**Week 5: Review and Analyze Conference Venue Review**

Nate Bachmeier

April 7th, 2019

TIM-8101: Principals of Computer Science

Northcentral University

# Describe the types of papers presented/research conducted at the venue

The European Software Engineering Conference and Symposium on the Foundations of Software Engineering focuses on “recent innovations, trends, experiences and challenges in the field of software engineering (ESEC/FSE, 2018).”

# Describe each track and types of papers that are submitted to each track

* Research Papers
  + A forum for original and unpublished results
* Journal-First
  + A forum to discuss published papers which have not been demonstrated yet
* New Ideas and Emerging Results (NIER)
  + A forum for early stage research and its potential applications
* Demonstrations
  + Discussions around tooling and concrete implementations
* Industry
  + Discussions around the application of software engineering
* A-TEST
  + Discussions around testing software implementations
* Workshop on Automated Specification Inference (WASPI)
  + Methods for discovering data and service contracts
* SWAN
  + A niche track that focuses on continuous integration and A/B testing
* Natural Language for Software Engineering (NL4SE)
  + Extracting artifacts from source code commits and documentation

# Paper 1: Darwinian Data Structure Selection

## What problems did they solve?

Developers often compose software libraries with generic data structures in their default configuration. Optimizing these parameters is tedious and is rarely performed, despite often gaining up to 25% performance improvement for some scenarios (Basios, Li, Wu, Kanthan, & Barr, 2018).

## What was their method?

Each generic data structure is identified by first parsing the code into an abstract syntax tree. These are swapped out with different concrete implementations and its effectiveness measured through a user defined benchmark set. After identifying the best combination, the system will create a pull request to update the original code.

## What are areas of future work/improvement?

There are often multiple ways to solve a software engineering problem, however it is expensive for humans to write multiple implementations. Using this approach, it could be possible to come up with multiple implementations of the same feature.

## What other works does this expand?

The Plastic Surgery Hypothesis predicted that the optimal solution could be found by reusing fragments within the existing code base.

## Why is this important?

This approach can consult external sources to determine optimal solutions.

# Paper 2: FraudDroid: Automated Ad Fraud Detection

## What problems did they solve?

Fraudulent mobile applications attempt to collect additional money from advertisers by forcing or tricking the end users into clicking the advertisement. Detection of this behavior can be complex as it requires inspecting runtime application state (Dong, et al., 2018).

## What was their method?

The application is loaded into an emulator and sent a series of UI actions. As the application replies to the action the UI State View Tree is monitored. Using heuristics, the system can determine the likelihood that malicious behavior has been injected into the tree.

## What are areas of future work/improvement?

The application has proven that it can detect fraudulent activity with very low false positive rate. The next step is to incorporate it into the operating system and prevent these scenarios in the first place.

## What other works does this expand?

Previous detection algorithms have relied on network traffic monitoring, which is challenging due to encryption and obfuscation techniques. Static analysis has also been to investigate how the application is using advertising UI controls.

## Why is this important?

Static analysis cannot detect all forms of dynamic state, especially when the software is intentionally fraudulent.

# Paper 3: Microtask Programming

## What problems did they solve?

It is challenging to gain participation on open source projects as it can often take days or weeks to onboard and make even small changes (LaToza, Lecce, Ricci, Towne, & Van der Hoek, 2018).

## What was their method?

Programming tasks are first broken down into micro tasks and queued for engineers to implement. Each micro task should be around 15 minutes and require no context of the broader application design.

## What are areas of future work/improvement?

## What other works does this expand?

Similar efforts attempt to reduce the onboarding time by providing the relevant context inside of the defect description. Typically, standardized build environments such as Docker containers are also provided to reduce the learning curve.

## Why is this important?

The microtask model removes the need to have even a basic understanding of the broader application context.

# References

Basios, M., Li, L., Wu, F., Kanthan, L., & Barr, E. (2018). Darwinian Data Structure Selection. *Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the FoundationsofSoftwareEngineering(ESEC/FSE’18),*.

Dong, F., Wang, H., L, L., Liu, T., Xu, G., Guo, Y., & Klein, J. (2018). FraudDroid:AutomatedAdFraudDetectionforAndroidApps . *ESEC/FSE ’18, November 4–9*.

ESEC/FSE. (2018). *ESEC/FSE 2018.* Retrieved from Researchr: https://conf.researchr.org/home/fse-2018

LaToza, T., Lecce, A., Ricci, F., Towne, B., & Van der Hoek, A. (2018). Microtask Programming. *IEEE Transactions on Software Engineering* .