A New Taxonomy of Software Testing Approaches
Seeking More Standardized Standards

Samuel Joseph Crawford

crawfs1@mcmaster.ca

Department of Computing and Software, McMaster University

April 22, 2023

Background

McMaster Start Coding and *STaBL Foundation*'s mission is to uplift underprivileged youth by equipping them with 21st-century skills like coding. We have taught lessons to over 30,000 students in the past seven years. We have found creating video games to be particularly engaging. We have developed and tested tools such as the state diagram tool [1], which has been applied successfully by Grade 4-12 and 1st-year students to create single player games. The previous attempt, PAL [2], at creating a multiplayer game / app framework proved too complicated to be reasonably used by students. TEASync is a simplified framework allowing students to create multi-user applications.

Background: The Elm Architecture

- Elm is a pure, strictly-typed functional programming language
- Every Elm program follows a rigid structure known as *The Elm Architecture* (TEA)

Theory: Extending The Elm Architecture to Multiple Clients

- Theorem 1: Two clients running Elm programs having an identical initial model, identical update functions, and processing an identical sequence of messages will end up with an identical model.
- Proof: This can be shown easily by using Elm's pureness property.
- Corollary: Any number of Elm clients having an identical initial model, identical update functions, and processing an identical sequence of messages will end up with an identical model.
- Observation: By ensuring every client receives the same messages in the same order, we effectively have a multi-user application, without writing any application-specific server code.

TEASync Framework Architectural Overview

Example Applications

The following is a simple application which allows multiple users to increment and decrement an integer. Figure **??** shows an example UI for this application. Figure **??** shows an example Pong game made using this framework.

```
type GlobalMsg = Increment | Decrement

type alias GlobalModel = { count : Int }

globalUpdate : GlobalMsg -> GlobalModel -> GlobalModel
globalUpdate msg globalModel =
case msg of
Increment -> { globalModel | count = globalModel.count + 1 }
Decrement -> { globalModel | count = globalModel.count - 1 }
```

Conclusions & Future Work

Leveraging the strictly-typed nature of Elm and its model-view-update architecture, we were able to create a simplified multi-user framework, requiring the programmer to write no server-side code. In addition to the upcoming pedagogical study, future work includes a data modelling extension allowing persistent, structured data, an authentication/authorization scheme, a binary data format to reduce network communication, and curriculum development for a TEASync-based summer camp.

References

[1] P. Pasupathi, C. W. Schankula, N. DiVincenzo, S. Coker, and C. K. Anand, "Teaching interaction using state diagrams," *arXiv preprint arXiv:2207.12701*, 2022.

[2] C. Schankula, E. Ham, J. Schultz, Y. Irfan, N. Thai, L. Dutton, P. Pasupathi, C. Sheth, T. Khan, S. Tejani, *et al.*, "Newyouthhack: Using design thinking to reimagine settlement services for new canadians," in *Innovations for Community Services: 20th International Conference, I4CS 2020, Bhubaneswar, India, January 12–14, 2020, Proceedings 20*, pp. 41–62, Springer, 2020.

Acknowledgments