Patterns for Detecting Sibling Bugs**

Lightweight static bug-detection tools such as FindBugs, PMD, Jlint, and Lint4j detect bugs with the knowledge of generic bug. Every bug patterns , different projects under analysis may have some project specific bug patterns. Such bug patterns are called Project-Specific Bug Patterns. Because of absence of such PSBP information, existing devices more often than not come up short in adequately distinguishing the greater part of this sort of bugs. if some sibling bugs are altered in a fix revision however some others remain, we regard such fix as a fragmented fix. To address such fragmented fixes, we propose a PSBP-based methodology for identifying sibiling bugs and actualize a device called Sibling-Bug Detector (SBD). Given a fix update, SBD first gathers the PSBPs inferred by the fix revision. At that point, in view of the induced PSBPs, SBD identifies their related sibilng bugs in the same venture. To assess SBD, we apply it to seven well known open-source ventures.

------------

**Automatically Identifying Special and Common Unit Tests for Object-Oriented Programs**

Software developers often create common tests and special tests, which exercise common behaviors and special behaviors of the class under test. So, researchers have developed a new approach for automatically identifying special common unit tests for a class without any need of specifications. Given a class, which consequently produce test inputs, recognize normal and unique tests among the created tests. This approach is based on statistical algebraic abstractions, program properties dynamically inferred based on a set of predefined abstraction templates. It uses statistical algebraic abstractions to characterize program behaviors and identify special and common tests.

-------------

**Defining and Measuring Policy Coverage in Testing Access Control Policies**

To encourage overseeing access control in a framework, security officers progressively compose access control arrangements in particular dialects, for example, XACML and utilize a committed programming segment called a Policy Decision Point (PDP). To expand certainty on composed arrangements, certain sorts of approach testing are generally led, which test the PDP with some ordinary demands and check PDP's reactions against expected ones. This paper creates afirst venture toward efficient approach testing by characterizing and measuring arrangement scope

at the point when testing strategies. Researchers have built up a scope estimation instrument to quantify approach scope given an arrangement of XACML strategies and an arrangement of requests. They have built up an instrument for solicitation era, which arbitrarily creates demands

for a given arrangement of strategies, and an apparatus for solicitation lessening, which avariciously chooses an almost insignificant arrangement of solicitations for accomplishing the same scope as the initially produced demands.

------------

**Tool-Assisted Unit-Test Generation and Selection Based on Operational Abstractions**

Unit testing, a common step in software development and its often insufficient to identify defects. Without any prior specifications programmers need to manually verify the ouputs of the execution which is generally impossible. In this paper, they presented the operational violation approach for unit-test generation and selection, a black-box approach without requiring a prior specifications. The approach dynamically generates operational abstractions from executions of the existing unit test suite. These operational abstractions guide test generation tools to generate tests to violate them. The approach selects those generated tests violating operational abstractions for inspection. These selected tests exercise some new behavior that has not been exercised by the existing tests.

--------------

**Checking Inside the Black Box: Regression Testing by Comparing Value Spectra**

Black-box program outputs have been used to characterize program behaviors and they are compared over program versions in traditional regression testing. In this paper, they have presented a new class of program spectra, value spectra, that enriches the existing program spectra family. They compared the value spectra of a program’s old version and new version to detect internal behavioral deviations in the new version. Based on the deviation-propagation call tree, they have proposed two heuristics to locate deviation roots, which are program locations that trigger the behavioral deviations.

-----------------